Chae Hoon Lim
Moti Yung (Eds.)

LNCS 3325

Information Security Applications

5th International Workshop, WISA 2004
Jeju Island, Korea, August 2004
Revised Selected Papers

Springer

Chae Hoon Lim   Moti Yung (Eds.)

Information Security Applications

5th International Workshop, WISA 2004
Jeju Island, Korea, August 23-25, 2004
Revised Selected Papers

Springer

Volume Editors

Chae Hoon Lim
Sejong University
Department of Internet Engineering
98 Gunja-Dong, Kwangjin-Gu, Seoul, 143-747, Korea
E-mail: chlim@sejong.ac.kr

Moti Yung
Columbia University
Department of Computer Science
S. W. Mudd Building, New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Library of Congress Control Number: 2005920313

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

ISSN 0302-9743
ISBN 3-540-24015-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11352440 06/3142 5 4 3 2 1 0

Preface

The 5th International Workshop on Information Security Applications (WISA 2004) was held in Jeju Island, Korea during August 23-25, 2004. The workshop was sponsored by the Korea Institute of Information Security and Cryptology (KIISC), the Electronics and Telecommunications Research Institute (ETRI) and the Ministry of Information and Communication (MIC).

The aim of the workshop is to serve as a forum for new conceptual and experimental research results in the area of information security applications from the academic community as well as from the industry. The workshop program covers a wide range of security aspects including cryptography, cryptanalysis, network/system security and implementation aspects.

The program committee received 169 papers from 22 countries, and accepted 37 papers for a full presentation track and 30 papers for a short presentation track. Each paper was carefully evaluated through peer-review by at least three members of the program committee. This volume contains revised versions of 36 papers accepted and presented in the full presentation track. Short papers were only published in the WISA 2004 pre-proceedings as preliminary versions and are allowed to be published elsewhere as extended versions.

In addition to the contributed papers, Professors Gene Tsudik and Ross Anderson gave invited talks, entitled *Security in Outsourced Databases* and *What does 'Security' mean for Ubiquitous Applications?*, respectively.

Many people have helped and worked hard to make WISA 2004 successful. We would like to thank all the people involved in the technical program and in organizing the workshop. We are very grateful to the program committee members and the external referees for their time and efforts in reviewing the submissions and selecting the accepted papers. We also express our special thanks to the organizing committee members for making the workshop possible. Finally, we would like to thank all the authors of the submitted papers and the invited speakers for enabling an interesting workshop program.

December 2004

Chae Hoon Lim
Moti Yung

Organization

Program Commitee

Co-Chairs:	Chae Hoon Lim	Sejong Univ., Korea
	Moti Yung	Columbia Univ., USA
Members:	Giuseppe Ateniese	Johns Hopkins Univ., USA
	Tuomas Aura	Microsoft Research, UK
	Feng Bao	Institute for Infocomm Research, Singapore
	Colin Boyd	QUT, Australia
	Dario Catalano	ENS, France
	Kijoon Chae	Ewha Womans Univ., Korea
	Gene Itkis	Boston Univ., USA
	Jong Soo Jang	ETRI, Korea
	Yonghee Jeon	Catholic Univ. of Daegu, Korea
	Jonathan Katz	Univ. of Maryland, USA
	Angelos Keromytis	Columbia Univ., USA
	Seungjoo Kim	Sungkyunkwan Univ., Korea
	Yongdae Kim	Univ. of Minnesota at Twin Cities, USA
	Klaus Kursawe	KU Keuven, Belgium
	Taekyoung Kwon	Sejong Univ., Korea
	Chi Sung Laih	National Cheng Kung Univ., Taiwan
	Kwok-Yan Lam	Tsinghua Univ., China
	Chae Ho Lim	Securitymap, Korea
	Kanta Matsuura	Tokyo Univ., Japan
	Refik Molva	Institut Eurecom, France
	Pascal Paillier	Gemplus, France
	Josef Pieprzyk	Macquarie Univ., Australia
	Zulfikar Ramzan	Docomo Labs, USA
	Pankaj Rohatgi	IBM Research, USA
	Bimal Roy	Indian Statistical Institute, India
	Jaechul Ryu	Chungnam National Univ., Korea
	Kouichi Sakurai	Kyushu Univ., Japan
	Diana Smetters	Palo Alto Research Center, USA
	Bulent Yener	Rensselaer Polytechnic Institute, USA
	Okyeon Yi	Kookmin Univ., Korea
	Heungyoul Youm	SoonChunHyang Univ., Korea
	Avishai Wool	Tel-Aviv Univ., Israel
	S.Felix Wu	UC Davis, USA

Table of Contents

Impacts of Security Protocols on Real-Time Multimedia Communications*

Kihun Hong[1], Souhwan Jung[1], Luigi Lo Iacono[2], and Christoph Ruland[2]

[1] School of Electronic Engineering, Soongsil University, 1-1, Sangdo-dong, Dongjak-ku, Seoul 156-743, Korea
kihun@cns.ssu.ac.kr, souhwanj@ssu.ac.kr

[2] Institute for Data Communications Systems, University of Siegen, Germany
{lo_iacono,ruland}@nue.et-inf.uni-siegen.de

Abstract. International Standards Committees like ITU and IETF have produced several security protocols for real-time multimedia communications. But, applying those security mechanisms may results in non-trivial degradation to real-time communications. This paper investigates the impacts of the standard security protocols on the delay, packet overhead, quality of service, and other features of real-time communications. Some of analytical and experimental results show the suitability of the security protocols.

1 Introduction

Emerging Internet applications transmit multimedia content more broadly. Some examples of existing multimedia applications are audio and video conferencing systems, media on demand and pay per view services, groupware for distributed collaborative working and Internet gaming. Internet multimedia communication is characterized by two different communication paths: one is used to exchange signaling data and the other serves for the transport of the media streams. The transport channels between the multimedia endpoints are established by the signaling path. Available signaling standards are the H.323 [1] components H.225.0 [2] and H.245 [3] of the ITU-T and SIP [4] and RTSP [5] of the IETF. H.323 and SIP are mainly used in IP telephony environments whereas RTSP focuses on media on demand services. The transport path supports the data stream transmission. Since the transmitted data has real-time properties, QoS aspects like delay, packet loss and jitter have to be considered. The reason e.g. why media streams are using the transport services provided by UDP instead of the ones offered by TCP is that the reliability and congestion avoiding mechanisms of TCP cause uncertain delay. Security for Internet multimedia communication has to consider both paths, whereas the signaling path does not demand for additional requirements than conventional Internet applications. The transport path instead does demand for additional requirements. The integration of security services into the media stream transmission has certainly an impact on these parameters.

* This work was supported by Korea Research Foundation Grant (KRF-2001-042-E00045).

In this paper we examine different security mechanisms suitable for real-time-oriented IP communication by focusing on the influences on the QoS. First we introduce available standards. Afterwards a list of criteria is presented which is the basis for our investigations and evaluations. In section 3 we present our results concluding what can be realized with the existing approaches and which problems are still open. In section 4 we describe our implementations and make the performance comparison of some security protocols based on communication overhead and error propagation. The results of our investigations are concluded in section 5.

2 Security Standards for Multimedia Communication

Different approaches exist to secure multimedia. IPsec as IP-level security protocol is one possible candidate. SSL/TLS relies on TCP and is therefore not suitable for securing UDP-based multimedia communication. Two more security mechanisms residing at the application layer are the standards H.235 [8] and SRTP [9]. H.235 is part of the umbrella standard H.323 of the ITU-T. SRTP is currently a RFC standard developed within the IETF.

2.1 IPsec

IPsec [6] is standardized within the IETF and provides security services for the Internet Protocol. It is mandatory for IPv6 and optional for IPv4. IPsec offers two security protocols, which can be used independently:

- **Encapsulating Security Payload (ESP)**
  The ESP provides the security services data confidentiality, integrity, anti-replay service, and limited traffic flow confidentiality.
- **Authentication Header (AH)**
  The security services provided by AH are integrity, and anti-replay service.

IPsec can be used to encrypt the media stream (IPsec in transport mode). Within H.323 the H.245 capability exchange messages indicate the support of IPsec. When a media channel is opened the logical channel procedures signals the use of IPsec. Another possibility is to establish a secure channel between two security gateways (IPsec in tunnel mode). In this case the multimedia application is not aware of the SA and therefore no specific signaling is needed. The signaling path (RAS, H.225.0, H.245, SIP, RTSP) can also be secured by IPsec.

2.2 H.235

H.323 [1] comprises a multitude of ITU-T standards with regard to multimedia communication. That makes it a so called umbrella standard. Not only signaling protocols (e.g. H.245, H.225.0) are part of H.323 but also codecs (e.g. G.711, H.261), transport protocols (RTP) and so forth. Finally the H.235 [8] standard describes security services for H.323. H.235 considers security services for both the signaling messages (RAS, H.225.0, and H.245) and the media stream (RTP) transmission. Among the provided security services usually more than one mechanism or algorithm can be used to achieve a security service. This flexibility can result in non-interoperable implementations. Therefore the ITU-T has specified two security profiles which mandate certain mechanisms and algorithms:

- **Baseline Security Profile**
  The baseline security profile provides message authentication/integrity for the signaling path. An option of the baseline security profile is the voice encryption profile, which offers media stream encryption.
- **Signature Security Profile**
  The signature security profile is suggested as an option suited for large environments where the mutual password or symmetric key assignment is not feasible. The signature security profile provides authentication, integrity, and non-repudiation for the signaling messages by using digital signatures. This profile can be used in conjunction with the Baseline Security Profile.

H.235 enables furthermore a so called media anti-spamming to detect flooding attacks.

2.3 SRTP

The Real-time Transport Protocol (RTP) [11] is the most widely used protocol for real-time data. Nearly every Internet multimedia application relies on RTP to packetize the data output by the codecs. RTP itself doesn't provide security mechanisms except the encryption of the packet payload. The Secure RTP (SRTP) [9] instead provides confidentiality and authentication for RTP as well as for RTCP. Furthermore a protection against replay attacks is included. SRTP is defined as a profile of RTP according to the Audio Video Profiles (AVP) [12] and is registered as "RTP/SAVP".

The encryption of SRTP or SRTCP packets is optional whereas the authentication for RTCP is mandatory but optional for RTP.

3 Comparing Criteria

3.1 Provided Security Services

3.1.1 Scope of Protection

IPsec provides authentication of the IP payload and parts of the IP header and encryption of the IP payload. All layers above benefit by IPsec security services. H.235 considers confidentiality and anti-spamming for RTP only. SRTP offers confidentiality, and message authentication and protection against replay-attacks for RTP and RTCP.

3.1.2 Confidentiality

The Encapsulating Security Payload (ESP) of IPsec provides confidentiality for IP datagrams. The format is designed to support a variety of encryption algorithms. The only mandatory cipher is DES operated in the cipher block chaining (CBC) [15] mode.

To encipher RTP packets, H.235 uses following algorithms in CBC-Mode: RC2, DES, and 3DES.

To encrypt the payload of RTP packets in SRTP a pseudorandom keystream is generated, which is XORed with the payload. AES [14] in Segmented Integer Counter (SIC) mode [15, 16, 17] is the default encryption scheme used to generate the keystream. AES in f8 mode [18] is defined additionally. Both modes operate the block cipher in encryption mode only SRTP is extensible to any other transform.

3.1.3 Data Integrity and Message Authentication

To increase the usability of the statistical values provided by RTCP reports it is very important to ensure the integrity of those values like inter-arrival jitter and packet loss rate. The authenticity of control messages like e.g. the BYE packet is even more important. Therefore message authentication and data integrity is not renounceable. Since real-time multimedia systems require a minimal latency of the media packets, in case of bit errors and lossy encoding in RTP payload, it is more useful to use the damaged data than to discard it as unauthentic instead of retransmission. But it is hard to find a difference between forged contents and simple bit errors.

The security protocol AH within IPSec provides data integrity and message authentication for IP packets. The message authentication is based on the use of Message Authentication Codes (MAC). The AH must support two MAC algorithms: HMAC/MD5 (96 bit) and HMAC/SHA-1 (96 bit) [19]. The MAC is calculated over IP header fields that don't change during transmission and payload.

Integrity of RTP and RTCP streams in H.235 is for further study. If an attacker modifies RTP payloads, the receiver decrypts the encrypted portion of the packet and processes the payload using the media codec whether the packet was modified or not. The anti-spamming mechanism described in section 3.1.6 provides a light-weighted RTP packet authentication.

The authenticated portion of a SRTP packet consists of the RTP header followed by the (encrypted) payload of the SRTP packet. Thus, if the header or the payload is modified, SRTP discards the packet. HMAC/SHA-1 [19] is the default algorithm for providing integrity and message authentication in SRTP. The problem of HMAC/SHA-1 is the fixed and large size of the MAC (20 octets). In SRTP it is truncated to the leftmost 32 bit. [19] mentions, that a truncation to less than the half of the generated output of the HMAC increases the possibility to attack the MAC because of the birthday-attack-bound. SRTP doesn't mandate the MAC to 32 bit. Alternatively other MAC algorithms can be used.

3.1.4 Packet Source Authentication and User Authentication

All of the schemes don't provide a method for packet source authentication. None of the analyzed security protocols has a mechanism to provide source authentication in Multicast configurations. Several schemes have been published and suggested to overcome this problem [20, 21, 22, 23], but without success of standardization.

User authentication in IPsec relies on the main mode of the IKE protocol using digital signatures. Though IPsec supports several authentication manners like pre-shared key authentication and public-key encryption, these don't support perfect user authentication.

Authentication in H.235 is accomplished by the utilization of pre-shared secrets. This may be a static password or some other a priori piece of information. The usage of digital certificates is possible.

As in IPsec the user authentication in SRTP depends on a separate protocol. Typical tasks are the negotiation of cryptographic parameters, the mutual authentication, and exchange of session keys and establishment of a security session. For this purpose the Multimedia Internet Keying (MIKEY) [24] protocol is proposed but any other suitable protocols can be applied.

3.1.5 Replay Protection

The AH in IPsec guards against replay attacks. This is realized by maintaining a replay list on the receiver-side, which conceptually contains the indices of all received authenticated packets. In practice, the list can use a sliding window approach, so that a fixed amount of storage suffices for replay protection.

In case of H.235 the receiver does not check the index of an incoming packet against a replay list. Replay protection in H.235 is for further study.

The authentication of the RTP header and payload in SRTP indirectly provides replay protection by authenticating the sequence number. In fact anti-replay is only possible when integrity protection is present. When message authentication is enabled, SRTP protects against such an attack through a replay list corresponding to the one used in IPsec.

3.1.6 DoS Protection

IPsec and SRTP have no countermeasure against message flooding.

H.235 defines a media anti-spamming mechanism. The sender calculates a MAC over the first block of the RTP header and appends it to the RTP packet. The receiver determines quickly whether a RTP packet stems from an unauthorized source. This can also be seen as light-weighted packet authentication.

The techniques used within IPsec and SRTP to provide message authentication are very equivalent to the described media anti-spamming in H.235. The only difference is the amount of data being input into MAC calculation and verification.

3.1.7 Key Management

IPsec uses the Internet Key Exchange (IKE) [10] protocol for key establishment. Main Mode and Quick Mode are the two phases used to set up an authenticated key. In Main Mode the two communication parties authenticate each other and establish a secure channel which is used by the Quick Mode protocol to exchange the session keys and to establish the Security Associations (SA). There are four different authentication methods specified: digital signatures, two forms of authentication with public key encryption, and pre-shared keys. The Quick Mode is used to establish SAs for other security services, such as AH and ESP. SAs are unidirectional. To secure one RTP-media channel three different SAs are needed on every endpoint (one for the unidirectional RTP-stream and two for the bi-directional RTCP-stream). If anti-replay is enabled, the transmitted sequence number must be unique. After transmission of the 2^{32}th packet, the sender's and receiver's counter must be reset by establishing a new SA. The notion of rekeying an SA in IPsec actually implies the new SA.

In case of H.235, the master chooses a random session key. The transport of session keys from the master to the slave in a secure manner can be done by using one of following possibilities:

1. If the H.245 signaling channel is already confidential, no additional encryption of the session key is necessary. The connection procedures indicate a secure mode of operation, the negotiated handshake and authentication shall occur for the H.245 logical channel before any other H.245 messages are exchanged. After completing the securing of the H.245 channel, the terminals use the H.245 protocol in the same manner that they would in an insecure mode.
2. If a secret key and algorithm has been established outside the H.245 channel (through an outband channel), the shared secret is used to encrypt the session key material. The enciphered key is included into the H.245 messages.
3. The session key is encrypted by using the public key of the slave. Certificates may be used when the H.245 channel is not secure, but may also be used in addition to the secure H.245 channel.

In H.235 the key refresh rate shall be such that no more than 2^{32} packets are encrypted using the same key. Implementations should refresh keys after encrypting 2^{30} packets using the same key. Both parties are free to change the media session key as often as considered necessary due to their security policy. The master distributes e.g. new session keys or the slave requests new session keys from the master.

SRTP does not define any key establishment protocol. It just describes how to derive the necessary session keys for encryption and authentication from the master keys. SRTP uses a key derivation mechanism that minimizes the signaling traffic between the communication parties. In fact there is no data exchange or signaling necessary to establish new session keys. The session keys are generated involving a key derivation function by every endpoint on its own. The function gets four input values: the master key, the master salt key, the SRTP packet index and a label. The master key and the master salt key are provided by an external key management system. They are used to secure both protocols SRTP and SRTCP. The label denotes which kind of session key should be derived. The SRTP packet index is generated out of the first RTP/RTCP packet the key will be applied to. The key derivation rate defines the rate in which new keys will be generated. It is a 16 bit value. If this value is set to zero only the mandatory initial session key derivation will be executed (at the beginning of the session). The generated session keys will then be used without any rekeying in the future, except the absolute allowed maximum usage for SRTP-keys in general. This maximum value is determined by the maximum value of the SRTP-index (for RTP 2^{48}-1 and for RTCP 2^{31}-1). If the value is greater than zero, the keys will be refreshed after the defined number of packets.

3.2 Data Expansion

VoIP systems generally transport one or two codec frames per packet. In case of the G.723.1 codec, one frame consists of 20 or 24 octets and contains 30 msec of speech. Figure 1 shows a typical voice packet consisting of one frame of 20 octets.

IPsec uses ESP for encrypting payload data. The ESP format consists of SPI, sequence number, payload data, padding, pad length, next header, and optional authentication data.

In H.235 the PKCS #7 [25] padding is used to obtain a multiple of cipher blocks. If the ciphertext stealing [26] in CBC mode is employed instead, then the encrypted portion has the exact size of the plaintext. In addition the media anti-spamming mechanism introduces a data expansion due to the appending of a MAC.

SRTP adds two new fields to the packet format as defined in RFC 1889. The SRTP Master Key Identifier (MKI) and the authentication tag are both optional fields with variable length. The MKI identifies the master key used to encrypt and/or authenticate the packet. The authentication tag is only present when the RTP packet is integrity protected. The same applies to SRTCP packets. Beside the MKI and the authentication tag SRTCP contains two more additional fields: a 1 bit E-flag indicating whether the payload is encrypted or not and the SRTCP index, which is a 31 bit counter for the SRTCP packets.

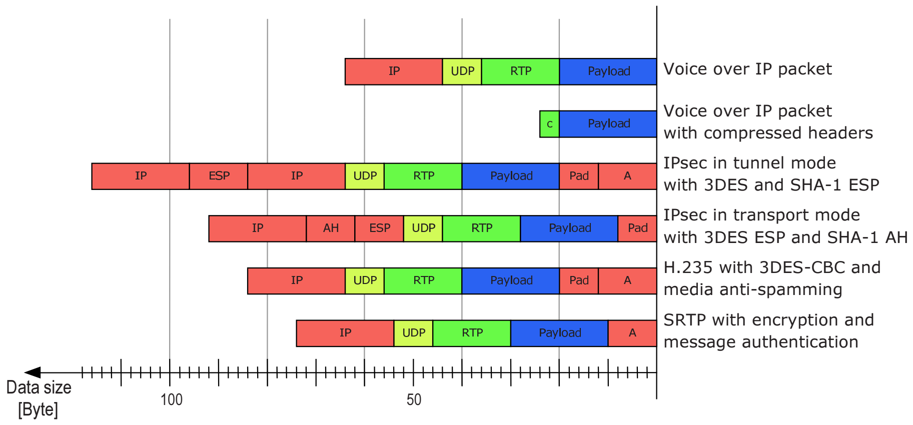


Fig. 1. Data Expansion added by Security Protocols

Both fields are required. Through the use of the keystream generator no padding is required and therefore the data size of the encrypted payload must not be padded. While the whole RTP packet is being authenticated, only the payload of the packet will be encrypted. This ensures the usage of header compression described in [27]. Both H.235 and SRTP enable the use of IP/UDP/RTP header compression.

3.3 Error Propagation

In case of CBC a transmission error affects two plaintext blocks. Suggest that one RTP packet has two frames (24 octets per frame) and the encryption algorithm has a blocksize of 16 octets. If one bit error occurs in the first cipher block on transmission, the receiver looses the first frame and 8 octets of the second frame after decryption.

In case of SRTP, the process of encrypting a packet by XORing with the key-stream causes no error propagation occurs.

In all cases error propagation is limited to the corrupted packet.

3.4 Computational Delay

IPsec introduces additional delay due to en-/decryption and MAC generation/verification.

Sender-side:
$IPsec computation delay = Enc(UDP header||RTP header||RTP payload) +
GenMAC(ESP header||UDP header||RTP header||RTP payload)
Receiver-side:
IPsec computation delay = Dec(UDP header||RTP header||RTP payload) +
VerMAC(ESP header||UDP header||RTP header||RTP payload)

H.235 encrypts and decrypts the payload of each RTP packet. It furthermore calculates and verifies a MAC on a small portion of the RTP header.

Sender-side:
H.235 computation delay = Enc(RTP payload) + GenMAC(RTP header)
Receiver-side:
H.235 computation delay = Dec(RTP payload) + VerMAC(RTP header)

The process of encrypting/decrypting a packet in SRTP consists of XORing that keystream. In addition a MAC is calculated over the RTP header and the (encrypted) RTP payload.

Sender-side:
SRTP computation delay = Enc(RTP payload) + GenMAC(RTP header || RTP payload)
Receiver-side:
SRTP computation delay = Dec(RTP payload) + VerMAC(RTP header || RTP payload)

3.5 Supported Signaling Standards

Since IPsec resides on the network layer, it is transparent to applications.

H.235 is one standard incorporated by the umbrella standard H.323 and is therefore used within H.323-based applications.

SRTP is intended to be used within SIP or RTSP-based applications.

3.6 Multicast Support

IPsec supports multicast communication but IKE provides only the establishment of SAs between two hosts the key management.

H.235 doesn't consider multicast environments. Multipoint conferences are realized using a so called Multipoint Controller (MC) or Multipoint Controller Unit (MCU). Privacy for individual sources within a common session (assuming Multicast) may be achieved with individual or common keys. These two modes may be arbitrarily chosen by the MC(U) and shall not be controllable from any particular endpoint except in modes allowed by MC(U) policy. In other words, a common key may be used across multiple logical channels as opened from different sources.

SRTP considers multicast environments. A suitable key management protocol for this purpose has to be selected.

3.7 Packetfilter Support

Since multimedia communication is secured end-to-end, the usage of IPsec prevents packet filtering. Packet-filter on the path from the sender to the receiver is not able to decrypt the IPsec secured payload to analyze addresses and ports.

H.235 and SRTP do not affect packet filtering since the encryption takes place at OSI layer 7.

3.8 Summary

The following table summarizes the comparison of IPsec, H.235, and SRTP:

Table 1. Summary of Properties

	IPsec	H.235	SRTP
Key management	✓	✓	✕
User Authentication	✓	✓	✕
Integrity of RTP payload	✓	✕	✓
RTCP Protection	✓	✕	✓
Pre-computation	✕	✕	✓
Error Propagation	✓	✓	✕
Data Size Expansion	✓ (High)	✓ (Medium)	✓ (Low)

4 Implementation

To examine the results we implemented H.235 and SRTP based on the openH323 project (http://www.openh323.org/). The openH323 project aim is to create a full featured, interoperable, open source implementation of the ITU-T H.323 teleconferencing protocol.

4.1 H.235 and SRTP

Currently openH323 supports H.235 for securing RAS messages but doesn't support security functions for H.225.0, H.245, and RTP. We extended the H.225.0 and H.245 signaling implementations and added the missing security fields and structures such as `CryptoToken`, `ClearToken`, and `H.235Key`. Voice encryption takes place between the codec and the RTP packet construction. All encryption algorithms as stated in H.235 Annex D were integrated.

The SRTP framework is considered as a *bump in the stack* implementation, which resides between the RTP application and the transport layer. We integrated our SRTP framework into the openH323 project and extended the OpenPhone application - a graphical H.323 client for Microsoft Windows operating systems - to support SRTP sessions. With this application it is possible to set up voice and video communications between two endpoints.

4.2 Delay Time

Upon our implementations we made measures to get a meaning about how many delay is introduced by H.235 and SRTP due to the cryptographic transforms and - if necessary - how the QoS-Parameters have to be adapted. Since both investigated protocols make only use of symmetric cryptography, the additional delay is kept reasonable. Table 2 shows some results collected on an Intel Pentium II system with 350 MHz clock, 128 MB RAM and the MS Windows 2000 operating system.

SRTP adds a total of 0.6 ms to the end-to-end delay when applied to a VoIP - application using the GSM codec (1 Frame per RTP packet) and the default algorithms (AES/SIC, HMAC/SHA-1).

Table 2. Measurement Results of delay time

Cryptographic Transform	GSM-1 (33 octets)	GSM-4 (132 octets)	G.711-1 (240 octets)	G.711-5 (1200 octets)
AES/SIC	Enc: 0.08 ms Dec: 0.08 ms	Enc: 0.12 ms Dec: 0.12 ms	Enc: 0.16 ms Dec: 0.16 ms	Enc: 0.52 ms Dec: 0.52 ms
AES/f8	Enc: 0.11 ms Dec: 0.11 ms	Enc: 0.16 ms Dec: 0.16 ms	Enc: 0.18 ms Dec: 0.18 ms	Enc: 0.52 ms Dec: 0.52 ms
3DES/CBC	Enc: 0.48 ms Dec: 0.48 ms	Enc: 0.80 ms Dec: 0.80 ms	Enc: 0.94 ms Dec: 0.94 ms	Enc: 2.5 ms Dec: 2.5 ms
HMAC/MD5	Gen: 0.09 ms Ver: 0.09 ms	Gen: 0.11 ms Ver: 0.12 ms	Gen: 0.15 ms Ver: 0.15 ms	Gen: 0.27 ms Ver: 0.28 ms
HMAC/SHA-1	Gen: 0.21 ms Ver: 0.23 ms	Gen: 0.27 ms Ver: 0.29 ms	Gen: 0.30 ms Ver: 0.31 ms	Gen: 0.55 ms Ver: 0.55 ms

4.3 Communication Overhead and Error Propagation Comparison

We show communication overhead of multimedia packet for a normal IP packet and the three security protocols-IPsec, H.235, and SRTP using a simulation. Figure 2 shows the percent of communication overhead when packet size is from 40 bytes to

500 bytes. H.235 and SRTP have almost same extension if they use the same MAC algorithm. However, since H.235 uses the block encryption algorithm and requires padding, it sometimes has more communication overhead. IPsec has the most communication overhead among security protocols, because it always requires ESP header for the confidentiality service.

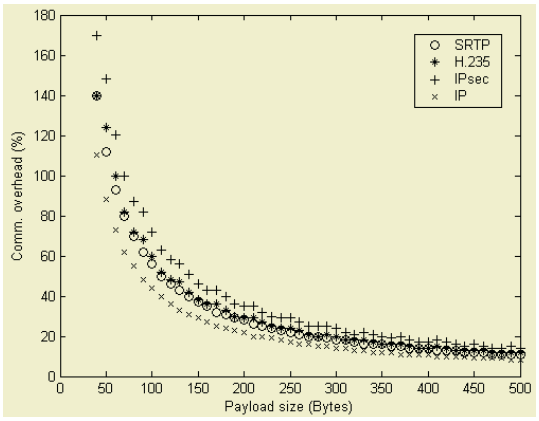


Fig. 2. Communication overheads versus payload size

We also show error propagation for the three security protocols-IPsec, H.235, and SRTP using a simulation. We performed simulation with 1,000,000 sample packet and use a 40 bytes payload consisting of 2 frame of G.723.1. The block size of encryption algorithm is 8 bytes. The error position of packet is random. H.235 and IPsec usually use CBC mode, which cause error propagation in the next block. If two error blocks include parts of two codec frames, the two codec is corrupt. Figure 3 shows the change in the number of corrupt frames for the three security protocols

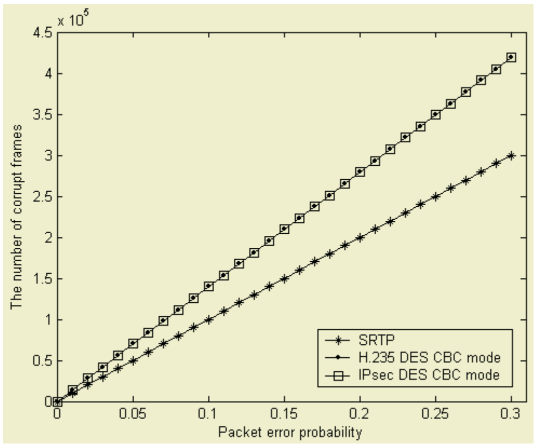


Fig. 3. Corrupt frames versus Packet error probability

versus the packet error probability. The asterisk markers represent the number of corrupt frames of SRTP, which increases in proportion to a rise in packet error probability. The square markers represent corrupt frames of IPsec and the point markers represent corrupt frames of H.235. In these cases, the number of corrupt frames increases faster than SRTP because of the error propagation.

5 Conclusions

We analyzed existing security protocols for their usability in Internet multimedia communications and investigated their impact on quality of service. All three investigated protocols provide security services for Internet multimedia communication. In case of H.235 the offered security for the media stream is incomplete. RTP has no mechanisms to detect manipulations in the payload and in some fields of the header. Furthermore the protection of RTCP is left out completely. That makes H.235 very vulnerable to a variety of attacks. Even worse is that the voice encryption profile is optional and must not be part of a H.235 implementation. IPsec instead protects both protocols but causes some overhead that influences QoS parameters. Especially the introduced data expansion results in an increasing demand of transmission rates. The end-to-end delay is also affected the most by IPsec since the amount of data which has to be encrypted is multiple higher then the RTP payload size. Moreover the end-to-end application of IPsec causes problems with intermediate packet-filtering. Finally IKE is a very complex protocol and often the reason for interoperability problems between different IPsec implementations. Nearly every multimedia application relies on RTP to packetize the data outputted by the various codecs before sending them. Therefore by defining security services for RTP every real-time-oriented communication application can benefit from them. This is the goal of the SRTP. SRTP defines optimized security functions as integrity of RTP payload, RTCP protection, pre-computation, and low communication overhead for real-time multimedia application using RTP. Besides the concepts and mechanisms incorporated in SRTP aim to a heterogeneous environment of IP-based communication devices. This scope contains mobile devices, as well as modern PCs. Mobile devices constitute special requirements to the used algorithms and implementations. They don't have so much computational power and less memory than PCs. Therefore SRTP uses modern and efficient cryptographic algorithms for confidentiality and integrity. The specified modes of operation enable to previously produce keystream segments using the packet index of upcoming packets.

References

1. ITU-T Recommendation H.323 Version 4: Packet Based Multimedia Communication Systems (2000)
2. ITU-T Recommendation H.225.0 Version 4: Call Signaling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems (2000)

3. ITU-T Recommendation H.245 Version 7: Control Protocol for Multimedia Communication (2000)
4. M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg: SIP: Session Initiation Protocol, IETF RFC 3261 (2002)
5. H. Schulzrinne, A. Rao and R. Lanphier: Real Time Streaming Protocol (RTSP), IETF RFC 2326 (1998)
6. S. Kent, R. Atkinson: Security Architecture for the Internet Protocol, IETF RFC 2401 (1998)
7. T. Dierks, C. Allen: The TLS Protocol Version 1.0, IETF RFC 2246 (1999)
8. ITU-T Recommendation H.235 Version 2: Security and encryption for H-Series (H.323 and other H.245-based) mulitmedia terminals (2000)
9. M. Baugher, D. McGrew, D. Oran, R. Blom, E. Carrara, M. Naslund, and K. Norrman: The Secure Real-time Transport Protocol, IETF RFC 3711 (2004)
10. D. Harkins, D. Carrel: The Internet Key Exchange (IKE), IETF RFC 2409 (1998)
11. H. Schulzrinne, S. Casner, R. Frederik, and V. Jacobson: RTP: A Transport Protocol for Real-Time Applications, IETF RFC 1889 (1996)
12. H. Schulzrinne: RTP Profile for Audio and Video Conferences with Minimal Control. IETF RFC 1890 (1996)
13. L. Lo Iacono, C. Ruland: Confidential Multimedia Communication in IP Networks, Proceedings of 8th IEEE International Conference on Communication Systems, Singapur (2002)
14. NIST: Advanced Encryption Standard (AES), FIPS PUB 197 (2001)
15. ISO/IEC 10116: Information technology - Security techniques - Modes of operation for an n-bit blockcipher, International Organization for Standardization (1997)
16. D. McGrew: Segmented Integer Counter Mode: Specification and Rationale, Cisco Systems, Inc. (2000)
17. W. Diffie, M. Hellman: Privacy and Authentication: An Introduction to Cryptography. In Proceedings of the IEEE, 67(3) (1979) 397-427
18. Technical Specification Group Services and System Aspects: Specification of the 3GPP Confidentiality and Integrity Algorithms, 3rd Generation Partnership Project (3GPP), Technical Specification, Document 1: f8 and f9 Specification (2001)
19. H. Krawczyk, M. Bellare, R. Canetti: HMAC: Keyed-Hashing for Message Authentication, IETF RFC 2104 (1997)
20. R. Gennaro, P. Rohatgi: How to Sign Digital Streams, Advances in Cryptology – CRYPTO ‘97 (1997) 180-197
21. A. Perrig, R. Canetti, B. Briscoe, J. Tygar, D. X. Song: TESLA: Multicast Source Authentication Transform, IETF Internet Draft (Work in progress) (2000)
22. A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song: Efficient Authentication and Signing of Multicast Streams over Lossy Channels, IEEE Symposium on Security and Privacy (2000) 56-73
23. C. Ruland, N. Schweitzer, L. Lo Iacono: Signing Digital Streams, Proceedings of the 4th International ITG Conference on Source and Channel Coding, VDE-Verlag Berlin (2002)
24. J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman: MIKEY: Multimedia Internet KEYing, IETF RFC 3830 (2004)
25. RSA Security: PKCS #7: Cryptographic Message Syntax Standard, Version 1.5, Revised November 1 (1993)
26. J. Daeman: Cipher and Hash Function Design, Ph.D. Thesis, Katholieke Universiteit Leuven (1995)
27. T. Koren, S. Casner, J. Geevarghese, B. Thonpson, and P. Ruddy: Enhanced Compressed RTP (CRTP) for links with high delay, packet loss and reordering, IETF RFC 3545 (2003)

An Improvement on Privacy and Authentication in GSM*

Young Jae Choi and Soon Ja Kim

School of Electrical Engineering and Computer Science, Kyungpook National University,
1370 Sangyuk-dong Buk-gu Daegu, Republic of Korea
cyj75@hotmail.com, snjkim@ee.knu.ac.kr

Abstract. There are a lot of subscribers of GSM across the world. GSM is one of the major achievements in modern cellular telephony. The authentication protocol for GSM has some drawbacks [1,2]. We show why drawbacks occur and propose an improvement on privacy and authentication in GSM not only to improve drawbacks but also to achieve the goals as follows: mutual authentication, reduction of bandwidth consumption between VLR and HLR, reduction of storage space of VLR database, reduction of authentication data flows, security and efficiency. In particular, we show a possible attack to threat MS's location privacy. The merit of the proposed protocol is that it is based on the existing security algorithm A3, A5 and A8.

1 Introduction

The Global System for Mobile communication (GSM) is a common standard issued by European Telecommunication Standards Institute (ETSI) and is the first digital mobile network architecture put into practice. GSM is widespread across the world and has always been the standard of the Pan-European digital cellular system [9, 11]. GSM is undoubtedly a major achievement in modern cellular telephony.

GSM is so convenient in that anyone can use it to communicate with anyone else in almost any place at any time. Because of the convenience of GSM, there are a lot of subscribers across the world [10].

The tremendous market growth of GSM system indicates the growing importance of mobile communication and an eminent need of security in mobile telephones during international communication. There are two major worries about security issues [12,13] ; The confidentiality of radio transmission, e.g., the privacy, and the authentication of the user. Confidentiality refers to the guarantee that the communication messages are not intercepted by eavesdroppers. On the other hand, authentication is carried out to ensure that any unauthorized user cannot fraudulently obtain his/her required services from the home domains. In some novel applications in modern wireless communication, these two issues are still the major concerns

In the data communication security framework, complete security relies on an implementation of a standard method over the complete path, including wireless and wired paths[1]. The radio path is by nature more susceptible to be eavesdropped and to be fraudulent in use than wired path.

* Part of this research was supported by the Korea Science and Engineering Foundation, R.O.K., under contract no. R05-2003-000-12083-0.

GSM considers that the network, except the interface of the MS and the VLR, is secure and all VLR/HLR are trustworthy. These are acceptable assumptions in mobile telephone systems, which are homogeneous, small and only provide simple services. However, these assumptions cannot be guaranteed in a large scale and heterogeneous communication system. Since GSM does not adopt ciphering mechanism between VLR and VLR/HLR, an eavesdropper can monitor the physical channel that connects to the HLR and eavesdrops MS's location updating information and Security Related Information. So the data confidentiality and location privacy in wired communication also should be provided [8].

The security functions of the GSM aim at two goals. One is to protect the network against unauthorized access, and the other is to protect the privacy of the user. Thus, the security features provided by GSM consist of three aspects as follows [1,9]:

- subscriber identity authentication
- subscriber identity confidentiality
- The confidentiality of user data and signaling information on radio path.

A mobile user must prove one's identity to access the network. Authentication protects against fraudulent uses and ensures correct billings. The subscriber identity confidentiality deals with the location privacy of mobile users. The confidentiality of user data and signaling information depends upon many aspects of the system. Among them, the user's subscription data, service profile and information were sent over open radio paths as well as the security parameters. It is associated with the confidentiality purpose [1,10].

The GSM has been improving in these security issues ever since. The protection mechanisms for mobile communication have been examined by many researchers [1,2,12]. Most of them offer the authentication mechanisms of the data confidentiality for the wireless transmission, not for the wired transmission. In order to offer the security for the wired path, some of them used the public key cryptography and signature scheme to improve the security. Some protocols applied for the symmetric cryptography scheme e.g. DES[6,7], to achieve a more secure authentication and to increase the privacy of mobile subscribers in wireless path. [3]. To equip the system with the high security functions, a symmetric cryptosystem or an asymmetric cryptosystem can be employed. However, in the view of mobile phone power and computational ability, the GSM system is still popular and widespread throughout the world because of its simplicity and efficiency [1].

The approach of Lee et al's modified protocols [1,2] proposed methods to reduce the amount of information and to eliminate the stored sensitive information in the VLR. They also proposed the security protection for the wireless and the wired path. We modify their protocols and propose an improved protocol for GSM to provide more efficient and secure method in authentication and location privacy.

The contents of this paper are divided into four parts. To begin with, we investigate the basic architecture of GSM. Secondly, the authentication protocol of GSM, where inefficient security mechanisms is discussed and we propose the enhanced authentication protocol. Thirdly, we show the location privacy protocol of GSM and its shortcomings and propose a new location privacy protocol. Finally, the features of the proposed protocol are presented by the cryptanalysis, the comparison on the basis of design goals and capacity analysis, among the original GSM system and other proposed approaches.

2 GSM System

2.1 Notation

The acronyms in table 1 are used.

Table 1. Acronyms

Acronyms	Meaning
MS	Mobile Station
HLR	Home Location Register
VLR o/n	Visitor Location Register old/new
MSC	Mobile services Switching Center
SIM	Subscriber Identity Module
ME	Mobile Equipment
IMSI	International Mobile Subscriber Identity
IMEI	International Mobile Equipment Identity
TMSI o/n	Temporary Mobile Subscriber Identity old/new
LAI	Location Area Identity
A3	Authentication Algorithm
A5	Signaling data and user data encryption algorithm
A8	Ciphering key generating algorithm
Kc	Ciphering session key
RAND	Random Number
SRES	Signed Response

2.2 GSM Architecture

The GSM system has two major components: fixed installed infrastructure and the MS, which use the services of the network and communicate over the radio path. The fixed installed GSM network can again be subdivided into three subnetworks: the radio network, the network switching network and the public network. These subnetworks are called subsystems in the GSM standard [10,15].

In the GSM architecture, as shown in Fig. 1, the MS communicates through radio path with Base Station Subsystems (BSS) which are connected to MSC. The MSC performs all the switching functions of a fixed-network switching node, e.g. routing path search, signal routing, and service feature processing. The Authentication Center (AuC) stores subscribers' secret keys and generates security parameters for the authentication. AuC would normally be attached to a HLR but located in a secure environment [1]. The HLR stores all permanent subscriber data and the relevant temporary data of all subscribers. Besides the fixed entries like service subscriptions, permissions and the stored data also contain a path to the current location of the mobile station. The HLR is needed as the central register for routing to the subscribers, for which it has administrative responsibility. An HLR record consists of three types of information: (a) mobile station information such as IMSI and the mobile station ISDN number (MSISDN), (b) location information such as the ISDN number (address) of a VLR, and (c) service information such as service subscription, service restriction, and supplementary services.

The VLR stores the data of all mobile stations which are currently staying in the administrative area of the associated MSC. A VLR can be responsible for the areas of

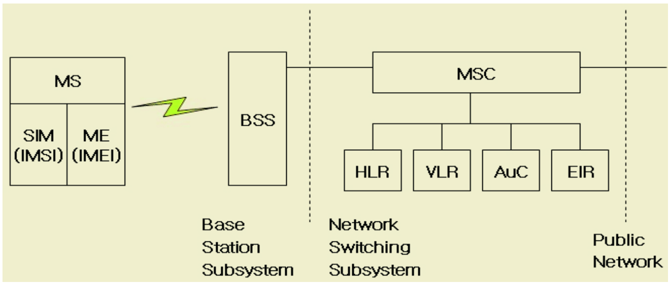


Fig. 1. The GSM architecture[1,15]

one or more MSCs. Mobile stations roam freely. Therefore, depending on their current location, they are registered in one of the VLRs of their home network or in a VLR of a foreign network. For this purpose, a mobile station has to start a registration procedure when it enters a Location Area (LA). The responsible MSC passes the identity of the MS and its current LAI to the VLR, which includes these values into its database and registers the MS. If the MS has not been registered with this VLR, the HLR is informed about the current location of the MS. The VLR information consists of three parts: (a) mobile station information, (b) location information, and (c) service information.

The Equipment Identity Register (EIR) is used to prevent the use of stolen or fraudulent MS equipments.

The MS is a piece of equipment which is used by mobile service subscriber for access to services. It consists of two major components: ME and SIM. Only the SIM of a subscriber turns a piece of mobile equipment into a complete mobile station with network usage privileges, which can be used to make calls or receive calls. In addition to the IMEI, the mobile station has subscriber identification and call number such as IMSI and Mobile Station ISDN number (MSISDN) as subscriber-dependent data. Thus, GSM mobile stations are personalized with the SIM card. All the cryptographic algorithms to be kept confidential are realized on the SIM, which implements important functions for the authentication and user data encryption based on the subscriber identity IMSI and secret key Ki [10].

3 GSM Authentication Protocol

3.1 Verification of Subscriber Identity

When a subscriber is added to a home network for the first time, a secret key (Ki) is assigned in addition to the IMSI to enable the verification of the subscriber identity. All security functions are based on the secrecy of this key. At the network side, the key Ki is stored in the AuC of the HLR. At the subscriber side, it is stored on the SIM card of the subscriber.

The process of authenticating a subscriber (Fig.2.) is essentially based on the A3 algorithm, which is performed at the network side as well as at the subscriber side. Network sends RAND to the MS. The MS calculates a SRES through A3 with Ki and RAND as inputs and transmits its SRES value to the network which compares it with its calculated value. If both values agree, the authentication is successful. Each execution of the algorithm A3 is performed with a new value of the random number RAND which cannot be predetermined; in this way recording the channel transmission and playing it back cannot be used to fake an identity [10].

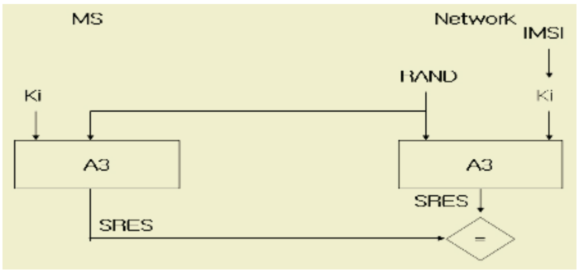
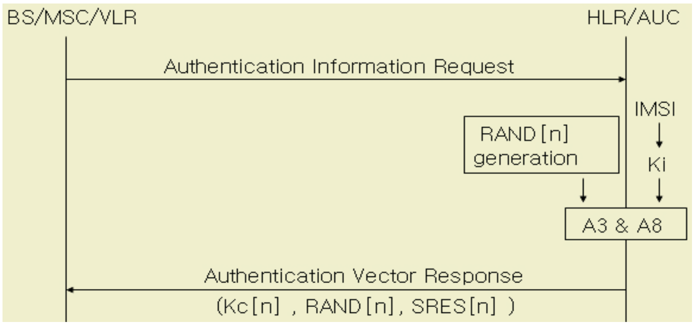


Fig. 2. Principle of subscriber authentication[10]

3.2 Generating Security Data

The security data of the current GSM authentication are described in the Fig. 3. At the network side, the 3-tuple (Kc, RAND, SRES) does not need to be calculated each time when authentication has to be done. Rather the AuC can calculate a set of (Kc, RAND, SRES) 3-tuples in advance, store them in the HLR, and send them on demand to the requesting VLR. The VLR stores these sets (Kc[n], RAND[n], SRES[n]) and uses a new 3-tuple from them for each authentication procedure. Each 3-tuple is used only once. When VLR consumes all sets of parameters, it requires HLR to send other sets of parameters.


BS/MSC/VLR
HLR/AUC
Authentication Information Request
IMSI
RAND [n] generation
Ki
A3 & A8
Authentication Vector Response
(Kc[n] , RAND[n], SRES[n])

SRES = A3(Ki, RAND) Kc = A8 (Ki, RAND)

Fig. 3. The authentication parameters

The random number RAND is generated and the pertinent signature SRES is calculated with the A3 algorithm, whereas the A8 algorithm generates the encryption session key Kc.

The set of security data, a 3-tuple consisting of Kc, RAND, and SRES, is sent to the HLR and stored there. In most cases, the HLR keeps a supply of security data, which can be transmitted to the local VLR, so that one does not have to wait for the AuC to generate and transmit a new key. When there is a change of LA into one belonging to a new VLR, the sets of security data can be passed on to the new VLR [9,10].

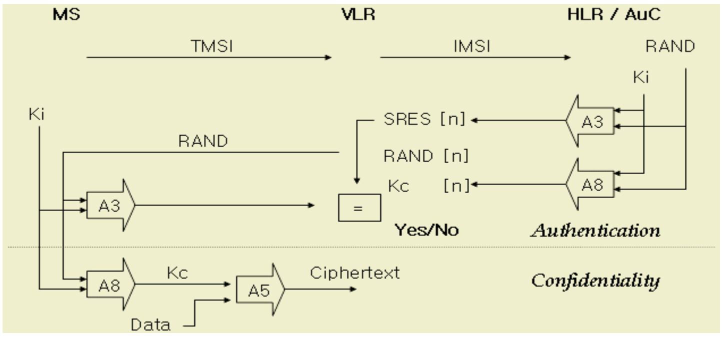
3.3 Encryption of Signaling and Payload Data

The encryption of GSM provides data confidentiality in wireless path. The encryption of data is performed at the transmitting side. On the receiving side, the decryption directly follows the demodulation of the data stream.

The encryption of signaling and user data is performed at the mobile station as well as at the base station. This is a case of symmetric encryption, e.g. ciphering and deciphering are performed with the same key Kc and the A5 algorithm. (Fig. 4.)

Based on the secret key Ki stored in the network, the session key Kc for a connection or signaling transaction can be generated at both sides, and the Base Transceiver Station (BTS) and the MS can decipher each other's data.

The authentication and data confidentiality of GSM are described in the Fig. 4. There are a few existing drawbacks of the current system [1,2]. First, the space overhead can occur when the VLR stores a set of authentication parameters. Secondly, the VLR needs the assistance of HLR when it identifies the MS. If VLR consumes all sets of authentication parameters of MS, it requests additional parameters to the HLR. Thirdly, there is bandwidth consumption between the VLR and the HLR, when the VLR needs other sets of authentication parameters. Fourthly, the authentication of VLR/HLR is not instituted in the GSM protocol. In fact, a fake VLR/HLR can give incorrect information to the user and cause a leak of confidential data in the MS. Once the sensitive information stored in the VLR is intercepted by an unauthorized user, the communication can be eavesdropped [1].



Ciphertext = A5(Kc, Message), Message = A5(Kc, Ciphertext)

Fig. 4. Subscriber identity authentication and user data confidentiality in GSM [1]

4 Enhanced Authentication Protocol

4.1 The Design Goals of Authentication Protocol

We have examined the drawbacks of GSM protocol in previous sections. In order to improve these drawbacks, we choose design goals of authentication protocol as follows:

(1) To achieve mutual authentication between an MS and the VLR
(2) To improve the location privacy
(3) To reduce the authentication flows
(4) To reduce the stored space in VLR
(5) To reduce bandwidth consumption between VLR and HLR
(6) Authentication of mobile users is to be done by the VLR instead of the HLR, even if the VLR does not know the subscriber's secret key Ki and A3 algorithm.

Our general assumption is that the HLR and the VLR shares a symmetric key. The key can be shared by several key-exchange algorithms [14].

4.2 Proposed Authentication Protocol

The improved protocol is depicted in the Fig. 5 and described in detail as follows:

(1) The MS chooses a RAND and calculates the SRES1 by using A8 algorithm, shared secret key Ki and RAND. In order to use A5 as the encryption algorithm and to hide the secret key Ki, the MS computes another cipher key Ku with its IMSI, HLR_ID and Ki. The Ku is 64 bit-length, and is computed through a one-way hash function. The MS encrypts RAND with Ku through algorithm A5 and sends it with other parameters: TMSI, LAI, SRES1. The VLR stores SRES1 to authenticate the MS for a while. The VLR checks LAI to find the VLRo and requests the MS's real identity IMSI by sending TMSI to the VLRo. When the VLRo sends IMSI to the VLRn, it encrypts subscriber information with the secret key shared with the VLRn. By using encryption scheme and a shared secret key, we can provide the protection method for location privacy in wired path.

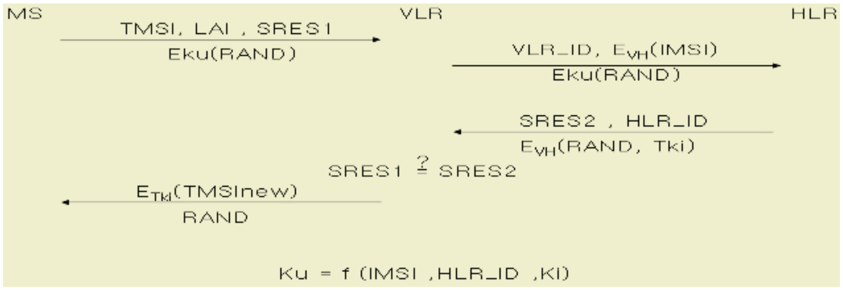


Fig. 5. Proposed authentication protocol

(2) In the similar way as in the existing authentication process for GSM, the VLR obtains the IMSI of the MS from the VLRo. Before sending IMSI to the HLR, the VLRn encrypts IMSI with the secret key K_{VH} shared with the HLR of the MS. Then it sends the IMSI along with its identification VLR_ID and the RAND encrypted by the MS. The VLR cannot know MS's RAND because the VLR does not have the secret key Ki.

(3) Once the HLR receives these messages, it checks the identity VLR_ID of the visiting VLR in the database. If the HLR finds the VLR_ID in its database, it gets the shared secret key K_{VH}. Then it decrypts the encrypted parameter by K_{VH} and finds the IMSI of the MS.

In the same way as in the existing authentication process, the HLR verifies the IMSI, and finds the assigned secret key Ki stored in the AuC. Then, the HLR decrypts the MS's parameter with the key Ki and obtains the RAND. Once the HLR obtains the RAND, it computes the Temporary Secret Key (Tki) through A8 algorithm and by using RAND and Ki as inputs. Finally, SRES2 is calculated through A3 algorithm and the two important factors: RAND, Ki.

Over the wired path, the HLR sends SRES2, HLR_ID and parameters (Tki and RAND) encrypted with the shared key K_{VH} to the visiting VLR of the MS.

(4) When the VLR receives these authentication messages, it checks the identity HLR_ID of the HLR in its database and finds the shared key. The VLR decrypts Tki and RAND with the key K_{VH}. Then, it compares SRES1 with SRES2. If these values agree, the identity of MS is authenticated. Now, Tki is used to compute the session key. The VLR creates the new TMSI of the visiting MS and encrypts it with the Tki through the A5 algorithm. Subsequently, it sends RAND and encrypted TMSI to the MS.

(5) The MS compares the RAND with its own RAND. If they are the same ones, the MS can know that the VLR is the valid one. If the VLR is not authenticated by the HLR, it cannot decrypt RAND. So, the authentication between the MS and the VLR is completed. The MS computes the Tki through the algorithm A8, RAND and secret key Ki. In fact, the MS can compute the Tki in advance, because MS chooses the RAND and has the key Ki. This procedure can save computation time. Then the MS decrypts the new TMSI and acquires it.

The security of the new authentication protocol for GSM is based on algorithm A3, A5, A8.

5 GSM Location Privacy Protocol

The MS roams from one place to another and has access to the network in any place at any time. The location of particular mobile user is the valuable information which needs protection [1,3]. The intent of location privacy function is to prevent disclosing which subscriber is using which resources in the network, by listening to the signaling traffic on the radio path. On one hand this should ensure the confidentiality of user data and signaling traffic, on the other hand it should also prevent localizing and tracking of a mobile station. This means that the IMSI should not be transmitted as plaintext. Instead of the IMSI, one uses a TMSI on the radio path for identification of subscribers. The TMSI is temporary and has only local validity, which means that a subscriber can only be uniquely identified by TMSI and the LAI. The association between IMSI and TMSI is stored in the VLR. The TMSI is issued by the VLR, at the latest, when the MS changes its location from one LA into another (location updating). When a new location area is entered, this is noticed by the mobile station which reports to the VLRn with the old LAI and TMSI (LAIold and TMSIold, Fig. 6.). The VLRn inquires the VLRo about MS's IMSI and all pertinent security information by passing the TMSIo to the VLRo.

The VLR issues a new TMSI for the MS. This TMSI is transmitted in encryption form. Then, the VLRn reports the IMSI of the MS to the corresponding HLR. The HLR stores the current location information of this MS in its database. Finally, the HLR clears all information relevant to the MS in the VLRo [10].

In the case of database failures, if the VLR database is partially lost or no correct subscriber data is available (loss of TMSI, TMSI unknown at VLR, etc), the GSM standard provides for a positive acknowledgement of the subscriber identity.

For the subscriber identification, the IMSI must be transmitted as plaintext before encryption is turned on. There is a possible attack on location privacy because of this procedure.

The design of the location privacy protocol in GSM relies upon the security of the wired connection that is traversed by VLR and HLR communication. The original GSM protocol has two problems [1]. First, when the VLR updates the location of an MS, IMSI is exposed and delivered throughout the network without any protection. Secondly, when a user roams to another VLR, the location is updated by sending IMSI to the VLRn while the VLRo is not accessible and no correct subscriber data is available. It is possible that an unauthenticated third party may eavesdrop on the IMSI and identify this mobile user.

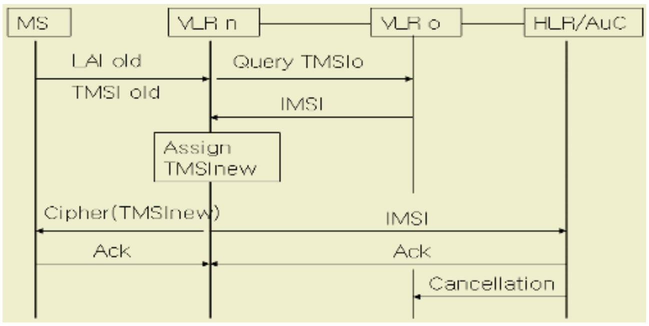


Fig. 6. Location update using TMSI [9,10]

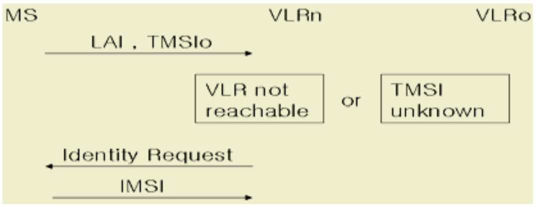


Fig. 7. Authentication at location updating in a new VLR, abnormal cases [9]

6 Location Privacy Protocol

6.1 Location Privacy Problems in the Authentication at Location Updating

When the MS updates its location, it sends TMSIo and LAI to the VLRn. In procedure, there are two abnormal cases(Fig.7). One is that the VLRo cannot verify the MS's IMSI. When the new VLR sends TMSIo to the VLRo, data loss occurs in the VLRo. If the VLRo cannot find the IMSI in its database, it sends the 'unknown' message to the VLRn. Then the VLRn sends 'Identity Request' message to the MS and the MS sends IMSI to the VLRn without any protection.

The other is that the VLRn cannot reach the VLRo. This process is similar to the above process. When the VLRn receives LAI and TMSI, it cannot reach the VLRo. In this case, the VLRn sends ‘identity Request’ message to the MS and the MS sends its IMSI without any protection.

In these cases, there are location privacy problems in the authentication at location updating. If a malicious attacker tries to obtain the MS's IMSI, it can do it by the procedure of the Fig. 8. When the MS sends LAI and TMSI to the VLR, an attacker pretends a VLR and sends ‘VLR not reachable’ message or ‘TMSI unknown’ message to the MS. Then the MS sends its IMSI to the attacker, and the attacker obtains the MS's IMSI. It is a possible location privacy attack, because there is no way to authenticate the VLR at the MS side.

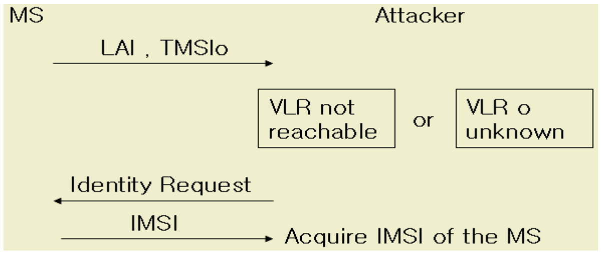


Fig. 8. Location privacy attack

6.2 Proposed Location Privacy Protocol

In order to prevent this location privacy attack, we propose a location privacy protocol. The protocol is depicted in the Fig. 9. This protocol is based on the Alias (AL). The alias is a unique identity of MS for traveling and assigned to the MS's IMSI one-by-one. The alias is similar with the structure of IMSI and has 15 bit-length. It is assigned to a user by the HLR. The assignment between aliases and real user identities should be kept secret by the HLR.

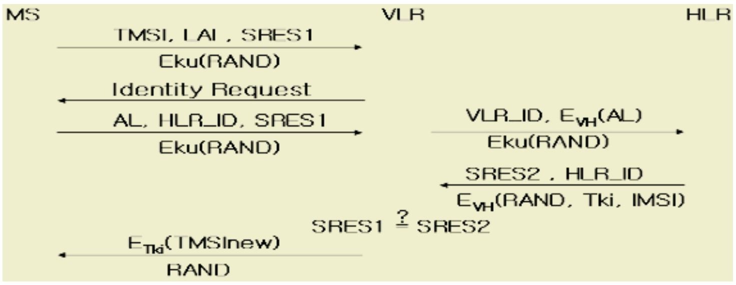


Fig. 9. Proposed location privacy protocol

When the VLR sends 'Identity request' message to the MS, the MS sends the AL and HLR_ID to the VLR. Then, the VLR encrypts the AL with the shared key and sends it to the HLR. The HLR decrypts the AL, and checks it in its database. If the HLR finds the IMSI which is assigned to the AL, it computes the Ku with the MS's information and decrypts the RAND. Then, the HLR sends encrypted the IMSI to the VLR with other security parameters. Finally, the VLR obtains the IMSI. In this way, this protocol can prevent attack mentioned above.

7 Cryptanalysis

Our authentication architecture is based on algorithms A3, A5, and A8 in order to apply for the existing Authentication protocol for GSM. In addition, we use another symmetric key: Ku. In original algorithm, the Ki is a 128 bit-length key, and the encryption Kc is a 64 bit-length. We use 64 bit-length key Ku to encrypt the MS's important authentication parameter RAND and to protect the Ki from guessing attacks.

Our protocol assumes symmetric keys to authenticate the old VLR, the new VLR and the HLR. These keys are the basis of location privacy and the mutual authentication between the MS and the VLR. When the old VLR sends the IMSI to the new VLR, it encrypts the IMSI with the shared symmetric key to protect against any attacks e.g. eavesdropping. The new VLR also sends the IMSI to the HLR with the same procedure. These procedures provide the location privacy in the wired transmission.

We use RAND as a important parameter. The MS generates the 128 bit-length RAND. During the authentication, nobody can know the RAND except the HLR because the secret key Ki is only known to the MS and the HLR. When the HLR decrypts the RAND and sends it to the VLR, the VLR can know the RAND. Then, the VLR sends it to the MS and the MS compares it with its own RAND. If these values agree, the MS can know that the VLR is valid because the only valid VLR has the shared key and decrypt RAND. The parameter RAND provides the mutual authentication between the MS and the VLR.

As the above, without the knowledge of Ki, no one can compute Ku, RAND, SRES and Tki. Therefore, the security of the proposed protocol is based on Ki.

8 Discussions

In the previous sections, we have reviewed the existing authentication protocols for GSM and shown their drawbacks. We have also described an improvement in the authentication protocol and location privacy for GSM. In the following, we shall demonstrate that our proposed protocol can achieve our requirements.

- **Mutual authentication:** It is assumed that the symmetric keys are set up before mobile communication and each VLR and HLR knows the shared key. If the VLR is a fake, it can neither obtain the IMSI from the old VLR nor encrypt the IMSI. By these symmetric keys, this protocol can achieve the mutual authentication between the HLR and the VLR. In addition, it can also achieve the mutual authentication between the MS and the VLR by comparing the RAND and SRES.
- **Improvement of the location privacy:** In order to provide the location privacy in wired transmission, we assume the shared symmetric keys among the network systems. e.g. VLR, VLR/HLR. In the abnormal case of transmission, we use Alias to protect the IMSI in wired transmission. It works in wireless transmission, too.
- **Reduction in authentication procedures:** The original GSM and Lee et al's protocol [1,2] are needed 5 data flows to finish the authentication procedure. In our protocol, the VLR authenticates the MS just in 3 data flows and the MS authenticates the VLR in 4 data flows. The fourth flow includes the new TMSI of MS. Therefore, it reduces the procedure of sending new TMSI.

- **Reduction in the storage of the VLR database:** In the proposed protocol, it has shown that the VLR stores only one copy of authentication parameter (Tki) instead of n copies (RAND[n], SRES[n], Kc[n]) in original protocol and (RAND, Tki) in Lee et al's protocol. Therefore, the proposed protocol can save the VLR database space.
- **Reduction of bandwidth consumption:** In the proposed protocol, the HLR gives the VLR Tki to authenticate the MS. As long as the MS stays in the coverage area of the visiting VLR, the VLR can use the Tki to authenticate the MS for each call. Though the visiting VLR consumes all sets of authentication parameters, it does not need to request the HLR to send other sets of parameters. Therefore, the signaling load is reduced between the VLR and the HLR and the bandwidth consumption is reduced too.
- In the existing authentication protocol for GSM, the VLR authenticates the MS with the assistance of the HLR when it used up all of the authentication parameters. In the proposed protocol, the VLR authenticates the MS with Tki assigned by the HLR. Once the VLR has the Tki, it can authenticate MS without assistance of the HLR.
- The security of the new protocol is based on algorithm A3, A5 and A8 of the original GSM.

We have mentioned about the original GSM architecture, its authentication scheme, drawbacks and the goals of secure authentication in detail. Many authentication protocols [1]-[6] cannot achieve our goals as shown in Table2. Our protocol meets the requirements without changing the architecture of the GSM system. Lee et al proposed protocols that did not change the original architecture [1,2]. But it cannot provide the location privacy in wired path.

Next, we have depicted our authentication protocol. The original GSM protocol and the Lee et al's protocol[1] do not support mutual authentication between the MS and the VLR. In our protocol, the MS authenticates the VLR with RAND generated by the MS. In fact, it can be a burden for the MS to generate RAND because of its weak computation power and small battery. But the MS does not need to generate RAND for every location update. Only when the MS moves to a new VLR, it gener-

Table 2. Comparison among the GSM authentication protocols

	GSM	Ours	[1]	[2]	[3]	[4]	[5]	[6]
Mutual Authentication between MS and VLR	N	Y	N	Y	Y	Y	N	N
Mutual Authentication between VLR and VLR/HLR	N	Y	Y	N	Y	Y	N	Y
Reduction of Bandwidth Consumption	N	Y	Y	Y	N	N	N	N
Reduction of the Storage of VLR database	N	Y	Y	Y	N	N	N	N
Authentication of MS by VLR instead of HLR	N	Y	Y	Y	N	N	N	Y
The Security Basis is same as the original architecture	-	Y	Y	Y	N	N	Y	N
The number of Data Flows to authenticate MS	5	3	5	5	3	3	5	6
The number of Data Flows in protocol	5	4	5	5	4	4	5	8
Identity Confidentiality in wired path	N	Y	N	N	Y	N	N	Y

[·]: Reference No.

ates RAND. In addition, the MS chooses RAND and generates SRES and Tki in advance. These procedures save the setup time. Since the VLR does not ask the HLR for another sets of authentication parameters in Lee et al's protocols [1,2] and our protocol, the bandwidth consumption is less than that of the original GSM protocol. In addition, In the Lee et al's protocols, a 2-tuple (RAND, Tki) stores in VLR for independent authentication of MS but our protocol is needed to stored only one parameter Tki. Therefore our protocol is needed less storage in VLR.

We have shown a possible attack in location privacy in wired transmission and proposed a new location privacy protocol that is secure in wireless and wired path. The alias assigned by the HLR can prevent the possible attack.

9 Conclusion

In this paper, we have examined some drawbacks of the existing authentication protocol and a possible attack on location privacy. In order to overcome these disadvantages, we propose an improvement on authentication and location privacy protocol. The proposed authentication protocol provides mutual authentication between VLR and MS, reduction of authentication data flow and storage space in VLR. The new location protocol can prevent a location privacy attack with the Alias which we proposed.

References

1. C.H. Lee, M.S. Hwang, and W.P. Yang: Enhanced privacy and authentication for global system for mobile communications, Wireless networks 5 (5.1999) pp. 231-243
2. C.C. Lee, M.S. Hwang, and W.P. Yang: Extension of authentication protocol for GSM, IEE Proc.-Commun. Vol. 150. No. 2 (4. 2003)
3. Refik Molva, Didier Samfat, and Gene Tsudik: Authentication of Mobile Users, IEEE Network (March/April 1994) pp. 26-34
4. Nawal El-Fishway, Mostafa Nofal, and Albert Tadros: An effective approach for authentication of Mobile users, IEEE VTC2002 (2002) pp 598- 601
5. M.S. Hwang, Y.L. Tang, C.C. Lee: An efficient authentication protocol for GSM Networks, IEEE (2000) pp 326-329
6. Nawal El-Fishway, Mostafa Nofal, and Albert Tadros: An Improvement on secure communication in PCS, IEEE (2003) pp175- 182
7. Min Xu and S. Upadhyaya: Secure Communication in PCS, IEEE VTC01 (2001) pp.2193-2197
8. S.P.Shieh, C.T.Lin, J.T.Hsueh: Secure communication in Global Systems for Mobile Telecommunications, Proc. 1st Workshop on Mobile Computing, ROC. (1995) pp136-142
9. ETSI TS 100 929 v8.1.0 (7.2001)
10. Jorg Eberspacher, Hans-Jorg Vogel and Christian Bettstetter: GSM, switching, Services and Protocols (2nd edition), WILEY(2001)
11. Asha Mehrotra: GSM System Engineering, Artech House Publishers (1997)
12. D.Broron: Techniques for privacy and authentication in personal communication systems, IEEE Personal communications (8.1995) 6-10
13. J.E. Willas: Privacy and authentication needs of PCS, IEEE personal communications (8.1995) 11-15
14. Bruce Schneier: Applied Cryptography(2nd edition), John Wiley & Sons Inc. (1996)
15. Khalid Al-Tawil, Ali Akrami: A new Authentication protocol For Roaming Users in GSM Networks, IEEE(1999)

Encrypted Watermarks and Linux Laptop Security

Markku-Juhani O. Saarinen

Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O. Box 5400, FIN-02015 HUT, Finland
mjos@tcs.hut.fi

Abstract. The most common way to implement full-disk encryption (as opposed to encrypted file systems) in the GNU/Linux operating system is using the encrypted loop device, known as CryptoLoop. We demonstrate clear weaknesses in the current CBC-based implementation of CryptoLoop, perhaps the most surprising being a very simple attack which allows specially watermarked files to be identified on an encrypted hard disk without knowledge of the secret encryption key.
We take a look into the practical problems of securely booting, authenticating, and keying full-disk encryption. We propose simple improvements to the current CryptoLoop implementation based on the notions of tweakable encryption algorithms and enciphering modes. We also discuss sector-level authentication codes. The new methods have been implemented as a set of patches to the Linux Kernel series 2.6 and the relevant system tools.

1 Introduction

Perhaps the most typical approach for protecting the confidentiality and authenticity of files is to use PGP or other similar encryption tools which allow users to encrypt files for transmission and storage. Explicit decryption is required before a file can be modified. After plaintext file is no longer needed, it must be re-encrypted and securely deleted (wiped). This can be cumbersome, so more transparent methods have been devised. These can be categorized into encrypting file systems and sector-level encryption.

Encrypting File Systems. These generally allow flexible control over which directories are encrypted and which are not. The main problem with encrypting file systems is that temporary files containing sensitive information are often stored on unencrypted portions of the disk, such as swap devices / page files, various caches, or "temp" directories. Examples of encrypting file systems are CFS [5] and TCFS [7] in the UNIX world, and Microsoft's EFS.

Sector-Level Encryption. These systems implement encryption below the file system level, and allow entire hard disks to be encrypted. Sector-level encryption systems do not usually allow fine-grained access control for files. The whole volume is protected with a single master key (many options for managing and storing this key exist). File system ("upper layer" from our viewpoint) accesses data as *sectors*. In our terminology a sector is a logical unit consisting of 512 bytes (4096 bits); larger physical sectors must be split into 512-byte pieces. We use this convention regardless of the actual sector size used by the file system (typically 1024, 2048 or 4096 bytes in the case of EXT2 [6]).

Since the file system must be able to perform quick single sector random access reads and writes on the disk, each sector must be "independent" of others; no other data is needed for encryption and decryption than the sector itself, sector identifier, and the secret key.

Sector-level encryption is often the preferred choice in cases where the hard disk has little physical protection (e.g. with personal portable computers). The Linux Crypto-Loop implements sector-level encryption, as does SFS [12] and many commercial systems for the Microsoft OS platforms.

Linux CryptoLoop. CryptoLoop is a facility provided by the Linux Kernel to easily integrate encryption below the file system level. It works regardless of the file system used. With CryptoLoop, a physical disk drive, a disk drive partition, or a container file is "looped" as a loop device (`/dev/loop0`, `/dev/loop1`, ...). After a key is provided to the Kernel using the `losetup` utility, the loop device driver transparently takes care of encryption and decryption whenever the loop device is accessed. The loop device can then be initialized and mounted using any file system. Figure 1 illustrates the call dependencies of CryptoLoop.

Since encryption and decryption is always done on independent 512-byte sectors, we call such a transformation *Sector Enciphering Operation* (SEO). SEO and its inverse can be characterized as follows:

$$C = \text{SEO}(P, K, T)$$
$$P = \text{SEO}^{-1}(C, K, T)$$

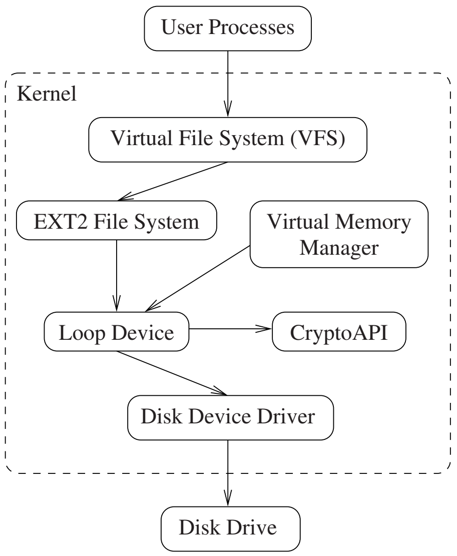


Fig. 1. CryptoLoop call structure.

Here P is the plaintext sector of 512 bytes (4096 bits), K is the secret key, T is the tweak (sector number), and C is the corresponding ciphertext sector.

Efficiency considerations usually dictate that SEO is length-preserving. Information-theoretic arguments can be used to show the impossibility of proper message authentication in such a case. We will return to these issues in section 5.

Structure of This Paper. In section 2 we illustrate some attacks against the present Linux implementation of CryptoLoop. We believe the "watermarking" technique in section 3 to be novel. In section 4 we introduce a practical security model for sector level-encryption (this is fundamentally the same as that proposed by Halevi and Rogaway in [14, 13] and implicitly by others before them [8]), and discuss various ways to achieve the goal set by the model. Section 5 discusses sector-level message authentication and makes an argument for not having sector-level authentication but rather authenticating at file system level. Section 6 contains some of our experiences and practical thoughts about implementation of sector-level encryption. Section 7 contains performance analysis of the implementation, and is followed by concluding remarks in section 8.

2 Attacks

Currently the Linux 2.6 series offers a selection of SEOs constructed from various well-known block cipher algorithms. Mode of operation can be chosen to be either ECB or CBC. We ignore the obviously weak ECB mode and concentrate on CBC, which is absolutely the most commonly used choice.

CBC is currently initialized for each sector by using the sector number directly as the initialization vector. This convention is also used at least by 2.2, 2.4 series of Linux kernels, and by versions of Jari Ruusu's loop-AES package that were published prior to December 2003 [23].

The 512-byte plaintext sector P is split into blocks $P = p_1 \mid p_2 \mid p_3 \cdots$. The corresponding ciphertext is $C = c_1 \mid c_2 \mid c_3 \cdots$. Encryption of plaintext block x with a secret key K is denoted by $E_k(x)$. Hence the SEO becomes:

$$\begin{aligned}
c_1 &= E_k(p_1 \oplus T)\\
c_2 &= E_k(p_2 \oplus c_1)\\
c_3 &= E_k(p_3 \oplus c_2)\\
&\cdots\\
c_{32} &= E_k(p_{32} \oplus c_{31})
\end{aligned}$$

Here we assume that a 128-bit block cipher is used. In the case of 64-bit block ciphers, the last plaintext and ciphertext blocks would naturally be p_{64} and c_{64}, correspondingly. Here T is the Tweak and is equal to the logical sector number.

Standard Attacks. In some conditions it is conceivable that an encrypted disk will be subject to repeated scans and even active manipulation based on such scans (typical scenario for such scans is during international travel – customs and baggage checks).

Replay attacks on CBC mode in CryptoLoop have been known for years (many were explicitly pointed out by J. Etienne [10]), yet it has persisted as the dominant disk-encryption system for Linux. We give some well-known examples:

1. **Corrupt.** Corruption of chosen data blocks is difficult to detect. As CBC decryption has little error propagation, modifying a ciphertext block within sector will only corrupt the corresponding plaintext block (8 or 16 bytes) and cause chosen bit changes the one block immediately following it.
2. **Move.** It is easy to shift multiple ciphertext blocks anywhere within the hard disk. Only the first plaintext block will be corrupted. This opens a big toolbox of "cut & paste" attacks.
3. **Revert.** It is possible to revert chosen sectors to their previous values without detection. An attacker can from two ciphertext images detect where the changes lie and choose the sectors to be reverted accordingly.

Together these options (possibly with the aid of watermarkings – see next section – to provide "location data") allow subtle and powerful manipulation of the hard disk on ciphertext level.

3 Encrypted Watermarks

We introduce the concept of *encrypted watermarks*, which are subtle markings on arbitrary plaintext files. The existence of these markings (and hence the existence of files that contain them) can be proven on the corresponding ciphertext without knowledge of the secret key[1]. Encrypted watermarks provide a powerful investigative technique for a forensic analyst who wishes to identify files such as restricted documents, pornography or "warez" on encrypted hard drives.

We recall a couple of basic facts that apply to most UNIX disk file systems, including EXT2, EXT3, ReiserFS, UFS, XFS, etc.

a) File data is mostly ($P > 0.99$) stored on consecutive sectors on the disk[2].
b) File data starts at an even multiple of the sector length.

Simple Watermarks. In CBC Linux CryptoLoop, we may create simplest kinds of watermarks in files by making the first block of two consecutive sectors differ only in the least significant bit. There is a significant probability that the corresponding ciphertext blocks will be equal, and hence can be identified with high certainty from the ciphertext alone (without knowledge of the secret key).

Advanced Watermarks. This idea can be extended in many ways, as long as the XOR-difference between the plaintext sectors' first plaintext blocks can be matched with the XOR-difference of their sector numbers. The probability is affected by the actual

[1] The concept of *encrypted watermark* is an extension of watermarking as understood in the context of Digital Rights Management.

[2] This is not necessarily true for FAT file systems, which are much more prone to fragmentation.

sector size and fragmentation properties of the file system above the loop device, but experiments have shown the probability to be very high with typical EXT2-based setup.

It is possible to devise more reliable and elaborate watermarking schemes by basing the markings on particular patterns of first-block collisions within a file. A file can contain multiple complementary watermarks. Such a marking (and file) can be located on an encrypted device with high certainty. Since the majority of the file is unchanged, such markings are stealthy and easy to insert into jpeg pictures, mp3 music files, ps/pdf files, program binaries etc.

Example (with a 64-bit block cipher):

```
0x00000  Any data ..
0x13000  Eight bytes 0000000000000000
0x13008  Any data ..
0x13200  Eight bytes 0000000000000001
0x13208  Any data ..
0x13400  Eight bytes 0000000000000000
0x13408  Any data ..
```

This file contains a watermark that will be visible in the encrypted device, as long as data at positions 0x13000 .. 0x135ff are contained in three consecutive sectors on the physical disk.

Proof. Let n, $n + 1$, and $n + 2$ be the sectors containing this section of the file. We note that for any n either $n \oplus (n + 1) = 1$ or $(n + 1) \oplus (n + 2) = 1$. Hence either $E_K(0 \oplus n) = E_K(1 \oplus (n+1))$ or $E_K(1 \oplus (n+1)) = E_K(0 \oplus (n+2))$. The watermark can thus be detected as the same ciphertext block can be found at the beginning of two consecutive ciphertext sectors (regardless of location).

4 A Security Model

Following the language of [13, 14, 18, 19], we want SEO to be a *strong*, *tweakable*, *pseudorandom permutation* (PRP); for a random key we wish SEO (and its inverse) to be indistinguishable from a random permutation.

Furthermore, we wish SEO to be resistant to various attacks based on key scheduling and tweaks. Equivalent keys, related keys, and other "weak key" classes (if they exist) should be computationally difficult to find. There should be no shortcut attacks based on chosen, weak, or related tweaks.

In short, there should be no analytic attack which is computationally cheaper than exhaustive search through the keyspace, regardless of the amount of chosen plaintext and/or ciphertext (with associated tweak values) available. If these conditions are violated by an attack, SEO can be considered broken.

Encryption Modes. We have found certain commercial disk encryption systems to utilize CTR and even ECB modes for encryption, both of which offer clearly unsatisfactory resistance to attacks. ECB for obvious reasons, and CTR in the case that multiple scans are made (all plaintext changes made between the scans will become visible when the two ciphertext disk images are “xorred” together).

Recently Halevi and Rogaway proposed the CMC and EME modes for this purpose [13, 14]. These remain unbroken. We observe that EME, CMC, and all other satisfactory modes for sector-level encryption are in fact *double-encryption modes*, and hence offer roughly half the speed of conventional modes such as CBC.

Special Tweakable Block Ciphers. As the use of provably secure sector encipherment modes appears to result a significant performance penalty (due to beforementioned double-encryption), crafting a custom 4096-bit block cipher seems justified.

This motivated us to design a prototype cipher named Herring [22]. Herring was designed with the explicit purpose of satisfying all the security requirements outlined above, while also maintaining encryption speed comparable to AES in “single - encryption” mode. Herring accepts a 128-bit key and a 128-bit tweak. Conclusions about the security of Herring can be drawn only after years of public analysis. We wish to make Herring public soon, although the findings of this paper are valid for any 4096-bit tweakable block cipher.

Some other 4096-bit block ciphers have been constructed explicitly for the purpose of sector level encryption. One proposal was the Mercy block cipher [8], which was broken by Fluhrer [11]. Another 4096-bit block cipher present in the literature is the unnamed proposal of Kaliski and Robshaw [17], which was found to be weak by Saarinen [21].

Wide-block block ciphers have also been constructed from other primitives. An interesting approach was taken by Anderson and Biham with the BEAR and LION ciphers, which combine a hash function with a stream cipher to produce a block cipher [2].

5 Why No Authentication at Sector Level?

Even if SEO satisfies the criteria given in the previous section, it is still prone to some replay attacks (e.g. reverting and corruption of sectors). We have considered a number of approaches for the sector-level authentication problem but we have not found satisfactory solutions. Different options include:

a) Check everything at mount time using a signature algorithm. This is clearly preventive, since it would involve both reading the entire hard disk during mount time and correspondingly signing it when unmounting (assuming that it was mounted read-write).
b) Incorporating MACs with the sectors. By including the sector number in the authenticated data, this makes other modifications other than reversion attack (where sector n is reverted to its previous contents) detectable.
   If implemented on loop device level, this would make the boundaries between physical sectors and logical sectors incompatible. A read operation on a single isolated sector would always imply two physical sector reads. A write operation would imply two sector reads and two sector writes. The performance drop would therefore be significant.

c) Maintain a table of MACs in memory, which is loaded during mount time and stored (and signed) in a file when the disk is unmounted. This is a reasonably implementable approach, especially if we use the actual file system sector size and a truncated MAC. Using HMAC-SHA1-64 [4] with sector size 4096 bytes corresponds to two megabytes of MACs for each gigabyte of disk (0.195%), which is manageable.
d) Dynamically maintain a Hash Tree of the sectors of the disk. The memory requirements are similar to the previous option, but an advantage exists in that the hash of the whole disk can be maintained at all times, thus perhaps allowing better recovery from crashes. We do not see this as a very effective solution.
e) Dynamically maintain a "sum" of MACs of each sector:

$$S = \bigoplus_{i=1}^{n} \mathrm{MAC}_K(i \mid P_i)$$

Here MAC_K is a keyed MAC with secret K and P_i is the sector with logical number i. It is plain that a sector write becomes a sector read-write in this option, but that performance penalty is acceptable. Even though read, write, mount, and unmount operations are quick and memory usage is acceptable, the drawback is that in order to *detect* changes (i.e. to find out that the sum doesn't match), the whole disk needs to be read and this doesn't even tell us where the change occurred!

Of these approaches, maintaining a table of MACs in memory (option C) appears to be the most feasible one, but it alone doesn't really provide a satisfactory solution for error recovery. It is nice to know that an error has occurred but what can you do with this information? In cryptographic communication protocols the connection is typically simply shut down if "MACs don't match", but making the whole disk unusable after a single bit error or operating system crash is not acceptable.

Therefore perhaps the most important question is to decide on how to respond to an anomalous situation, i.e. when the authentication code doesn't match. Mechanisms do not exist in the current Linux architecture for the loop device to communicate to a file system layer that the error may be a security issue rather than a simple disk error (such sectors will simply be marked "bad"). And how could such a determination be reached?

If an error is found (after a crash or unclean shutdown) and the encrypted disk is `fsck`'ed, the recovered file system may be perfectly healthy and the ill effects cancelled. Therefore it is reasonable to incorporate authentication at file system level.

Practical Approach. Since CryptoLoop cannot really protect against all attacks that modify ciphertext in the disk, we recommend regular use (e.g. by a `cron` mechanism) of systems such as Tripwire (see `www.tripwire.org` or `www.tripwire.com`), which can detect malicious changes to files. It is noteworthy that these tools will not only detect physical attacks when a computer is at unauthorized hands, but also many network-based attack vectors (esp. "backdooring") while the computer is being used by its authorized user.

6 Implementation

We can see three different methods of implementation for a SEO.

a) A software implementation as a part of the operating system kernel. This is the easiest option, but will cause a performance penalty in disk-intensive applications. Such an implementation may also utilize cryptographic hardware speedups via DMA.
b) Hardware support in the disk controller or the hard drive itself. After keying the disk controller or the drive itself will transparently encrypt and decrypt everything. This would essentially make the encryption process independent of the operating system itself.
c) Implementation as a "bump in the cable" on the (IDE/ATA or SCSI) cable. This method of implementation would be easy to integrate into existing systems, but it may end up being more costly than hardware support in the controller.

We have only experimented with the first option, but plans exist for a FPGA-based hardware implementations as well.

Our Current Implementation. The required modifications to the present Linux 2.6 series kernel source were minor. The patch contains about 3000 lines of code, majority of it being the Herring cipher. The patch also adds support for a new mode of operation and "tweak" keying required for sector-level encryption. The needed changes can therefore be implemented in the framework of current Linux cryptographic support.

Perhaps surprisingly, no changes were required for `losetup` or other v.2.12 utilities that are used in setting up and keying the loop devices. A script was written which requests a passphrase from the user, hashes it (with salting), and passes it on to `losetup`. More elaborate key management and authentication methods would be easy to add. We had also to create small statically linked versions of these utilities so that they fit into the initial ram disk, discussed below.

Booting. One of the trickiest things about encrypted Linux laptops is to come up with a reasonably secure boot procedure. Since we're using standard hardware, it is almost impossible to come up with a "bullet-proof" solution, as the hardware itself can be modified to include key loggers or similar intercept devices. It is not within the scope of this paper to discuss methods for preventing hardware modification in any detail, but some level of inexpensive protection can be achieved by simple seals and manual inspection of hardware after suspected modification.

As a rule of thumb, one would like to have as little as possible of the computer hard disk to be in plaintext. Some practioners have decided to keep the "static" parts of the hard disk unencrypted (e.g. `/bin`, `/sbin`, `/usr`, but not `/etc`, `/var` or `/home`). It is relatively easy to backdoor such systems with standard rootkit tools.

Currently our best practice for booting is as follows:

1. **Firmware setup.** We are in practically forced to use standard BIOS firmware. We enable the available security features (BIOS setup passwords etc.), and disable booting from other devices than the main hard disk.

2. **Boot loader.** The boot loader resides on the Master Boot Record on the hard disk. We use the LILO boot loader in “simple mode”, so that it will directly load up the compressed kernel image and initrd (initial ram disk) using BIOS routines.
3. **Boot partition.** Since we saw little purpose in incorporating encryption into the LILO itself, we are forced to keep the kernel image and the initrd unencrypted in a separate partition. These require a total of 2.5 MB of space.
4. **Initrd.** The initial ram disk contains a small file system that is first mounted as root. A passphrase prompt becomes visible within 5 seconds after power-on. Other authentication mechanisms can also be easily incorporated into the initrd environment (which roughly corresponds to single-user mode). A master key is derived from the passphrase and used to set up the loop device. The encrypted loop device is then mounted as the new root.
   At this stage a “rescue system” can also be activated, which contains the following items (all in ram disk):
   a) **Diskwipe.** A program for quickly wiping the contents of entire hard disk. The wipe procedure exceeds the requirements set in DoD 5220.22-M standard (“sanitize” method D for non-removable rigid disks, p. 8-3-5 [9]).
   b) **Convert.** A tool for re-encrypting the hard disk using an alternative master key or cipher algorithm.
   c) **Busybox.** A small shell-like environment for rescue purposes[3].

   There is no access control to reach the rescue system.
5. **Boot verification.** In the first stages of (encrypted) boot, digital signatures of the kernel image and the initrd image are verified at the boot partition. The corresponding signatures and public keys reside on the encrypted partition of the disk are thus difficult to forge (in fact it would be sufficient to check their message digests against known values).
   Also an attempt is made to verify the integrity of firmware by comparing it to known digest values with physical memory image accessible through `/dev/ram`. However, not all of the firmware binary appears to be visible at this point.

We acknowledge that this boot procedure is not wholly secure against all software-based attacks, although such attacks would seem to require non-trivial human effort. One possible attack would involve crafting modifications to the kernel so that it is able to maintain the appearance that everything is going smoothly (by returning false values to system calls) while also containing a trojan horse for capturing and transmitting the master encryption key. A reasonable amount of obscurity and variation in the implementation details guards against such attacks (obscurity is unfortunately the only method available to resist against this class of attacks)[4].

Alternative Boot and Keying Methods. We have also experimented with other boot sequence options. Small, lightweight, and inexpensive USB solid state memory devices have become available in recent years. Many BIOS Firmware vendors allow booting

[3] Busybox was not developed as a part of this research effort. See `www.busybox.net`

[4] Trusted Computing Group is an emerging industry standard that will provide mechanisms to detect BIOS and MBR manipulation (among other things) and therefore to improve the security of early steps of booting. See `www.trustedcomputinggroup.org`

from these devices. It is relatively straight-forward to include the boot loader, kernel image, and initial ram disk into such a device, and thus allow 100 % of the hard disk to be encrypted, the unencrypted portion being part of one's keychain! Similar approach can be taken with CD-ROMs, albeit they are not as easily transportable. USB tokens and various smartcard systems are easily incorporated into the initial ram disk phase of booting. There is no need to store the key on the token itself, so even national ID cards which support public key decryption may be used (e.g. FINEID in case of Finland). For yet another approach see AEGIS [3] and its smartcard-based extension sAEGIS [15].

Encrypting Swap. Linux has full support for encrypted swap devices. Even easier solution is to use a swap file which resides on the same encrypted partition with the main file system[5].

Other Implementations. To our knowledge, tweakable modes have not been previously implemented in the Linux kernel.

The BestCrypt for Linux 1.5.1 [16] package from Jetico Inc. uses IV with CBC mode in a similar (although not entirely compatible) fashion as current CryptoLoop, and hence is vulnerable to the attacks described in this paper[6].

TCFS [7] appears to either use ECB mode (before version 3) or CBC mode with zero IV (after version 3), and is vulnerable to similar attacks.

Matt Blaze's Cryptographic File System [5] utilizes a combination of OFB and ECB modes, but is also vulnerable to attack. Peter Guttman's Secure File System (SFS) used MDC/SHS, a special construction which turns the SHA hash function into a block cipher in CFB mode. This does not satisfy our security requirements either. The construction based on MD5 (MDC/MD5) was shown to be weak by Saarinen [12, 21].

We have also evaluated several commercial disk encryption systems for which source code or full specification is not publicly available. Generally speaking, most commercial sector encryption products do not appear to offer effective protection against watermarking / multiple scanning attacks[7].

7 Performance

We measured the speed of read and write operations on a typical modern PC laptop, Acer TravelMate 420, which has a 2 gHz Pentium 4 CPU and ATA disks.

The performance was measured with one gigabyte (2^{30} bytes) continuous reads and writes in single-user mode using `dd` (i.e. data transfers from the loop device to `/dev/null` and from `/dev/zero` to the loop device). No special optimization was used, and caches were disabled. The operating system was our custom modified Linux 2.6.1.

[5] It appears that at the time of writing all mainline Linux versions may have deadlock problems with encrypted swap. We hope that this will be fixed soon.

[6] Since the Linux version is compatible with the Windows versions of BestCrypt from the same vendor, we believe these to be also vulnerable.

[7] Since some of our security analysis methods may be interpreted as reverse engineering or disclosure of trade secrets and thus violation of certain local laws, we are restricted in discussing these results.

The following table summarizes our measurements.

Encryption algorithm	Read MB/s	Write MB /s
None	14.8	14.7
AES-128 CBC	13.0	14.6
Herring	11.0	10.2
AES-128 EME	7.3	9.7

Implementation of AES and CBC were the the standard ones in Linux 2.6 kernel. The AES implementation is based on the work by Brian Gladman and is reasonably optimized. The implementation of EME mode [14] was by the author, with sector size 512 bytes (hence 65 AES block operations per sector, compared to 32 required by CBC). The implementation of the preliminary version of Herring was also by the author. All the implementations are in portable C language without assembly optimizations. There is room for performance improvement.

It should also be noted that laptop hard drives are somewhat slower than those drives now common on desktop machines. However, we feel confident in concluding that full sector-level encryption does not present a significant performance bottleneck for day-to-day computer use.

8 Concluding Remarks

Most government agencies and many large corporations have security policies in place which make encryption of hard disks mandatory for laptops. Easiest and most transparent method of achieving such protection is by using sector-level encryption, which leaves as little as possible of the disk unencrypted.

Disk encryption is not a performance bottleneck nor does it significantly decrease the usability of the system. Therefore there are few excuses for not deploying it where ever possible.

Careful analysis has shown that many products and techniques widely used for sector-level encryption are vulnerable to active manipulation and even watermarking; the presence of (planted) restricted data can be detected without breaking the encryption key.

However, good solutions can be reached with limited resources and open software. Sector-level encryption also offers a good motivation for research into very wide-block tweakable block cipher designs and tweakable enciphering modes.

Acknowledgements

The author would like to thank the anonymous program committee members, Jari Ruusu, and various active members of the Linux Kernel Mailing List for valuable comments. This work was supported by Finnish Defence Forces Research Institute of Technology.

References

1. *Specification for the Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication 197, 2001.
2. R. Anderson and E. Biham. *Two Practical and Provably Secure Block Ciphers: BEAR and LION*. Proc. Fast Software Encryption '96, LNCS 1039, Springer-Verlag, 1996. pp. 113–120.
3. W. A. Arbaugh, D. J. Farber, and J. M. Smith. *A Secure and Reliable Bootstrap Architecture* Proc. 1997 IEEE Symposium on Security and Privacy, ACM Press, 1997. pp. 65–72.
4. M. Bellare, R. Canetti, and H. Krawczyk. *Keying Hash Functions for Message Authentication*. Proc. Crypto 1996, LNCS 1109, Springer-Verlag, 1996.
5. M. Blaze. *A Cryptographic File System for Unix*. Proc. First ACM Conference on Computer and Communications Security, Fairfax, VA, 1993.
6. R. Card, T. Ts'o, Stephen Tweedie. *Design and implementation of the Second Extended Filesystem*. In Frank B. Brokken et al, editor, Proc. of the First Dutch International Symposium on Linux, Amsterdam, December 1994.
7. G. Cattaneo, L. Catuogno, A. Del Sorbo, and P. Persiano. *The Design and Implementation of a Transparent Cryptographic File System for UNIX*. USENIX Annual Technical Conference '01, Freenix Track. 2001.
8. P. Crowley. *Mercy: A Fast Large Block Cipher for Disk Sector Encryption*. Proc. Fast Software Encryption 2000, LNCS 1978, Springer-Verlag, 2000, pp. 49–63.
9. Department of Defense. *Industrial Security Manual for Safeguarding Classified Information,* Department of Defense Manual, DoD 5220.22-M, June 1987.
10. J. Etienne, *Vulnerability in encrypted loop device for Linux*, Manuscript available from `http://www.off.net/~jme/`, 2002.
11. S. R. Fluhrer. *Cryptanalysis of the Mercy Block Cipher*. Proc. Fast Software Encryption 2001, LNCS 2355, Springer-Verlag, 2002, pp. 28–36.
12. P. C. Gutmann. *SFS Version Documentation.* `http://www.cs.auckland.ac.nz/~pgut001/sfs/`
13. S. Halevi and P. Rogaway. *A Tweakable Enciphering Mode.* Proc. Crypto '03, LNCS 2729, Springer-Verlag, 2003.
14. S. Halevi and P. Rogaway. *A Parallelizable Enciphering Mode.* To appear in Proc. RSA Conference 2004 – Cryptographer's Track. Springer-Verlag, 2004.
15. N. Itoi, W. A. Arbaugh, S: J. Pollack, and D. M. Reeves. *Personal Secure Booting.* Proc. ACISP 2001, LNCS 2119, Springer-Verlag, 2001, pp. 130–144.
16. Jetico Inc. *BestCrypt for Linux v.1.5.1 with Linux 2.6 support*. available from `http://www.jetico.com`, 2004.
17. B. S. Kaliski and M. J. B. Robshaw. *Fast Block Cipher Proposal*. Proc. Fast Software Encryption 1993, LNCS 0809. Springer-Verlag, 1994, pp. 33 – 40.
18. M. Liskov, R. L. Rivest, and D. Wagner. *Tweakable Block Ciphers*. Proc. CRYPTO 2002, LNCS 2442, Springer-Verlag, 2002, pp. 31–46.
19. M. Luby and C. Rackoff. *How to construct Pseudorandom Permutations from Pseudorandom Functions*. SIAM J. of Computation, 17(2), April 1988.
20. R. L. Rivest. *All-Or-Nothing Encryption and The Package Transform*. Proc. Fast Software Encryption '97, LNCS 1267, Springer-Verlag, 1997, pp. 210–218.
21. M.-J. O. Saarinen. *Cryptanalysis of block ciphers based on SHA-1 and MD5*. Proc. Fast Software Encryption 2003. LNCS. Springer-Verlag, 2004.
22. M.-J. O. Saarinen. *Herring: A Tweakable Block Cipher for Sector level Encryption*. Manuscript (to be submitted for publication), 2004.
23. J. Ruusu, *Loop-AES Source and Documentation*. `http://loop-aes.sourceforge.net/`

Inconsistency Detection of Authorization Policies in Distributed Component Environment

Chang-Joo Moon[1] and Hoh Peter In[2,*]

[1] Center for Information Security Technologies (CIST)
Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
mcjmhj@korea.ac.kr

[2] Department of Computer Science & Engineering
Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
hoh_in@korea.ac.kr

Abstract. In distributed component environment, a Role-Based Access Control (RBAC) server manages all authorization policies of components in the same domains whereas the components are distributed to an application server on a network with their authorization policies. It is necessary to detect inconsistency between the authorization policies of the RBAC server and the distributed components to guarantee the authorization policy integrity while system configuration and policies are changing. In this paper, an inconsistency detection method between the two authorization policies is proposed. Conditions of guaranteeing consistency are investigated to detect the inconsistencies and validated in a case study.

1 Introduction

In distributed component environment, a component is published through its remote-call interfaces to outside, and a client application can access the methods of the component through the published interfaces [8, 11]. However, it is highly vulnerable to enable the component to be accessible from anyone including other untrusted parts. It is needed to provide proper permission for access control to the component to avoid undesirable access control problems such as information leaking due to improper handling of the component and misuse of the service provided by the component [5]. The authentication and authorization are proposed to prevent the improper user access to the component. The authorization is directly related to the component access control than the authentication, since the authentication verifies a clamed identity who can access the component whereas the authorization decides the access authority of the authenticated users who access the component.

The Discretionary Access Control (DAC) and Mandatory Access Control (MAC) are emerged for military systems but not sufficient for commercial systems [3, 7]. The researchers have studied access control models for commercial systems to support trusted transactions, subject-based security policy, conflict-of-interest and least privilege. Role-Base Access Control (RBAC) was proposed by Ferraiolo and Kuhn.

* Hoh Peter In is the corresponding author. This work was supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

For most of component platforms, the RBAC model can be applied for component access control [2, 5].

In distributed component environment, the RBAC server generally manages all authorization policies of components in the same domains whereas the components are distributed to an application server on a network with their authorization policies. Therefore the RBAC server maintains authorization policy integrity of an entire component-based system using constraints [1, 4, 9]. Since the authorization policies are embedded into the components, it is possible that the authorization policy integrity of the whole system is not guaranteed if the embedded authorization policies of the components are not properly updated with ones of the RBAC server. Especially, a distributed component environment has many changes such as addition, deletion, and modification of the components in order to effectively adapt to the changes in a business process. Because of these changes, it is necessary to change authorization requirements, and then addition, deletion, and modification of the roles. The permissions should be issued in the RBAC server. Therefore, in order to guarantee the integrity of component authorization policies accommodating such these changes, it is indispensable to guarantee the consistency of two authorization policies. In this case, it is a challenging task that proper access permissions should be managed in a fast-changing business environment [3].

This paper proposes an inconsistency detection method of the authorization policies between the RBAC server and the distributed components. The conditions of guaranteeing consistency are investigated and are used to detect the inconsistencies. The inconsistency patterns between two authorization policies are presented and validated in a case study.

The rest of this paper consists as follows. Section 2 provides the background knowledge and terms to understand this paper. In section 3, the conditions of guaranteeing consistency are formally specified and inconsistencies are analyzed. In Section 4, the conditions of guaranteeing consistency are validated through a case study. Finally, the conclusions and future work are presented in Section 5.

2 Background

2.1 RBAC Model

The concept of RBAC began with multi-user and multi-application on-line systems pioneered in the 1970s [1, 2, 4, 9, 10]. The RBAC model can effectively manage users and their permissions using the role concept. In RBAC, permissions are associated with roles, and users are made members of roles, thereby

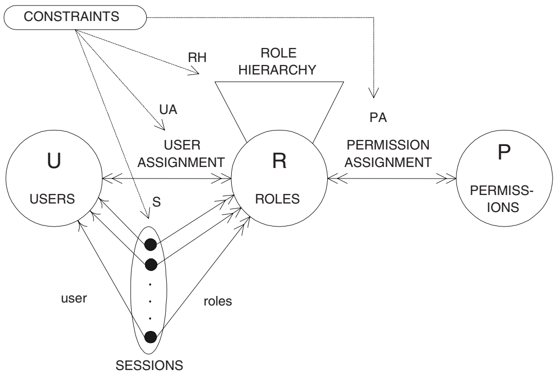


Fig. 1. RBAC model

acquiring the users' permissions. This basic concept has the advantage of simplifying the understanding and management of permissions. Fig. 1 is a conceptual diagram on the RBAC96 model [9]. The model has four elements: users, roles, permissions and sessions. A user (U) represents a human activity or an autonomous agent, while a role (R) is a job function or job title within an organization with some associated semantics regarding the authority and responsibility conferred on a member of the role [1, 2, 9]. A permission (P) is an approval of a particular mode of access to one or more objects in the system. As shown in Fig. 1, both User Assignment (UA) and Permission Assignment (PA) have many-to-many relations. A constraint specifies rules which must be observed in UA and PA. Role Hierarchy (RH) specifies a hierarchical structure of roles and is a specific form of the constraint. A senior role inherits the permissions of its junior ones through the hierarchy.

2.2 Distributed Component Environment

The general schema of distributed component environment is shown in Fig. 2. The components are distributed and deployed in many application servers, and a client program invokes remote interfaces of the components through a network. A component can call remote interfaces of other components independent on the location of the application server in which it is deployed. Each application server provides the authentication service by interacting with various kinds of authentication servers, and the authorization service by interacting with the RBAC server.

Generally, an authorization policy is composed of User Assignment Information (UAI) and Permission Assignment Information (PAI). Since the permission means the remote interface of component, the user can only call the remote interfaces assigned to their own roles. UAI and PAI are located in the RBAC server to provide necessary information to authentication servers whenever they request. The authentication servers confirm the UAI whenever a user logs in. PAI is used to describe who can be accessible to the callable remote interfaces of a component. To check out authorization efficiently, the components with their PAI are distributed in the application servers and the application servers perform the access control according to the

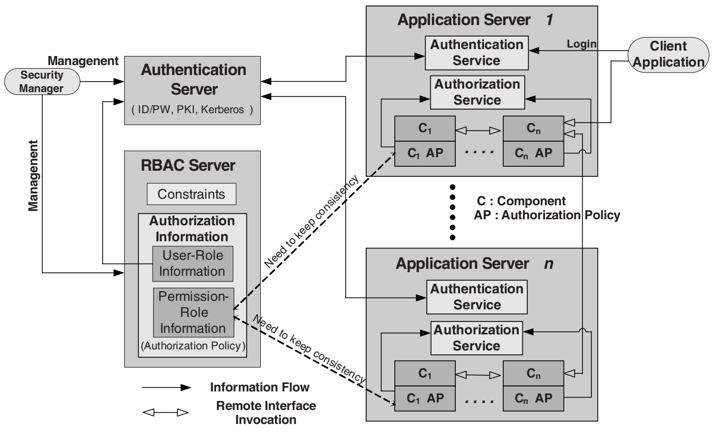


Fig. 2. Distributed component environment

PAI of the components. In this case, it is less concerned to consider the network traffic due to this distribution because the PAI should be confirmed whenever an authenticated user calls any remote interfaces.

The RBAC server manages UAI and PAI of all components, and guarantees the integrity of UAI and PAI using the UA and PA constraints [1, 6, 9]. However, it is evitable that the policy inconsistencies between the RBAC server and the application servers occur while the components and the authorization requirements are changed. It is necessary to maintain the policy consistencies. However, a few researches on policy inconsistency detection based on RBAC have been done, especially, in rapidly-changing distributed component environment. Thus, this paper investigates the conditions to detect component policy inconsistencies in the rapidly-changing environment.

3 Inconsistency of Authorization Policy

The basic elements [1, 9] and the specification of the policy inconsistencies are presented in Fig. 3. The RBAC server references the RBAC96 definition and is enhanced by new functions. The component is newly added to specify component policies.

3.1 Conditions of Guaranteeing Consistency

Definition 1 is to define the condition of guaranteeing consistency. If **Definition 1** is satisfied, consistency is guaranteed. Otherwise, inconsistency may exist. In this way, the conditions are used to detect the inconsistencies. **Theorem 1** and **Theorem 2** are defined based on Fig.3.

Definition 1 (Conditions of guaranteeing consistency)**.** In order to guarantee policy consistencies between the RBAC server and the distributed components, **Theorem 1** and **Theorem 2** should be satisfied.

Theorem 1. The union set of permissions assigned to role *cr* in the policies of all components is equal to the set of permissions assigned to the role *cr* in the policy of the RBAC server.

The formal specification for **Theorem 1** is as follows:

$$\forall cr \left(\bigcup_{k=1}^{r} \mathbf{Cperms}(cr_, c_k) = \mathbf{perms}(cr) \right)$$

Proof. The union set of permissions assigned to role *cr* in the policies of all components is all remote interfaces that role *cr* can access. Therefore the set of permissions assigned to role *cr* in the RBAC server should also be the same as these interfaces. If $\bigcup_{k=1}^{r} \mathbf{Cperms}(cr_, c_k) \subset \mathbf{perms}(cr_m)$ and **Theorem 1** is not satisfied accordingly, then the interfaces, which are accessible by the RBAC server, are prevented by a component, and hence an inconsistency takes place. If $\bigcup_{k=1}^{r} \mathbf{Cperms}(cr_, c_k) \supset \mathbf{perms}(cr_m)$ and **Theorem 1** is not satisfied accordingly, then the interfaces, which are accessible by a component, are prevented by the RBAC server and hence an inconsistency takes place.

Theorem 2. The set of roles to which permission *cp* is assigned in policy of component *c* is equal to the set of roles to which permission *cp* is assigned in policy of the RBAC server.

The formal specification for **Theorem 2** is as follows:

$\forall c \forall cp$ (**Croles**($cp_{,} c$) = **roles**(cp))

1) RBAC server

R = a set of all roles, $\{r_1, \ldots, r_n\}$

P = a set of all permissions, $\{p_1, \ldots, p_o\}$

$\mathbf{PA} \subseteq \mathbf{P} \times \mathbf{R}$, a many-to-many permission-to-role assignment relation.

$\mathbf{perms} : \mathbf{R} \rightarrow \mathbf{2^P}$,

a function mapping each role r_a to a set of permissions. ($1 \leq a \leq n$)

$\mathbf{perms}(r_a) = \{p \in \mathbf{P} \mid (p, r_a) \in \mathbf{PA}\}$,

a function returns all permissions that are assigned r_a.

$\mathbf{roles} : \mathbf{P} \rightarrow \mathbf{2^R}$,

a function mapping each permission p_b to a set of roles. ($1 \leq b \leq o$)

$\mathbf{roles}(\boldsymbol{p_b}) = \{r \in \mathbf{R} \mid (p_b, r) \in \mathbf{PA}\}$,

a function returns all roles that p_b is assigned.

2) Component

C = a set of all components, $\{c_1, \ldots, c_r\}$

has_roles(c_k) = **CR**, a set of roles belonged to authorization policy of component $c_{k,}$

$\{cr_1, \ldots, cr_s\}$, $\mathbf{CR} \subseteq \mathbf{R}$, ($1 \leq k \leq r$)

has_perms(c_k) = **CP**, a set of permissions belonged to authorization policy of component

$c_{k,}$ $\{cp_1, \ldots, cp_t\}$, $\mathbf{CP} \subseteq \mathbf{P}$

$\mathbf{CPA} \subseteq \mathbf{CR} \times \mathbf{CP}$,

a many-to-many permission(*cp*)-to-role(*cr*) assignment relation.

$\mathbf{Cperms} : \mathbf{CR} \rightarrow \mathbf{2^{CP}}$,

a function mapping each role cr_m to a set of permission *cp*. ($1 \leq m \leq s$)

Cperms(cr_m, c_k) = $\{cp \in$ **has_perms**(c_k) $\mid (cp, cr_m) \in \mathbf{CPA}\}$, a function returns all permission *cp* assigned cr_m in authorization policy of component c_k .

$\mathbf{Croles} : \mathbf{CP} \rightarrow \mathbf{2^{CR}}$,

a function mapping each permission cp_n to a set of role *cr*. ($1 \leq n \leq t$)

Croles($cp_{n,} c_k$) = $\{cr \in$ **has_roles**(c_k) $\mid (cp_n, cr) \in \mathbf{CPA}\}$, a function returns all role *cr* which has permission cp_n in authorization policy of component c_k.

Fig. 3. Basic element and function for authorization policy specification

Proof. The roles, to which permission *cp* is assigned, are all the roles accessible to permission *cp* in the policy of component *c*. Therefore, these roles should be identical to the roles to which permission *cp* is assigned in the policy of the RBAC server. If (**Croles**($cp_{,} c_k$) $\subset$ **roles**(cp)) and **Theorem 2** is not satisfied accordingly, then a role, which is accessible to *cp* by the RBAC server, is prevented by the component c_k, and hence an inconsistency take place. If (**Croles**(cp, c_k) $\supset$ **roles**(cp)) and **Theorem 2** is not satisfied accordingly, then a role, which is accessible to *cp* by the component c_k, is prevented by the RBAC server, and hence an inconsistency take place.

3.2 Classification of Inconsistency

3.2.1 Inconsistency by RBAC Server Change

If following changes are made in the policy of the RBAC server, an inconsistency exists with the policy of distributed components.

1) Inconsistency 1: IC1 (Role Addition)

If the new role r_{n+1} is added to R, r_{n+1} is assigned to the callable remote interface *p* of the component and becomes $\mathbf{perms}(r_{n+1}) = \{ p \in \mathrm{P} \mid (p, r_{n+1}) \in \mathbf{PA} \}$. That is, there exist interfaces assigned to r_{n+1}. Therefore, the policy of component c_k, which has remote interface *p*, should be allowed to access r_{n+1}. However, an inconsistency arises between two policies, since $r_{n+1} \in \mathbf{roles}(p)$ but $r_{n+1} \notin \mathbf{Croles}(p, c_k)$. In this case, the user, who assigns to r_{n+1} due to the inconsistency, can't call the remote interface assigned to r_{n+1}.

2) Inconsistency 2: IC2 (Role Deletion)

If role $r_a (1 \leq a \leq n)$ is deleted from R, assignment information for a user and the remote interface assigned to r_a is deleted along with it. Therefore, $\mathbf{perms}(r_a) = \emptyset$ and no user is assigned to r_a. However, there exists component c_k with $r_a \in \mathbf{has\_roles}(c_k)$, and $\mathbf{Cperms}(r_{a,} c_k) \neq \emptyset$, therefore an inconsistency arises. In this case, because of the inconsistency, c_k's policy still allows the user assigned role r_a to access. Although the user assigned role r_a doesn't exist, the potential problem may exist owing to the user access.

3) Inconsistency 3 (Role Modification)

Role modification means that PAI is modified. That is, a new permission is assigned to a specific role, or the existing assignment is deleted from it. The modified role name is replaced by role deletion and role addition.

- **Inconsistency 3-1: IC3-1 (Permission Assignment Addition)**
  This means that a permission, which was not assigned to a particular role, is newly assigned to the role. That is, if permission *p* is added to role r_a, $p \in \mathbf{perms}(r_a)$ and user assigned to r_a can access *p*. However, there exists $p \notin \mathbf{Cperms}(r_{a,} c_k)$ in the policy of component c_k which has *p*, thus inconsistency arises. In this case, user assigned to r_a cannot call remote interface *p* and then not receive wanted service.
- **Inconsistency 3-2: IC3-2 (Permission Assignment Deletion)**
  This means that a permission assigned to a particular role is deleted from the role. Only permission assignment is deleted from the policies. The permission itself still exists and can be assigned to other roles. If permission *p* is deleted from role r_a, $p \notin \mathbf{perms}(r_a)$ and user assigned to r_a cannot access *p*. However, if there exists $p \in \mathbf{Cperms}(r_{a,} c_k)$ in the policy of component c_k which has *p*, thus inconsistency arises. In this case, user assigned to r_a still can call remote interface *p*.

4) Permission Modification

Though security manager can manage the role and permission assignment, it cannot create, modify, or delete remote interfaces by itself. Because all permissions in the RBAC policy should belong to the component as a remote interface, the permission modification depends on the change in the component deployed in the application server. Therefore, the permission modification is not considered change of the RBAC server.

3.2.2 Inconsistency by Component Policy Changes

If the following changes are made in an application server, the inconsistency occurs with the policies of the RBAC server. Because the creation and deletion of role is the own operation of security manager in RBAC side, these are not considered during the preparation of component policy.

1) Inconsistency 4: IC4 (Component Addition)

If a new component c_k is deployed to the application server, the added c_k will have arbitrary policy. Since c_k is not registered to the RBAC server, **roles**(cp) = ∅ on cp ($cp \in$ **has_perms**(c_k)). However, **Croles**($cp_, c_k$) ≠ ∅, thus inconsistency arises. In this case, the stability of access to added component cannot be guaranteed because remote interface call which is not allowed by the RBAC server can be allowed by c_k.

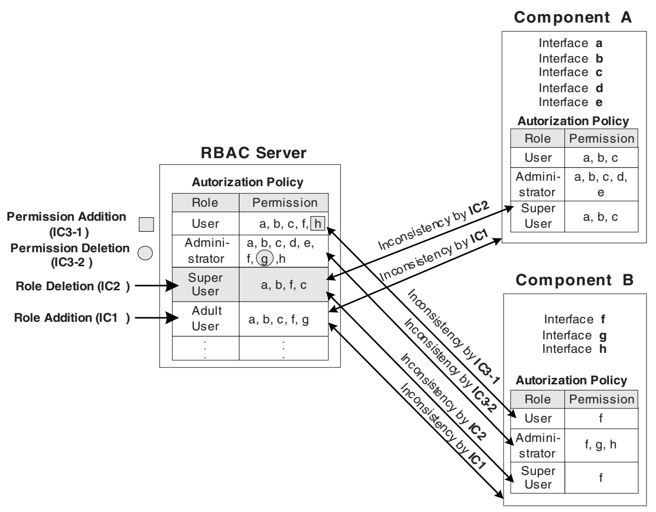
2) Inconsistency 5: IC5 (Component Deletion)

If component c_k is deleted from the application server, c_k's policy is deleted along with it, thus **Cperms**($r_{a,} c_k$) = ∅. However, if there is no synchronization, there exists permission which belongs to c_k among the elements of **perms**(r_a), thus inconsistency arises. There is no security problem due to the access to the deleted component, because the component itself does not exist.

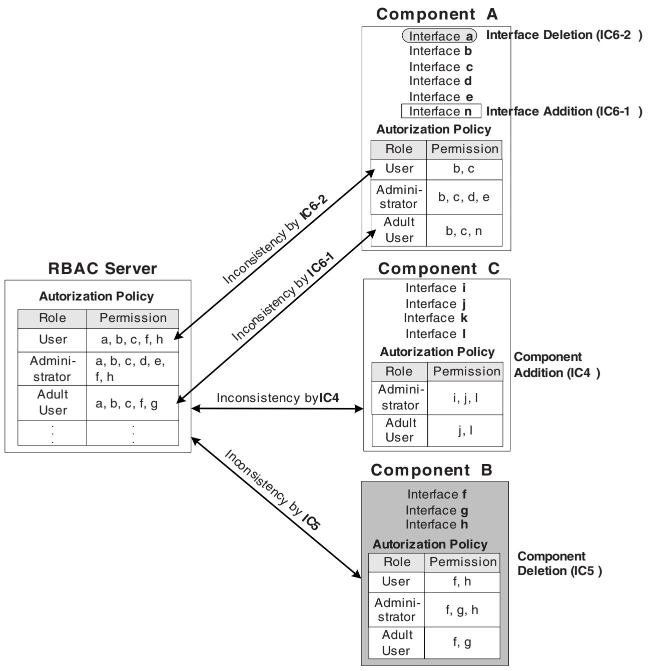
3) Inconsistency 6 (Component Modification)

The component modification means that a component policy is modified as remote interface is added to or deleted from component, or that only policy is modified without change in a component itself.

- **Inconsistency 6-1: IC6-1 (Interface Assignment Addition)**
  If interface p is newly assigned to role r_a in component c_k's policy, $p \in$ **Cperms**($r_{a,} c_k$) but $p \notin$ **perms**(r_a), thus inconsistency arises. In this case, c_k allows service that is not allowed by the RBAC server, thus the security problem arises.
- **Inconsistency 6-2: IC6-2 (Interface Assignment Deletion)**
  If interface p is deleted from role r_a in component c_k's policy, $p \notin$ **Cperms**($r_{a,} c_k$) but $p \in$ **perms**(r_a), thus inconsistency arises. In this case, c_k dose not allow permission that is allowed by the RBAC server, thus user can not receive service which is allowed for oneself. In case that inconsistency arises as c_k's remote interface p is deleted, security problem does not arise, since deleted remote interface does not exist.



(a) Inconsistency by the RBAC server change



(b) Inconsistency by component policy change

Fig. 4. Example of inconsistency

4 A Case Study

The purpose of this case study is to validate the proposed conditions of guaranteeing consistency. The proposed conditions to detect all inconsistencies (IC1 ~ IC6-2) described in section 3.2 are applied into a hypothetical component-based application. The procedure of the validation is to investigate all the inconsistencies manually (without using the proposed conditions), detect all inconsistencies (IC1 ~ IC6-2) based on the proposed conditions, and compare these two results to examine whether the proposed conditions can detect all manually-investigated inconsistencies (IC1 ~ IC6-2).

Fig. 4 (a) shows that four inconsistencies (IC1 ~ IC3-2) are occurred by the changes of the RBAC server. The Fig. 4 (b) shows that four inconsistencies (IC4 ~ IC6-2) are occurred by the changes of the component policies. Two directed arrows point to the exact part where inconsistency arises. In Fig.4 (a), the directed arrows point to added or deleted roles. Since deleted roles do not exist, they are presented in gray. The added permissions are presented in gray boxes and the deleted permissions are presented in gray circles. In Fig.4 (b), the deleted components are also presented in gray, since they do not exist. Interfaces added to component are presented in boxes, and deleted interfaces are presented in gray ellipses.

The result of Table 1 shows that **Definition 1** can detect eight inconsistencies (IC1 ~ IC6-2). That is, if **Definition 1** is satisfied, the inconsistency is not occurred. Consequently the integrity of component policies can be guaranteed.

The below shows the process of truth value creation for each row of Table 1.

Table 1. Result of inconsistencies detection

Condition / Inconsistency	Theorem 1	Theorem 2	Definition 1
Role Addition : IC1 (**Add Role** *Adult User*)	<True>	<False>	<False>
Role Deletion : IC2 (**Delete Role** *Super User*)	<False>	<False>	<False>
Permission Assignment Addition: IC3-1 (**Add Permission** *h*)	<False>	<True>	<False>
Permission Assignment Deletion : IC3-2 (**Delete Permission** *g*)	<False>	< False>	<False>
Component Addition : IC4 (**Add Component** *C*)	<False>	<False>	<False>
Component Deletion : IC5 (**Delete Component** *B*)	<False>	<True>	<False>
Interface Assignment Addition : IC6-1 (**Add Interface** *n*)	<False>	<False>	<False>
Interface Assignment Deletion : IC6-2 (**Delete Interface** *a*)	<False>	<True>	<False>

1) Inconsistency 1

If a role, *Adult User*, is added, the **Theorem 2** is as follows:

Theorem 2: roles(a) $\neq$ Croles(a, A)

roles(a) = { *User*, *Administrator*, *Super User*, *Adult User* }
Croles(a, A) = { *User*, *Administrator*, *Super User* }

Because **Theorem 2** is false, **Definition 1** can detect *inconsistency 1*(IC1)

2) Inconsistency 2

If a role, *Super Use,* is deleted, the **Theorem 1** is as follows:

Theorem 1: $\bigcup_{k=1}^{2}$ Cperms(*Super User*, c_k) ≠ perms(*Super User*)

$\bigcup_{k=1}^{2}$ Cperms(*Super User*, c_k)={a, b, c, f}, perms(*Super User*) = ∅

Because **Theorem 1** is false, **Definition 1** can detect *inconsistency 2* (IC2)

3) Inconsistency 3-1

If the permission h is added to a role, *User*, the **Theorem 1** is as follows:

Theorem 1: $\bigcup_{k=1}^{2}$ Cperms(*User*, c_k) ≠ perms(*User*)

$\bigcup_{k=1}^{2}$ Cperms(*User*, c_k) = {a, b, c, f}, perms(*User*) = {a, b, c, f, h}

Because **Theorem 1** is false, **Definition 1** can detect *inconsistency 3-1* (IC3-1)

4) Inconsistency 3-2

If the permission g is deleted from a role, *Administrator*, the **Theorem 1** is as follows:

Theorem 1: $\bigcup_{k=1}^{2}$ Cperms(*Administrator*, c_k) ≠ perms(*Administrator*)

$\bigcup_{k=1}^{2}$ Cperms(*Administrator*, c_k) = { a, b, c, e, f, g, h }

perms(*Administrator*) = { a, b, c, e, f, h }

Because **Theorem 1** is false, **Definition 1** can detect *inconsistency 3-2* (IC3-2)

5) Inconsistency 4

If component C is added, the **Theorem 2** is as follows:

Theorem 2: roles(i) ≠ Croles(i, C)

roles(i) = ∅, Croles(i, C) = { *Administrator* }

Because **Theorem 2** is false, **Definition 1** can detect *inconsistency 4* (IC4)

6) Inconsistency 5

If component C is deleted, the **Theorem 1** is as follows:

Theorem 1: $\bigcup_{k=1}^{1}$ Cperms(*User*, c_k) ≠ perms(*User*)

$\bigcup_{k=1}^{1}$ Cperms(*User*, c_k) = { b, c }, perms(*User*) = { b, c, f, h }

Because **Theorem 1** is false, **Definition 1** can detect *inconsistency 5* (IC5)

7) Inconsistency 6-1

If permission n is added to component A, the **Theorem 2** is as follows:

Theorem 2: $\text{roles}(n) \neq \text{Croles}(n, A)$

$\text{roles}(n) = \varnothing, \text{Croles}(n, A) = \{ \textit{Adult User} \}$

Because **Theorem 2** is false, **Definition 1** can detect *inconsistency 6-1* (IC6-1)

8) Inconsistency 6-2

If permission a is deleted from component A, the **Theorem 1** is as follows:

Theorem 1: $\bigcup_{k=1}^{2} \text{Cperms}(\textit{User}, c_k) \neq \text{perms}(\textit{User})$

$\bigcup_{k=1}^{2} \text{Cperms}(\textit{User}, c_k) = \{b, c, f, h\}, \text{perms}(\textit{User}) = \{a, b, c, f, h\}$

Because **Theorem 1** is false, **Definition 1** can detect *inconsistency 6-2* (IC6-2)

Definition 1 keeps up the consistency of the authorization policies between the component and the RBAC server. A security manager can effectively manage the authorization policies of many components through **Definition 1**. If the inconsistency is detected between two policies, the inconsistency with synchronization process needs to be solved. **Definition 1** is used as criteria which decide that synchronization is successfully performed as well.

5 Conclusions

The component is distributed on a network with its authorization policies. On the other hand the RBAC server manages all component authorization policies in the same domains and guarantees the integrity of policies. Therefore in order to satisfy access control, it is necessary to guarantee the authorization policy consistency between the RBAC server and the components.

In this paper, the conditions to detect inconsistencies are proposed. The conditions of guaranteeing the integrity of component authorization policies are also proposed. In a case study, the conditions are validated and are used to analyze the inconsistencies. A consistency checking tool based on the proposed conditions can be developed and enable to detect inconsistencies of component authorization policies without significant efforts.

For future studies, it is necessary to study on the synchronization algorithms which can solve the detected inconsistency using the proposed conditions. The automated inconsistency detection tool and an inconsistency solver based on the synchronization algorithms are under development.

References

1. Gail-Joon Ahn, Ravi Sandhu, Role-Based Authorization Constraints Specification, ACM Transactions on Information and System Security, Vol. 3, No. 4, November 2000, pages 207-226.

2. Konstantin Beznosov and Yi Deng, "A Framework for Implementing Role-based Access Control Using CORBA Security Service" In Proceedings of the Fourth ACM Workshop on Role-Based Access Control, pages 19-30, Fairfax, Virginia, USA, October 1999.
3. David F.Ferraiolo et al. Role-Based Access Control, Artech House, 2003
4. David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, Proposed NIST Standard for Role-Based, Access Control, ACM Transactions on Information and System Security, Vol. 4, No. 3, August 2001, pages 224-274.
5. Bert Hartman et al., Enterprise Security with EJB and CORBA, WILEY, 2001, ch8 Scaleable Security Policies.
6. Chang-Joo Moon, Dae-Ha Park, Soung-Jin Park, Doo-Kwon Baik, Symmetric RBAC Model that Takes the Separation of Duty and Role Hierarchies into Consideration, Computers & Security, Vol.23, No.2, 2004 March, 126~136.
7. Sylvia Osborn, Ravi Sandhu, Qamar Munawer, Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies, ACM Transactions on Information and System Security, Vol. 3, No. 2, May 2000, Pages 85–106.
8. OMG, CORBA Component Model, v3.0, ch1 Component Model
9. Ravi S. Sandhu, Edward J. Coynek, Hal L. Feinsteink, Charles E. Youmank, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.
10. Ravi Sandhu, David Ferraiolo, Richard Kuhn, The NIST Model for Role-Base Access Control: Toward A Unified Standard, ACM Transactions on Information and System Security August 2001 Volume 4 Issue 3
11. Sun Microsystems, Enterprise JavaBeansTM Specification Version 2.1, ch4 Enterprise Beans as components

Custodian-Hiding Verifiable Encryption*

Joseph K. Liu[1], Victor K. Wei[1], and Duncan S. Wong[2]

[1] Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
{ksliu9,kwwei}@ie.cuhk.edu.hk
[2] Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
duncan@cityu.edu.hk

Abstract. In a verifiable encryption, an asymmetrically encrypted ciphertext can be publicly verified to be decipherable by a designated receiver while maintaining the semantic security of the message [2, 6, 9]. In this paper, we introduce *Custodian-Hiding Verifiable Encryption*, where it can be publicly verified that there exists at least one custodian (user), out of a designated group of n custodians (users), who can decrypt the message, while the semantic security of the message and the anonymity of the actual decryptor are maintained. Our scheme is proven secure in the random oracle model. We also introduce two extensions to decryption by a subset of more than one user.

Keywords: Verifiable Encryption, Publicly Verifiable, Anonymity

1 Introduction

We introduce the new paradigm *custodian-hiding verifiable encryption, CH-VE*. It allows a sender to verifiably encrypt a message to a group of receivers in a way that only one of them is able to decrypt it. In addition, any public verifier can ascertain this fact while he knows nothing about the plaintext and cannot compute the identity of the actual decryptor. Before formal definitions, we give a few motivating applications of CH-VE.

First consider the following scenario. Alice wants to send a public-key encrypted message to Bob, who works for ABC Company. For some security reason the company gateway system does not allow the message in unless it is for a company employee. However, Bob does not wish to divulge his private key. Without knowing Bob's private key, how can the gateway ensure the message is intended for a company employee? Furthermore, Alice and Bob do not want the company gateway to know that Bob is the actual recipient. By knowing only the public information of the company employees, the gateway system has to

* The work described in this paper was fully supported by a grant from CityU (Project No. 9360087).

determine if the encrypted incoming data is for a company employee without being able to identify the actual recipient. In addition, other employees of the company should not be involved in the secret communication between Alice and Bob and should be totally unaware of the entire process.

Another scenario is about key escrow. In a key escrow, Alice encrypts a secret key under the public key of a custodian and sends to an organization or a government this ciphertext together with a proof that the ciphertext is indeed an encryption of her own secret key. In order to increase the level of trust, n custodians maybe used instead of one single custodian. The key is shared among a particular set of t custodians so that only by these t custodians cooperating together can decrypt the secret of Alice while other subset of custodians cannot do so. It is more secure as the organization or the government does not know which particular t custodians know the secret. It takes an exponential time to find out which those t custodians are if t is equal to half of n.

In this paper, we present solutions to the above problems as well as some other application problems. Below, we briefly review verifiable encryption before introducing custodian-hiding verifiable encryption.

Verifiable Encryption. A verifiable encryption [20, 2, 6, 4, 9] allows a prover to encrypt a message and sends to a receiver such that the ciphertext is publicly verifiable. That is, any verifier can ensure the ciphertext can be decrypted by the receiver yet knowing nothing about the plaintext. There are numerous applications of verifiable encryption. For example, in a publicly verifiable secret sharing scheme [20], a dealer shares a secret with several parties such that another third party can verify that the sharing was done correctly. This can be done by verifiably encrypting each shares under the public key of the corresponding party and proves to the third party that the ciphertext encrypt the correct shares. Another scenario is in a fair exchange environment [2], in which both parties want to exchange some information such that either each party obtain the other's data, or neither party does. One approach is to let both parties verifiably encrypt their data to each other under the public key of a trusted party and then to reveal their data. If one party refuses to do so, the other can go to the trusted party to obtain the required data. Verifiable encryption can be also applied in revokable anonymous credential [7]. When the administration organization issues a credential, it verifiably encrypts enough information under the public key of the anonymity revocation manager, so that later if the identity of the credential owner needs to be revealed, this information can be decrypted.

Custodian-Hiding Verifiable Encryption (CH-VE). In a Custodian-Hiding Verifiable Encryption (CH-VE), a *Prover* is to send a public-key encrypted message to one among n *Custodians* through a *Verifier*. The Prover and the Verifier agree upon a group of n public keys, and then conduct an interactive protocol such that, if the Verifier is satisfied and relays a ciphertext from Prover to the n decryptors, at least one of the decryptor can recover the message. Furthermore, the message is semantically secure to the Verifier, and the identity of the actual decryptor is anonymous to the Verifier. CH-VE can solve the motivating applications earlier.

Receiver-Oblivious Transfer. CH-VE can be also regarded as a form of "*Receiver-Oblivious Transfer*". It is a dual to the following equivalent formulation of (interactive) *Oblivious Transfer (OT)* [18, 15, 5, 11–13, 1, 17]:

1. Verifier sends auxiliary encryption parameters.
2. Sender encrypts n messages to n public keys.
3. Verifier processes and then relays n ciphertexts to n Decryptors, at most one Decryptor can recover its message.

CH-VE does the following:

1. Sender sends n ciphertexts.
2. Verifier challenges. If satisfied with responses, relays ciphertexts to n Decryptors, at least one Decryptor can recover its message.

In OT, Sender is oblivious of the identity of the capable Decryptor. In CH-VE, Verifier is oblivious of the identity of the capable Decryptor. From this perspective, we consider CH-VE as a dual paradigm to the important fundamental paradigm of OT.

We also introduce two extensions. In the first extension a *targeted* subset of t custodians (out of n) jointly recover the message. In the second extension, any member of a *targeted* subset of t custodians can recover the encrypted message. Both extensions preserve the anonymity of the targeted subset.

These extensions can be useful in the following scenario. Bob belongs to a cluster of t members in a group of n members. For example, the cluster can be a small unit in a temporarily formed task force for a special mission. Our extension schemes can be used to transmit confidential messages to the unit, or to unit members, while keeping non-unit members of the task force and the security gateway of the task force at bay.

Contributions: We introduce a new paradigm: Custodian-Hiding Verifiable Encryption (CH-VE), which is an extension of (ordinary) Verifiable Encryption (VE). It retains the basic properties of VE: Message is encrypted to a designated decryptor/custodian in such a way that a public third party can verify that fact while knowing nothing about the plaintext. CH-VE adds the following basic anonymizing property: The decryptor/custodian is anonymized, in such a way that it is indistinguishable/hidden among a designated group of n decryptors/custodians. The public verifier can ensure one of the n decryptors/custodians can actually decrypt the message, but the verifier cannot identify the actual decryptor.

We present formal security models and definitions of security notions of CH-VE. The models are very strong models, formulating the anonymity in terms of IND-CCA2 games, with random oracle, decryptor oracle, and colluder oracle.

We present two constructions of CH-VE for DL homomorphic image. The cut-and-choose methodology is used [2, 6]. The schemes are proven secure in the random oracle model. The security is reduced to that of the hashing and the underlying encryption. The second of our constructions is secure against some Decryptors colluding with Verifier.

Our schemes support perfectly separability [16, 8], where Decryptors can use different encryption functions.

We also introduce the new paradigm of Custodian-Hiding Group Verifiable Encryption (CH-GVE). The first one requires a *particular* subset of t custodians out of n custodians to work jointly in decrypting the message while other subsets cannot. The second extension allows any member of a *particular* subset of t custodians to decrypt the message while members outside this subset cannot. Both extensions preserve the anonymity of the targeted decipherers.

The CH-VE is different from Group VE of [6]. The latter can verify encryption to a group of n decryptors where any subset of decryptors satisfying an access structure can recover the message. However, the Verifier always knows the entire access structure, and therefore knows which subset of decryptors can recover. There is no anonymizing of the designated decryptors. In CH-VE, the identities of the designated decryptors who can jointly recover is anonymized, and uncomputable by the Verifier.

Organization: This paper is organized as follows. We describe some related work in Sec. 2. This is followed by our security model specified in Sec. 3. Our basic schemes are described in Sec. 4. The paper is concluded in Sec. 5.

2 Related Work

Verifiable Encryption (VE) was first introduced by Stadler [20] in 1996 for the use of publicly verifiable secret sharing scheme [10]. The VE is based on ElGamal's public key system [14]. It allows a public verifier to determine if a ciphertext contains the discrete logarithm of a given value without decrypting it. The scheme uses the cut-and-choose methodology. Later, Asokan, et al. [2] presented a very general cut-and-choose based VE for encryption of pre-image of any homomorphic one-way function. Their scheme also provides *perfect separability* in such a way that the scheme can take any type of encryption algorithm and encryption key associated to the receiver. Camenisch, et al. [6] proposed another VE which is also perfectly separable and is proven secure without relying on random oracle. It is not limited to homomorphic one-way function but generalized to any boolean relation. Bao [4] proposed a deterministic VE for discrete logarithm without using the cut-and-choose methodology. Camenisch, et al. [9] propose another VE for discrete logarithm without using cut-and-choose methodology and achieved provable security under chosen-ciphertext-attack security model. Ateniese [3] also propose an efficient VE for digital signatures.

In all the above schemes the verifier knows the identity of the receiver. An anonymous verifiable encryption scheme proposed by Camenisch, et al. [7] hides the identity of the receiver from the verifier. Their scheme requires the prover to know the private key of the receiver.

Comparisons with Group Verifiable Encryption (GVE) [6]: Camenishch and Damgård proposed two schemes in [6]. The first one is a VE scheme which is targeted for one decryptor only. The second one is a Group VE (GVE) scheme which is targeted for any t decryptors. They allow the prover to choose an access structure Γ for a group of n receivers such that *any* subset of t members can jointly recover the message. The access structure can be instantiated by a secret sharing scheme. That is, the prover divides the message m into n pieces of shares $m_1, \ldots, m_n$ such that any t of them are enough to reconstruct m. Then he encrypts m_i using the encryption function of user i, for $i = 1, \ldots, n$, and sends all ciphertext to the verifier. It is clear that the message m can be reconstructed if any t users decrypt their corresponding ciphertext to get the shares.

In summary, their GVE scheme is a threshold version of their VE scheme by using a secret-sharing scheme to share the escrowed information such that any qualified subset (under the access structure) of custodians/decryptors can jointly recover the information. But the Verifier knows the identities of each subset of custodians who can recover the information.

Our proposed CH-VE basic scheme hides the identity of the one *targeted* custodian among n possible custodians. Others $n-1$ custodians cannot decrypt the message. The identity of the "trusted" or "designated" custodian is anonymized to n possible custodians. Our extensions, the CH-GVE schemes deploy some threshold-sharing technique of the escrowed secret (plaintext message) to t possible custodians and hiding the identity of the t "trusted" or "designated" custodians among n possible custodians (in certain ways).

In general, all our proposed scheme allows the escrowed information to be shared out among a specific subgroup of t custodians. The Verifier is assured that there exists t among the total population of n custodians that can recover the escrowed information, but the Verifier cannot compute the identities of these t custodians.

3 Security Model

In this section we define the security model to be used.

An CH-VE scheme is a tuple $(\mathcal{G}, P, V, R)$. Components specified below.

A PPT (polynomial probabilistic time) algorithm $\mathcal{G}(1^{\lambda_s})$, on input security parameter λ_s, generates n pairs of encryption functions E_i (with public key PK_i) and decryption functions D_i (with private key SK_i), for $i = 1, \ldots, n$, which are secure against chosen-ciphertext attack [19].

There is a two-party PPT protocol between P (Prover) and V (Verifier), a PPT algorithm R (Recovery Algorithm), also known as the Decryption Algorithm, and a homomorphic one-way function f. P accepts as inputs the security parameter λ_s, some appropriate binary string m (a *message*), n public-key encryption functions $\{E_i\}_{1 \le i \le n}$ which are secure against chosen-ciphertext attack [19], and an integer $\pi \in \{1, \cdots, n\}$ (the index of the actual decryptor). V accepts as inputs λ_s, f and $\{E_i\}_{1 \le i \le n}$ only. If the protocol completes without early termination, V outputs two finite binary strings $d = f(m)$ (homomorphic image) and $\mathcal{C}$ (ciphertext). We call it successful. Otherwise, V outputs Reject.

The Recovery Algorithm $R : (d, \mathcal{C}, D_i) \mapsto \{m', NULL\}$ accepts as inputs the homomorphic image, the ciphertext and one of the n decryption functions corresponding to $\{E_i\}_{1 \le i \le n}$ and outputs either a finite string m' or the $NULL$ string.

We formulate the security definition more precisely as below:

Definition 1. *The CH-VE scheme is* complete *if V always outputs homomorphic image d and ciphertext $\mathcal{C}$ such that $R(\mathcal{C}, D_\pi) = m$ and $d = f(m)$ for some $1 \le \pi \le n$, for arbitrary input m, homomorphic image d and ciphertext $\mathcal{C}$ satisfying $d = f(m)$, whenever both P and V are honest.*

Definition 2. *The CH-VE scheme is* sound *if, whenever V completes a protocol run without early termination and outputs homomorphic image d and ciphertext $\mathcal{C}$, then with overwhelming probability $f(R(\mathcal{C}, D_\pi)) = d$ for some π, $1 \le \pi \le n$.*

Next we define some oracles we are going to use in this model:

Prover Oracle $\mathcal{PO}$: Upon query, it interacts with the querier in the role of the Prover P.

Decryption Oracle $\mathcal{DO}$: Upon input a ciphertext, decrypt it or output Reject on invalid ciphertext.

Colluder Oracle $\mathcal{CO}$: Upon input a public key generated by $\mathcal{G}$, output its corresponding secret key.

We define a game, **Game D**, between Adversary $\mathcal{A}$ and Simulator $\mathcal{S}$:

1. *Setup Phase:* Algorithm $\mathcal{G}$ is invoked to generate n key pairs. The public keys $\mathsf{PK}_1, \cdots, \mathsf{PK}_n$, are published.
2. *Probe-1 Phase:* $\mathcal{A}$ queries $\mathcal{DO}$, $\mathcal{PO}$, and $\mathcal{CO}$.
3. *Gauntlet Phase:* $\mathcal{A}$ generates message m_1, decryptor identity π_1, $1 \le \pi_1 \le n$, and give them to $\mathcal{S}$. $\mathcal{S}$ generates message m_0, decryptor identity π_0, and $b_G \in_R \{0, 1\}$; computes and sends to $\mathcal{A}$ the CH-VE encryption, consisting of homomorphic image d and ciphertext $\mathcal{C}$, of message m_{b_G} targeted for decryptor π_{b_G}.
4. *Probe-2 Phase:* $\mathcal{A}$ interacts, as Verifier, with $\mathcal{S}$ as an honest Prover while querying $\mathcal{DO}$, $\mathcal{PO}$, and $\mathcal{CO}$. Except $\mathcal{A}$ cannot query $\mathcal{DO}$ with the gauntlet ciphertext $\mathcal{C}$.
5. *Output Phase:* $\mathcal{A}$ outputs its estimate $\hat{b}_G$ of b_G

We say $\mathcal{A}$ wins the Game if $\hat{b}_G = b_G$. There are two sub-games: **Game D-C:** $\pi_0 = \pi_1$ and $\mathcal{S}$ chooses m_0 randomly. The *advantage* of $\mathcal{A}$ is its probability of winning the game, minus 1/2. **Game D-A:** $m_0 = m_1$, $\mathcal{S}$ choses π_0 randomly from $\{1, \cdots, n\} \setminus \{\pi_1\}$. The *advantage* of $\mathcal{A}$ is its probability of winning the game, minus the probability of winning the game by random guessing. The latter probability equals $\frac{1}{2}(1 + \frac{q_C}{n-1})$ where q_C is the number of Colluder Oracle queries. We require $q_C \le n - 1$, and $\mathcal{A}$ cannot query $\mathcal{CO}$ with PK_{π_1}.

Definition 3. (Zero Knowledge) *The CH-VE scheme is* zero-knowledge *if no PPT adversary $\mathcal{A}$ can win Game D-C or Game D-A with non-negligibly advantage. It is* no-colluder zero-knowledge *if no PPT adversary $\mathcal{A}$ can win Game*

D-C or Game D-A with non-negligibly advantage without making any query to the Colluder Oracle.

Definition 4. *A CH-VE scheme is* secure *if it is complete, sound, and zero-knowledge. It is* no-colluder secure *if it is complete, sound, and no-colluder zero-knowledge.*

Remark: In Game D-C and when $b_G = 0$, the honest prover $\mathcal{S}$ does not use m_1 at all. This fact can be used to prove that our zero knowledge implies the zero knowledge in Asokan, et al.[2], p.599, l.13.

4 Secure Custodian Hiding Verifiable Encryption (CH-VE) Schemes

We specify two schemes in this section. The first scheme is a no-colluder-secure CH-VE scheme while the second one is secure even with the existence of colluders. Both schemes use the 3-choice cut-and-choose methodology. In each of N cut-and-choose rounds, a three-move protocol (commit, challenge, respond) is conducted between Prover P and Verifier V. V flips a three-way coin to issue one of three possible challenges. Depending on the challenge, P provides suitable response. If all cut-and-choose rounds are satisfactory, V outputs a image d and a ciphertext $\mathcal{C}$. Otherwise, it aborts. Each receiver i attempts to decipher using its own asymmetric decryption function D_i, $1 \leq i \leq n$. At least one receiver will succeed.

4.1 A No-Colluder-Secure CH-VE Scheme

Let (E_i, D_i), $1 \leq i \leq n$, be secure public-key encryption and decryption functions generated by $\mathcal{G}$. Let π index the targeted receiver. Let p, q be large primes, $q \mid p$–1, and $g \in F_p$, order(g)=q. Let the security parameter λ_s be as large as $|q|$. Let f be defined by $x \rightarrow g^x$ which is an instantiation of the one-way group homomorphism from $\mathbb{Z}_q$ to $<g>$.

Let $m \in \mathbb{Z}_q$ be a message. Let N be the number of cut-and-choose rounds. Let $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{\lambda_s}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be some statistically independent and cryptographically strong hash functions.

Sometimes, we may pass in an element in $\mathbb{Z}_q$ for encryption and we implicitly assume that certain appropriate encoding method is applied.

When the Prover P computes any probabilistic public-key encryption function, P needs to send the corresponding coin flip sequence to the Verifier V and the sequence is to be carried on wherever the original message goes. We do not explicitly specify such in the following.

Sym(n) denotes the symmetric group of order n. It consists of all permutations on n objects. We instantiate a one-way homomorphic mapping of m, typically used in VE (verifiable encryption) literature [2, 6, 9] by discrete exponentiation g^m, and assume DLP (discrete logarithm problem) to be secure. We do so throughout the paper.

[**Protocol Between P and V (Encryption)** (Illustrated in Figure 1)]

1. P computes $d = g^m \bmod p$ and sends d to V.
2. Repeat the following steps N times in parallel.
   a. (*Commitment*) P randomly picks $s \in_R \mathbb{Z}_q$, $r_i \in_R \{0,1\}^{\lambda_s}$ for $1 \le i \le n$, and $\phi \in_R \mathrm{Sym}(n)$. P computes

$$\begin{aligned}\lambda &= E_1(r_1)||\cdots||E_n(r_n)\\ \gamma &= (g^{H_2(r_{\phi(1)})} \bmod p,\ \cdots,\ g^{H_2(r_{\phi(n)})} \bmod p)\\ \alpha' &= E_1(s)||\cdots||E_n(s)\\ \alpha &= H_1(\alpha')\\ \beta &= g^{H_2(r_\pi)s} \bmod p\\ \theta &= H_1(\lambda||\gamma||\alpha||\beta)\end{aligned}$$

   P sends θ to V.

   b. (*Challenge*) V picks $b \in_R \{1,2,3\}$ and sends b to P.
   c. (*Response*)
      – Case $b = 1$, P sends $r_1, \cdots, r_n$, γ, α and β to V
      – Case $b = 2$, P sends λ, γ, and s to V.
      – Case $b = 3$, P sends λ, γ, α', and $s' = H_2(r_\pi)s + m \bmod q$ to V.
   d. (*Verification* by V)
      – Case $b = 1$:
        • Verify that r_1, $\cdots$, r_n are distinct.
        • Verify that there exists a unique permutation $\delta \in \mathrm{Sym}(n)$ such that $\gamma = (g^{H_2(r_{\delta(1)})} \bmod p,\ \cdots,\ g^{H_2(r_{\delta(n)})} \bmod p)$
        • Verify that $\theta = H_1(\hat{\lambda}||\gamma||\alpha||\beta)$ where $\hat{\lambda} = E_1(r_1)||\cdots||E_n(r_n)$.

        Continue only if all verifications succeed.
      – Case $b = 2$:
        • Denote $\gamma = (\gamma_1, \cdots, \gamma_n)$.
        • Compute $\tilde{\alpha} = H_1(E_1(s)\ ||\ \cdots\ ||\ E_n(s))$ and $\beta_i = \gamma_i^s \bmod p$, for $i = 1, \cdots, n$.
        • Verify that $\theta = H_1(\lambda\ ||\ \gamma\ ||\ \tilde{\alpha}\ ||\ \beta_i)$ for exactly one index $i \in \{1, \cdots, n\}$.

        Continue only if the verification succeeds.
      – Case $b = 3$:
        • Compute $\beta' = g^{s'}/d \bmod p$
        • Verify that $\theta = H_1(\lambda||\gamma||H_1(\alpha')||\beta')$

        Continue only if the verification succeeds.
3. (*Output*) V terminates if any verification fails in any of the N cut-and-choose Rounds. Otherwise, it outputs d and the four-tuple sequences $(\alpha', \lambda, \beta', s')$ for all Case-($b{=}3$) Rounds to all n receivers as the ciphertext, $\mathcal{C}$.

Recovery Algorithm R.
Denote $\bar{\lambda}_1||\cdots||\bar{\lambda}_n = \lambda$ and $\bar{\alpha'}_1||\cdots||\bar{\alpha'}_n = \alpha'$. For d and each four-tuple sequence $(\alpha', \lambda, \beta', s')$, each receiver i, $1 \le i \le n$, independently performs the following steps.

Prover P **Verifier V**

$d = g^m \bmod p$

$$\xrightarrow{\quad d \quad}$$

Repeat N times:

$s \in_R \mathbb{Z}_q$

$r_i \in_R \{0,1\}^{\lambda_s}$, $1 \le i \le n$

$\phi \in_R \mathrm{Sym}(n)$

$\lambda = E_1(r_1)|| \cdots ||E_n(r_n)$

$\gamma = (g^{H_2(r_{\phi(1)})} \bmod p,\ \cdots,\ g^{H_2(r_{\phi(n)})} \bmod p)$

$\alpha' = E_1(s)|| \cdots ||E_n(s)$

$\alpha = H_1(\alpha')$

$\beta = g^{H_2(r_\pi)s} \bmod p$

$\theta = H_1(\lambda||\gamma||\alpha||\beta)$

$$\xrightarrow{\quad \theta \quad}$$

$b \in_R \{1,2,3\}$

$$\xleftarrow{\quad b \quad}$$

Case $b = 1$

$$\xrightarrow{\quad r_1, \cdots, r_n, \gamma, \alpha, \beta \quad}$$

$r_i \overset{?}{\neq} r_j,\ \forall i,j \in \{1,\cdots,n\}, i \neq j$

$\overset{?}{\exists}\ \delta \in \mathrm{Sym}(n),\ \gamma = (g^{H_2(r_{\delta(1)})}, \cdots, g^{H_2(r_{\delta(n)})})$

$\hat{\lambda} = E_1(r_1)|| \cdots ||E_n(r_n)$

$\theta \overset{?}{=} (\hat{\lambda}||\gamma||\alpha||\beta)$

Case $b = 2$

$$\xrightarrow{\quad \lambda, \gamma, s \quad}$$

$(\gamma_1, \cdots, \gamma_n) \leftarrow \gamma$

$\tilde{\alpha} = H_1(E_1(s)|| \cdots ||E_n(s))$

$\beta_i = \gamma_i^s \bmod p$, $1 \le i \le n$

$\overset{?}{\exists}\ i \in \{1,\cdots,n\},\ \theta = H_1(\lambda||\gamma||\tilde{\alpha}||\beta_i)$

Case $b = 3$

$s' = H_2(r_\pi)s + m \mod q$

$$\xrightarrow{\quad \lambda, \gamma, \alpha', s' \quad}$$

$\beta' = g^{s'}/d \bmod p$

$\theta \overset{?}{=} H_1(\lambda||\gamma||H_1(\alpha')||\beta')$

If no rejection in all N rounds,
output d and $(\alpha', \lambda, \beta', s')$ for all Case-($b = 3$) rounds

Fig. 1. A custodian-hiding verifiable encryption scheme.

1. Compute $r_i = E_i^{-1}(\bar{\lambda}_i)$ and $s = E_i^{-1}(\bar{\alpha'}_i)$.
2. Compute $m' = s' - H_2(r_i)s \bmod q$.
3. Verify that $g^{s'} = g^{m'}\beta' \bmod p$. If the verification succeeds, then receiver i is the targeted decypherer and it outputs the decrypted message m' and halts. Otherwise, the receiver repeats the steps for another four-tuple sequence.

4.2 Security Analysis

Theorem 1. *Our CH-VE scheme above is no-colluder secure under the random oracle model, provided all encryptions are IND-CCA2 secure.*

Proof in the Appendix.

Practical Security and Performance: We recommend N to be approximately $80-100$ which is equivalent to about 2^{-64} and it should be sufficient for most applications. We have in mind OAEP for the component encryptions and elliptic curve is used as the homomorphic one-way function. Let the length of each encryption be 1024 bits. The communication bandwidth is about 4096 bits for each round. If there are 100 rounds, the total bandwidth is about $50n$ kbytes and the size of the ciphertext is about $17n$ kbytes, where n is the number of custodians. Our bandwidth is larger than the VE scheme of [2], for preserving the anonymity of the actual decryptor.

4.3 A Secure Custodian Hiding Verifiable Encryption (CH-VE) Scheme (with Colluders)

The previous CH-VE scheme is not secure if Adversary has a colluder. In that case Adversary obtains s from Colluder, then exhaustively tests out $g^{m_b} \overset{?}{=} g^s g^{H_2(r_i)}$ to distinguish m_0 from m_1. In this section, we modify the above scheme to provide security even in the existence of colluders. We use the same notation as described in the first scheme unless otherwise stated.

[**Protocol Between P and V (Encryption).** Let $L = \log n$]

1. P computes $d = g^m \bmod p$ and sends d to V.
2. Repeat the following steps N times in parallel.
   a. (*Commitment*) P randomly picks $s \in_R \mathbb{Z}_q$, $r_i^{(j)} \in_R \{0,1\}^{\lambda_s}$ for $1 \le i \le n$, $1 \le j \le L$ and $\phi_j \in_R \mathrm{Sym}(n)$ for $1 \le j \le L$. P computes
   $$\begin{aligned}\lambda^{(j)} &= E_1(r_1^{(j)})||\cdots||E_n(r_n^{(j)})\\ \gamma^{(j)} &= (g^{H_2(r^{(j)}_{\phi_j(1)})} \bmod p,\ \cdots,\ g^{H_2(r^{(j)}_{\phi_j(n)})} \bmod p)\\ \alpha' &= E_1(s)||\cdots||E_n(s), \quad \alpha = H_1(\alpha')\\ \beta^{(j)} &= g^{H_2(r_\pi^{(j)})s} \bmod p\\ \theta^{(j)} &= H_1(\lambda^{(j)}||\gamma^{(j)}||\alpha||\beta^{(1)}\ \cdots\beta^{(L)})\end{aligned}$$
   and sends $\theta^{(j)}$ to V, for $1 \le j \le L$.
   b. (*Challenge*) V picks $b \in_R \{1,2,3\}$ and sends b to P.
   c. (*Response*)
      - Case $b = 1$, P sends $r_1^{(j)}, \cdots, r_n^{(j)}$, $\gamma^{(j)}$, α and $\beta^{(j)}$ to V, for $1 \le j \le L$.
      - Case $b = 2$, P sends $\lambda^{(j)}$, $\gamma^{(j)}$, and s to V, for $1 \le j \le L$.
      - Case $b = 3$, P sends $\lambda^{(j)}$, $\gamma^{(j)}$, α', and $s' = (H_2(r_\pi^{(1)}) + \ldots + H_2(r_\pi^{(L)}))s + m \bmod q$ to V, for $1 \le j \le L$.

d. (*Verification* by V)
  - Case $b = 1$:
    - Verify that $r_1^{(j)}, \cdots, r_n^{(j)}$ are distinct, for $1 \leq j \leq L$.
    - Verify that there exists a unique permutation $\delta \in \mathrm{Sym}(n)$ s.t. $\gamma^{(j)} = (g^{H_2(r_{\delta(1)}^{(j)})} \bmod p, \ \cdots, \ g^{H_2(r_{\delta(n)}^{(j)})} \bmod p)$ for $1 \leq j \leq L$.
    - Verify that $\theta^{(j)} = H_1(\hat{\lambda}^{(j)}||\gamma^{(j)}||\alpha||\beta^{(1)} \cdots \beta^{(L)})$ where $\hat{\lambda^{(j)}} = E_1(r_1^{(j)})|| \cdots ||E_n(r_n^{(j)})$, for $1 \leq j \leq L$.

    Continue only if all verifications succeed.
  - Case $b = 2$:
    - Denote $\gamma^{(j)} = (\gamma_1^{(j)}, \cdots, \gamma_n^{(j)})$.
    - Compute $\tilde{\alpha} = H_1(E_1(s) \ || \ \cdots \ || \ E_n(s))$ and $\beta_i^{(j)} = (\gamma_i^{(j)})^s \bmod p$, for $1 \leq i \leq n$, $1 \leq j \leq L$.
    - Verify that $\theta^{(j)} = H_1(\lambda^{(j)}||\gamma^{(j)}||\tilde{\alpha}||\beta_{i_1}^{(1)} \cdots \beta_{i_L}^{(L)})$ for exactly one index $i_\ell \in \{1, \cdots, n\}$, $1 \leq \ell \leq L$, $1 \leq j \leq L$.

    Continue only if the verification succeeds.
  - Case $b = 3$:
    - Compute $\beta' = g^{s'}/d \bmod p$
    - Verify that $\theta^{(j)} = H_1(\lambda^{(j)}||\gamma^{(j)}||H_1(\alpha')||\beta')$ for $1 \leq j \leq L$.

    Continue only if the verification succeeds.

3. (*Output*) V terminates if any verification fails in any of the N cut-and-choose Rounds. Otherwise, it outputs d and the four-tuple sequences $(\alpha', \lambda^{(j)}, \beta', s')$ for $1 \leq j \leq L$ and for all Case-($b = 3$) Rounds to all n receivers as the ciphertext, $\mathcal{C}$.

Recovery Algorithm R**.** Denote $\bar{\lambda}_1^{(j)}|| \cdots ||\bar{\lambda}_n^{(j)} = \lambda^{(j)}$ and $\bar{\alpha'}_1|| \cdots ||\bar{\alpha'}_n = \alpha'$. For d and each four-tuple sequence $(\alpha', \lambda^{(j)}, \beta', s')$, $1 \leq j \leq L$, each receiver i, $1 \leq i \leq n$, independently performs the following steps.

1. Compute $r_i^{(j)} = E_i^{-1}(\bar{\lambda}_i^{(j)})$ and $s = E_i^{-1}(\bar{\alpha'}_i)$.
2. Compute $m' = s' - (H_2(r_i^{(1)}) + \ldots + H_2(r_i^{(L)}))s \bmod q$.
3. Verify that $g^{s'} = g^{m'}\beta' \bmod p$. If the verification succeeds, then receiver i is the targeted decypherer and it outputs the decrypted message m' and halts. Otherwise, the receiver repeats the steps for another four-tuple sequence.

Theorem 2. *The above CH-VE scheme is secure under the random oracle model, provided that all encryptions are IND-CCA2 secure.*

A sketch of the proof is in the Appendix.

4.4 Extensions

We present extensions of our CH-VE schemes to CH-GVE (Custodian-Hiding Group Verifiable Encryption) schemes.

Verifiable (t, t, n) Encryption for Anonymous Ad Hoc Groups. In our basic VE-VE scheme, only a single targeted member can decrypt the message. Here we make an extension such that a targeted t-member subset of the ad hoc group of n receivers can jointly recover the message. On the notation of (t, t, n), symbol 'n' represents that P spontaneously forms a group of n receivers; the second symbol 't' represents that t targeted members of the group can recover a message; and the first symbol 't' means that all the t targeted members need to work jointly to recover the message. By using similar notation, we propose another extension shortly which allows *any* member of a targeted t-member subset of the ad hoc group of n receivers to recover the message. Hence the notation of the second extension is $(1, t, n)$.

Below is a (t, t, n) CH-GVE scheme.

Encryption. Here we use $\pi_1, \cdots, \pi_t$ to index the targeted receivers, where $t < n$ and $\pi_1, \cdots, \pi_t \in \{1, \cdots, n\}$ are distinct. The encryption algorithm is similar to the basic scheme described in Sec. 4, with the following modifications.

1. P also sends t to V before Commitment.
2. (Commitment) Compute as before, except
$$\beta = (g^{H_2(r_{\pi_1})} \cdot g^{H_2(r_{\pi_2})} \cdot \ldots \cdot g^{H_2(r_{\pi_t})})^s \bmod p$$
3. (Response) Compute as before, except that in Case b=3:
$$s' = (H_2(r_{\pi_1}) + \cdots + H_2(r_{\pi_t}))s + m \bmod q$$
4. (Verification)
   (a) Case b=1: Same as before.
   (b) Case b=2: Process γ, $\tilde{\alpha}$, β_i as before. Verify
$$\theta = H_1(\lambda \mid\mid \gamma \mid\mid \tilde{\alpha} \mid\mid \beta_{i_1} \cdots \beta_{i_t})$$
   for a unique t-element subset $\{i_1, \cdots, i_t\} \subset \{1, \cdots, n\}$.
   (c) Case b=3: No change.

Decryption. Same as before, except that t targeted decypherers jointly compute
$$m' = s' - (H_2(r_{\pi_1}) + \cdots + H_2(r_{\pi_t}))s \bmod q$$

$(1, t, n)$ CH-GVE Scheme. The $(1, t, n)$ CH-GVE scheme allows *any* one in a targeted set of t receivers to recover the encrypted message, and then the receivers are anonymized.

Encryption. Let $\pi_1, \cdots, \pi_t$ be the index of t targeted receivers. The encryption algorithm is similar to before, but with small modifications:

1. P also sends t to V before Commitment.
2. (Commitment) Same as before except
$$\beta = g^{H_2(r_{\pi_1})s} \bmod p \mid\mid \cdots \mid\mid g^{H_2(r_{\pi_t})s} \bmod p$$

3. (Response) Same as before except in Case b=3, replace the original s' with

$$s'_i = H_2(r_{\pi_i})s + m \bmod q$$

for $i = 1, \cdots, t$.
4. (Verification) Same as before except in
   (a) Case b=2, verify that

$$\theta = H_1(\lambda \,||\, \gamma \,||\, \tilde{\alpha} \,||\, (\beta_{i_1}||\cdots||\beta_{i_t}))$$

   for a unique t-member ordered tuples $\{i_1, \cdots, i_t\} \subset \{1, \cdots, n\}$.
   (b) Case $b = 3$, compute

$$\beta' = g^{s'_1}/d \bmod p \,||\, \cdots \,||\, g^{s'_t}/d \bmod p$$

5. (Output) Same as before except replacing the original s' with $s'_1, \cdots, s'_t$.

Decryption
- Denote $\bar{\beta}'_1 \,||\, \cdots \,||\, \bar{\beta}'_t = \beta'$.
- Step 2 is modified as: receiver i computes $m'_{i,j} = s'_j - H_2(r_i)s \bmod q$ for $j = 1, \cdots, t$.
- Step 3 is modified as: receiver i checks if $g^{s'_j} \overset{?}{=} g^{m'_{i,j}}\bar{\beta}'_j \bmod p$ for $j = 1, \ldots, t$. If one of them equal, then receiver i is one of the targeted decypherers.

5 Concluding Remarks

In this paper, we propose a new paradigm of Custodian-Hiding Verifiable Encryption (CH-VE) which allows the prover to specify any set of n receivers and send an encrypted message such that the verifier can make sure that the encrypted message can be decrypted by at least one of the receivers. Yet the verifier knows nothing about the identity of the actual decryptor. The complexity of our proposing scheme is linear in the size of the receiver group. We give two instantiations of CH-VE with different level of security.

We believe that other intriguing and efficient CH-VE schemes and various security models can be attained. Other variants and features may also be constructed. For example, it would be interesting to construct a general verifiable (k, t, n) encryption scheme.

References

1. B. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: how to sell digital goods. In *Proc. EUROCRYPT 2001*, pages 119–135. Springer-Verlag, 2001. LNCS No. 2045.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Proc. EUROCRYPT 98*, pages 591–606. Springer-Verlag, 1998. LNCS No. 1403.

3. G. Ateniese. Verifiable encryption of digital signatures and applications. *ACM Transactions on Information and System Security*, 7(1):1–20, February 2004.
4. F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *Proc. Smart Card Research and Applications (CARDIS) 1998*, pages 213–220. Springer-Verlag, 2000. LNCS No. 1820.
5. G. Brassard, C. Crepeau, and J. Robert. Information theoretic reductions among disclose problem. In *Proc. 27th IEEE Symp. on Foundations of Comp. Science*, pages 168–173. IEEE, 1986.
6. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proc. ASIACRYPT 2000*, pages 331–345. Springer-Verlag, 2000. LNCS No. 1976.
7. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocations. In *Proc. EUROCRYPT 2001*, pages 93–118. Springer-Verlag, 2001. LNCS No. 2045.
8. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Proc. CRYPTO 99*, pages 413–430. Springer-Verlag, 1999. LNCS No. 1666.
9. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. LNCS No. 2729, 2003.
10. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proc. 26th IEEE Symp. on Foundations of Comp. Science*, pages 383–395, Portland, 1985. IEEE.
11. C. Crepeau. Equivalence between two flavours of oblivious transfers. In *Proc. CRYPTO 87*, pages 350–354. Springer-Verlag, 1987. LNCS No. 293.
12. C. Crepeau and J. Kilian. Weakening security assumptions and oblivious transfer. In *Proc. CRYPTO 88*, pages 2–7. Springer-Verlag, 1988. LNCS No. 403.
13. B. den Boer. Oblivious transfer protecting secrecy. In *Proc. EUROCRYPT 90*, pages 31–46. Springer-Verlag, 1990. LNCS No. 473.
14. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):496–472, July 1985.
15. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *Proc. CRYPTO 82*, pages 205–210. Springer-Verlag, 1982.
16. J. Kilian and E. Petrank. Identity escrow. In *Proc. CRYPTO 98*, pages 169–185. Springer-Verlag, 1998. LNCS No. 1642.
17. Y. Mu, J. Zhang, and V. Varadharajan. m out of n oblivious transfer. In *ACISP02*, pages 395–405. Springer-Verlag, 2002. LNCS No. 2384.
18. M. Rabin. How exchange secrets by oblivious transfer. Technical Report TR-81, Computer Science Laboratory, Harvard, 1981.
19. C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proc. CRYPTO 91*, pages 433–444. Springer, 1991. LNCS No. 576.
20. M. Stadler. Publicly verifiable secret sharing. In *Proc. EUROCRYPT 96*, pages 191–199. Springer-Verlag, 1996. LNCS No. 1070.

Appendix: Proofs

Due to length restrictions, proofs are left to the long version of this paper.

Proving Key Usage*

Malek Bechlaghem[1] and Vincent Rijmen[1,2]

[1] Cryptomathic NV
Interleuvenlaan 62/19, B-3001 Heverlee, Belgium
{malek.bechlaghem,vincent.rijmen}@cryptomathic.com
[2] IAIK, Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria
vincent.rijmen@iaik.tugraz.at

Abstract. In the context of a validation server, or more general PKI server, it could be interesting to give the server the possibility to check whether a certain public key was used. We investigate here some possibilities. We also present an escrow system as an application of the technology.

1 Introduction

In this paper we consider an IT system using a *PKI server*. Similar to the idea of authentication servers, which can provide centralized services with respect to access control, authentication and authorization, PKI servers provide centralized PKI services to client applications. After proper authentication to the server, the client can request PKI services. Such systems have been proposed in the past, for different kinds of reasons.

The PKI server introduced here has the capability to check whether a given ciphertext was produced with a given public key, without having access to the message plaintext. Both purely asymmetric and hybrid encryption schemes are considered.

This work is related to the concept of *verifiable encryption* [4]. Verifiable encryption schemes allow to prove certain properties about an encrypted message, without revealing the message. We aren't concerned with properties of the message however, but only examine the key used to encrypt it.

2 Background

We distinguish two classes of PKI servers. In both classes, we can further distinguish between partially trusted servers and completely trusted servers. In general, it is more easy to design systems using completely trusted servers. One design challenge is to reduce the required trust in centrally managed servers as much as possible.

* This work was supported by the funding of IWT Flanders, project: "ITEA 02011 SATURN: Security applications and technologies for universal information networks."

2.1 Voluntary PKI Servers

By voluntary PKI servers, we mean servers that provide services which could in principle be provided as well by the client applications. In order to reduce the complexity of PKI applications, the functionality is centralized. We could imagine a system where the functionality is provided by both the client applications and the server, and there is some criterion to select at run-time to perform the service locally or centrally.

A good example are validation servers that relieve the applications from complicated tasks like certificate chain construction, certificate status check, certificate attribute check (e.g. key usage) etc. [1, 9].

2.2 Mandatory PKI Servers

By mandatory PKI servers, we mean servers that can't be ignored by the client applications. The PKI trust model is based on the fact that the server plays its role. This type of servers is introduced not only in order to solve practical problems, but also some of the theoretical problems inherent to a distributed PKI model, like for instance providing both non-repudiation and revocation services.

A good example here are server-aided signatures like mediated RSA [3], a variant of RSA where the private key of the user is split into two parts, one owned by the user and the other by a trusted server so that both parts are used during an on-line protocol in order to create an RSA signature. On-line Trusted third parties (TTP) involved in non-repudiation and certified email protocols [5, 11] are another good example of mandatory PKI servers. Such TTP's are involved in order to provide mandatory non-repudiation evidence. The centralized PKI system described in this paper is of this type.

2.3 Setting

When used in an organization, it might be desired to configure the PKI server in a way that it not only provides services, but it also checks on the users/applications. For instance, it can be a policy that users/applications may not accept a signed message unless the PKI server has validated the signature. In such a setting, it may also be desirable that the server checks on outgoing messages. For instance, the server could verify whether the messages are encrypted using the proper keys.

In this paper, we study this problem: how can we give a PKI server the power to check whether outgoing messages are encrypted using the proper key material, without being able to read the messages.

Finding a solution to this problem can be use be useful in several settings. One typical setting could be such that there is outgoing data to destinations with different clearances and encryption keys at different strength levels. The PKI server acting for instance as an enterprize secure messaging gateway, can then check whether outgoing data has been encrypted with a key of the correct strength.

3 Proving Usage of a Public Key

We first discuss the situation where encryption is done with asymmetric encryption schemes. For this situation, we can reuse protocols that haven been proposed in the literature. The schemes will mainly be useful in the next section, where we discuss the situation of hybrid encryption.

In an asymmetric encryption setting, the PKI server sees a ciphertext c and wants to ascertain that c is the encryption of an unknown message m under a given public key. In many public-key schemes, an arbitrarily generated bit string will be a valid encryption of *some* message. Hence, we can rephrase the goal of the server to the verification of the fact that the sender *knows* the plaintext message m.

3.1 RSA Encryption

In an RSA setting, usage of a public key can be proven in the following way. Suppose Alice wants to send a message m, which encrypts to $c = m^e \bmod n$. Alice generates a random value r, computes the witness $x = r^e \bmod n$ and submits the tuple (c, x). Subsequently, the PKI server generates a challenge $u \in \{0, 1\}$ and submits it to Alice. Alice then computes the response $y = rm^u \bmod n$ and returns it to the server. Finally, the PKI server verifies whether $y^e = xc^u$.

This protocol corresponds to the Guillou-Quisquater identification protocol [7], but here Alice proves her knowledge of the message m instead of the secret accreditation data. Alice should not send out the same message m to many different recipients, nor should she re-use the random values r.

3.2 Rabin Encryption

If the Rabin encryption scheme [10] is used, the ciphertext c is constructed as $c = m^2 \bmod n$. The Fiat-Shamir identification protocol [6] can be used to prove knowledge of m without revealing m.

4 Proving Key Usage in a Hybrid Encryption Scheme

Purely asymmetric encryption is rarely used to protect messages. Instead, hybrid encryption schemes are used. We develop here a scheme that can be used to prove key usage in a hybrid encryption scheme.

Let P be a message to be sent out, consisting of the blocks $P_1, P_2, P_3, \ldots, P_n$. In a encryption hybrid scheme, a random session key is generated and used in a symmetric scheme to encrypt the message. The session key itself is encrypted using an asymmetric encryption scheme.

We present now our scheme in several steps, starting from a very unpractical setting and progressing to a more practical scheme.

4.1 First Version

Let the message blocks be split in shares A_i, B_i with $P_i = A_i \oplus B_i$. The strings A and B are composed by concatenating the blocks A_i, respectively B_i. At the end of each string, a *redundancy block* is added, e.g.

$$A_{n+1} = \text{SHA-1}(A_1 A_2 \ldots A_n), \tag{1}$$

$$B_{n+1} = \text{SHA-1}(B_1 B_2 \ldots B_n). \tag{2}$$

Two random symmetric keys K_A, K_B are generated. Each key is used to encrypt one string, producing the ciphertext strings D and E.

$$D = \text{enc}_{K_A}(A) \tag{3}$$

$$E = \text{enc}_{K_B}(B) \tag{4}$$

The encrypted message consists of the two strings D, E plus the two session keys K_A, K_B encrypted separately under the recipient's public key.

When the message passes by the PKI server, it first checks whether the encrypted session keys are indeed encrypted with the correct public key. This can be done with the asymmetric schemes described in Section 3. Subsequently, the server still needs to ascertain that the symmetric keys encrypted with the public key were indeed used to encrypt the message shares. The server will do this by choosing A or B and asking the sender to reveal the corresponding key K_A or K_B. Suppose the server chooses A. Then the sender is asked to reveal K_A. The server verifies whether K_A encrypts to a value it has seen. Subsequently, the server decrypts D with K_A and checks whether the last block of the decryption indeed obeys the redundancy rule (1).

The server may ask for only one key, hence it will not learn anything about the message. Assuming that it is not feasible to produce a string D and a key K_A that will result in a plaintext with the correct redundancy by any other means than following the scheme, it follows that a sender who doesn't follow the scheme will be caught with probability at least 50%.

Note that this version of the protocol can be simplified somewhat: the PKI server can decide not to use the protocols of Section 3 and only verify the encryption of the session key it selected to be revealed. However, the public-key protocols will be useful in the next version, described below.

4.2 Second Version

The first thing we need to improve upon, is the probability to catch cheaters, which may be as low as 50% in the first version of our protocol. We can improve the probability to catch cheaters by splitting up the message P in t sub-messages, each containing a part of the blocks P_i. Each sub-message is extended with a redundancy block and its shares are encrypted under a new pair of random keys.

For each sub-message, the server decides between A and B and asks to reveal one key. The probability to catch a cheater increases now with the number of sub-messages that are cheated with: for q malformed sub-messages, the probability becomes $1 - 2^{-q}$.

The protocol can also be extended by having more than two shares for each sub-message. Using s shares, from which the server may ask to reveal any set of $s-1$, the probability to cheat and not be detected reduces to s^{-q}.

Increasing s and/or t results in having to generate and encrypt more session keys (st in total). Hence we need a scheme to derive many session keys from a few values. Let l denote the number of session keys we send over, then we want a derivation scheme that allows to derive st keys in such a way that revealing $(s-1)t$ keys will not disclose any information on the remaining t keys. Note that this demand is sufficient, but not necessary: not all selections of $(s-1)t$ keys can be revealed in practice. We can define t sets of s keys from which at most $s-1$ are revealed. However, we can easily meet the more strict demands.

Let k be the size of a session key. Let X, Y be an l-dimensional respectively st-dimensional (row) vector with coordinates in $\mathrm{GF}(2^k)$. We choose $l = (s-1)t+1$ and compute Y as $Y = X \cdot A$. Here A is a (fixed) $l \times (st)$ matrix with the property that all $l \times l$ sub-matrices are of full rank. If $2^k > st$, then matrices satisfying these constraints can be constructed from linear codes over $\mathrm{GF}(2^k)$, with length st, dimension l and minimal distance t. The constraint $2^k > st$ is easy to meet in practical situations.

Now, in order to save on the number of session keys that need to be generated, protected and transmitted, the following is done. The sender generates a random value K of length l, encrypts it with the receiver's public key and transmits it with the message. The st session keys used to encrypt the message parts are derived from $K \cdot A$. This allows to save the encryption and transmission of $t-1$ k-bit values.

5 Application in a Key Escrow System

The type of PKI server introduced in this paper, can also be used as part of an escrow system.

5.1 Principle

The escrow system consists of two independently operated servers. The first escrow server is a PKI server as described above. It is on-line, sees all communication passing by and has the capability to check whether the messages are encrypted under the proper keys. The second escrow server is off-line. It only receives the messages that need to be decrypted. The public key of the second server is published and available to all users.

Users who want to transmit a message, are obliged to encrypt it twice: once with the key for the intended recipient, and once with the public key of the second escrow server. If the users perform this double encryption dutifully, the second escrow server can decrypt messages when required. The task of the first escrow server is to verify whether the users follow the protocol.

5.2 Message Format

Users who want to transmit a message P are obliged to encrypt it in a way similar to discussed in Section 4. We describe the scheme using the version of the protocol described in Section 4.1.

Let the message P consist of blocks $P_i, i = 1, \ldots, n$. The blocks are split into shares $A_i, B_i, i = 1, \ldots, n$. Redundancy blocks aren't necessary here. Four symmetric keys are generated: K_A, K_B, L_A, L_B. The string A is encrypted with a symmetric cipher, once under key K_A to produce the ciphertext string D, and once under key L_A to produce the ciphertext string F. Similarly, B is encrypted under K_B to produce the string E, and under L_B to produce the string G.

The keys K_A, K_B are encrypted under the public key of the recipient, and the keys L_A, L_B under the the public key of the second escrow server. This allows both the recipient and the second escrow server to recover the message P.

The PKI server executes the following steps in order to determine whether the sender has produced messages of the correct format.

1. (Optional) Check whether the symmetric keys L_A, L_B have been encrypted under the public key of the second escrow server.
2. Choose A or B and ask the sender to reveal the keys K_A, L_A, respectively K_B, L_B.
3. Check whether the revealed symmetric key L_A or L_B encrypts to the value that was received with the encrypted message.
4. Use the revealed keys to decrypt D and F, respectively E and G. Check whether the decrypted values are equal. If the values are not equal, then the sender didn't follow the required format.

Senders that don't follow the required format, are caught with probability 50%. The system can be extended in a similar way as described in Section 4.2.

6 Related Work

In the asymmetric encryption setting, our proof of usage of a certain public key, is in fact a proof of knowledge of the plaintext corresponding to the ciphertext under examination. Proving knowledge of the plaintext corresponding to a submitted ciphertext, has been called *plaintext-aware encryption* before [2].

Regarding proving of public key usage for encrypting messages, without having access to the plaintext message, we know of no published papers addressing this issue.

Our protocols are related to protocols based on verifiable encryption [4] since we describe an entity (PKI server in our settings) that tries to find out some property on an encrypted message while the message is given in an encrypted form. Verifiable encryption is meant to ensure that a entity accepts the encryption of an invalid message only with negligible probability. Typically verifiable encryption are used with digital signatures representing hence a way to encrypt a message under a designated public key and subsequently prove that the resulting ciphertext indeed contains such signature. Contrarily to protocols using verifiable encryption, we are only interested in proving that a designated public key has been used to encrypt a message without having access to the plaintext and to the decryption private key.

The schemes that we developed in this paper allow also implementing a powerful non-repudiated message delivery authority. Such authority has been defined in the literature [8] as providing the sender and recipient of a message with signed, time-stamped proof of non-repudiation of origin, of receipt and of submission. The sender and recipient can use such proofs in resolving disputes occurring between them. Our schemes allow extending such delivery authority with an additional feature which is making sure that the sender and recipient exchange confidential messages so that the messages are encrypted with valid designated keys.

7 Conclusions and Further Work

We presented the idea of a PKI server that can check which key has been used to encrypt messages, without having access to the message plaintext. We described asymmetric and hybrid protocols for achieving this goal. Finally, we explained as a special application how an escrow system can be based on this PKI server.

The hybrid scheme presented in this paper is provided as a first step in a new direction. More research is required to get more certainty about the security level and to develop schemes that are more usable.

Secondly, we think it will be interesting to look at new applications for the primitive introduced in this paper. We are particulary interested in developing new non-repudiation and fair-exchange protocols.

References

1. D. Barbecaru, A. Lioy, "Towards Simplifying PKI Implementation: Client-Server based Validation of Public Key Certificates," in Proceedings of IEEE ISSPIT 2002, 2002.
2. M. Bellare and P. Rogaway, "Optimal asymmetric encryption– how to encrypt with RSA," in Advances in Cryptology, Proceedings of EUROCRYPT '94, LNCS 950, Springer-Verlag, 1995, pp. 92–111.
3. D. Boneh, X. Ding, G. Tsudik, and B. Wong, "Instanteneous revocation of security capabilities," in Proceedings of the USENIX Security Symposium 2001, 2001.
4. J. Camenish, and I. Damgaard, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in Advances in Cryptology, Proceedings of ASIACRYPT 2000, LNCS 1976, Springer-Verlag, 2000, pp. 331–345.
5. T. Coffey, P. Saidha, "Non-repudiation with mandatory proof of receipt," in ACM-CCR: Computer Rommunication Review 26.
6. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in Advances in Cryptology, Proceedings of CRYPTO '86, LNCS 263, Springer-Verlag, 1987, pp. 186–194.

7. L.C. Guillou and J.-J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," in Advances in Cryptology, Proceedings of EUROCRYPT '88, LNCS 330, Springer-Verlag, 1988, pp. 123–128.
8. A. Herzberg, "Guaranteed delivery for secure electronic commerce and payments," unpublished manuscript.
9. M. Jalali-Sohi, P. Ebinger, "Towards Efficient PKIs for Restricted Mobile Devices," in Proceedings of IASTED International Conference Communication and Computer Networs, 2002.
10. M.O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979. "Certified electronic mail," in Proceedings of Computer Security - ESORICS'96, 1996.
11. J. Zhou, D. Gollman, "Certified electronic mail," in Proceedings of Computer Security – ESORICS'96, 1996.

Public Key Encryption with Conjunctive Field Keyword Search[⋆]

Dong Jin Park, Kihyun Kim, and Pil Joong Lee[⋆⋆]

IS Lab, Dept. of EEE, POSTECH, Pohang, Kyoungbuk, Korea
{djpark,khkim}@oberon.postech.ac.kr, pjl@postech.ac.kr

Abstract. In a public key encryption, we may want to enable someone to test whether something is a keyword in a given document without leaking anything else about the document. An email gateway, for example, may be desired to test whether the email contains a keyword "urgent" so that it could route the email accordingly, without leaking any content to the gateway. This mechanism was referred as *public key encryption with keyword search* [4]. Similarly, a user may want to enable an email gateway to search keywords *conjunctively*, such as "urgent" email from "Bob" about "finance", without leaking anything else about the email. We refer to this mechanism as *public key encryption with conjunctive field keyword search*. In this paper, we define the security model of this mechanism and propose two efficient schemes whose security is proved in the random oracle model.

Keywords: Conjunctive keyword search, pairing-based cryptography

1 Introduction

Recently, Boneh, Cresenzo, Ostrovsky and Persiano proposed *public key encryption with keyword search scheme* (or searchable public key encryption) [4]. And, Park, Kim and Lee provided another efficient scheme for the mechanism [12]. In this mechanism, encrypted message has the following form:

$$[E_{A_{pub}}[M], \mathsf{PEKS}(A_{pub}, (W_1)), \ldots, \mathsf{PEKS}(A_{pub}, (W_m))]$$

where A_{pub} is a receiver's public key, M is the message, W_i's are keywords, and PEKS is an encryption algorithm with properties; the PEKS values do not reveal any information about the message, but enable searching for specific keywords.

We borrow two examples in [4] to motivate searching on data that is encrypted using a public key system. The first example is an email gateway. Suppose user Bob sends encrypted email to user Alice using Alice's public key. Both the contents and the keywords are encrypted. The goal of the searchable public key encryption is that Alice can specify a few keywords that the email gateway

[⋆] This research was supported by University IT Research Center Project, the Brain Korea 21 Project.

[⋆⋆] On leave at KT Research Center, Korea.

can search for, but learns nothing else about incoming mail. Using the searchable public key encryption, Alice can make the gateway route email with the keyword "urgent" to her pager without information leakage to the gateway. Another example is an email server which stores many emails and each email contains a small number of keywords. All these emails created by various people are encrypted using Alice's public key. Using the searchable public key encryption, Alice can enable the server to retrieve emails containing a keyword by giving a trapdoor. The server learns nothing else about the email.

However, schemes in [4, 12] are inappropriate for a conjunctive keyword search, such as finding "urgent" email form "Bob" about "finance". Two simple solutions for the conjunctive search are set intersection and meta keywords. However, neither solution is appropriate; the former enables the server to learn which documents match each individual keyword, and the latter requires exponential storage proportional to the number of keyword fields [7]. Thus Golle, Staddon and Waters proposed *secret key* encryption with conjunctive field keyword search and gave two schemes. However, their schemes are not applicable to a public key system. Thus their schemes can not be used in routing email in the gateway. And, if Alice wants to search her email in the email server using schemes in [7], before the searching, she should decrypt her emails using her private key, encrypt using her secret key again and store the encrypted email; this seems to be undesired solution and can be a big burden in some case.

Thus, we define the model of *public key encryption with conjunctive field keyword search*. And we propose two schemes which are the earliest schemes for public key encryption with conjunctive field keyword search. The proposed scheme 1 requires just one pairing operation in Test step, which is the most time critical in general. Thus, the proposed scheme 1 is suitable for searching on the stored data. The proposed scheme 2 does not employ *admissible encoding scheme*, and is easier than scheme 1 in PECK step. Both schemes are secure in our security model assuming known intractable assumptions, such as decision bilinear Diffie-Hellman assumption, decision bilinear Diffie-Hellman inversion assumption, and strong Diffie-Hellman assumption. Moreover, our schemes are more efficient than schemes in [7] when ours are used as a secret key system by keeping A_{pub} as a secret.

This paper is organized as follows: In Section 2, we present some preliminaries. In Section 3, we define security model for the conjunctive keyword search in public key system. Proposed schemes with the security proofs are presented in Section 4 and 5. This paper concludes in Section 6.

2 Preliminary

2.1 Bilinear Map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q for some large prime q. Our scheme makes use of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ between these two groups. The map satisfies the following properties:

1. Bilinear: We say that a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is bilinear if $\hat{e}(aU, bV) = \hat{e}(U, V)^{ab}$ for all $U, V \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1, \mathbb{G}_2$ are groups of prime order this implies that if P is a generator of $\mathbb{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(U, V)$ for any $U, V \in \mathbb{G}_1$.

We can make the bilinear map using the Weil pairing or the Tate pairing [1, 5, 6, 8]. In the pairings, the group $\mathbb{G}_1$ is a subgroup of the additive group of points of an elliptic curve. The group $\mathbb{G}_2$ is a subgroup of the multiplicative group of a finite field. Therefore, throughout the paper we view $\mathbb{G}_1$ as an additive group and $\mathbb{G}_2$ as a multiplicative group.

2.2 Complexity Assumptions

We review two hardness assumptions, Bilinear Diffie-Hellman (BDH) assumption and Bilinear Diffie-Hellman Inversion (BDHI) assumption, on which the security proofs of our proposed schemes are based.

Bilinear Diffie-Hellman Assumption [5, 8]
The BDH problem in $\mathbb{G}_1$ is as follows: given a tuple $P, \alpha P, \beta P, \gamma P \in \mathbb{G}_1$ as input, output $\hat{e}(P, P)^{\alpha\beta\gamma} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH in $\mathbb{G}_1$ if

$$Pr\left[\mathcal{A}(P, \alpha P, \beta P, \gamma P) = \hat{e}(P, P)^{\alpha\beta\gamma}\right] \geq \epsilon$$

where the probability is over the random choice of $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$ and random bits used by $\mathcal{A}$. Similarly, we say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the *decision* BDH problem in $\mathbb{G}_1$ if

$$\left|Pr\left[\mathcal{B}(P, \alpha P, \beta P, \gamma P, \hat{e}(P, P)^{\alpha\beta\gamma}) = 0\right] - Pr\left[\mathcal{B}(P, \alpha P, \beta P, \gamma P, R) = 0\right]\right| \geq \epsilon$$

where the probability is over the random choice of $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$, the random choice of $R \in \mathbb{G}_2^*$, and the random bits of $\mathcal{B}$.

Definition 1. *We say that the (decision) (t, ϵ)-BDH assumption holds in $\mathbb{G}_1$ if no t-time algorithm has advantage at least ϵ in solving the (decision) BDH problem in $\mathbb{G}_1$.*

Bilinear Diffie-Hellman Inversion Assumption [2, 11]
The q-BDHI problem is defined as follows: given the $(q+1)$-tuple $(P, xP, x^2P, \ldots, \ldots, x^qP) \in (\mathbb{G}_1^*)^{q+1}$ as input, compute $\hat{e}(P, P)^{1/x} \in \mathbb{G}_2^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-BDHI in $\mathbb{G}_1$ if

$$Pr\left[\mathcal{A}(P, xP, x^2P, \ldots, x^qP) = \hat{e}(P, P)^{1/x}\right] \geq \epsilon$$

where the probability is over the random choice of $x \in \mathbb{Z}_p^*$ and random bits used by $\mathcal{A}$. Similarly, we say that an algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving the *decision* q-BDHI problem in $\mathbb{G}_1$ if

$$\left| Pr\big[\mathcal{B}(P, xP, \ldots, x^q P, \hat{e}(P,P)^{1/x}) = 0\big] - Pr\big[\mathcal{B}(P, xP, \ldots, x^q P, R) = 0\big] \right| \geq \epsilon$$

where the probability is over the random choice of $x \in \mathbb{Z}_p^*$, the random choice of $R \in \mathbb{G}_2^*$, and random bits used by $\mathcal{B}$.

Definition 2. *We say that the (decision)* (t, q, ϵ)*-BDHI assumption holds in* $\mathbb{G}_1$ *if no* t*-time algorithm has advantage at least* ϵ *in solving the (decision)* q*-BDHI problem in* $\mathbb{G}_1$*.*

A similar assumptions are employed in [3, 11, 14].

3 Conjunctive Field Keyword Search Scheme

3.1 Model

Sender sends the following message:

$$[E_{A_{pub}}[M], \mathsf{PECK}(A_{pub}, (W_1, W_2, \ldots, W_m))]$$

where A_{pub} is a receiver's public key, M is the email body, m is the number of keyword fields and PECK is an algorithm with properties discussed below. The PECK values do not reveal any information about the message, but enable searching for specific keywords.

Here, we employ the same assumptions as in [7]:

1. The same keyword never appears in two different keyword fields in the same document.
2. Every keyword field is defined for every document.

These requirements can be easily satisfied [7]. If documents were emails, for example, we could define four fields, such as "From", "To", "Date" and "Subject". The first requirement is satisfied by prepending keywords with the name of field they belong to. The second requirement is by assigning the keyword "{THE NAME OF A FIELD}:NULL" to the field that does not have a valid keyword.

We identify a document with the vector of m keywords. Thus, the i-th document D_i becomes $(W_{i,1}, W_{i,2}, \ldots, W_{i,m})$. To simplify the description, we ignore $E_{A_{pub}}[M]$ that can be encrypted with any secure public key encryption.

To search keywords conjunctively, a format of query is defined as $Q = (I_1, I_2, \ldots, I_t, \Omega_1, \Omega_2, \ldots, \Omega_t)$, where I_i's are values, between 1 and m, of position of a keyword in the keyword fields, Ω_i's are keywords to search and t is the number of keywords in Q. The corresponding trapdoor becomes T_Q that can test whether a document has keywords in their keyword fields.

We call such a system *non-interactive public key encryption with conjunctive field keyword search (PECKS).*

Definition 3. *A non-interactive public key encryption with conjunctive field keyword search scheme consists of the following polynomial time randomized algorithms:*

1. *$\mathsf{KeyGen}(1^k)$: Takes a security parameter, 1^k, and generates a public/private key pair A_{pub}, A_{priv}.*
2. *$\mathsf{PECK}(A_{pub}, D)$: For a public key A_{pub} and a document D, produces a conjunctive searchable encryption of D.*
3. *$\mathsf{Trapdoor}(A_{priv}, Q)$: Given a private key A_{priv} and a query Q, produces a trapdoor T_Q.*
4. *$\mathsf{Test}(A_{pub}, S, T_Q)$: Given a public key A_{pub}, a conjunctive searchable encryption $S = \mathsf{PECK}(A_{pub}, D)$, and a trapdoor $T_Q = \mathsf{Trapdoor}(A_{priv}, Q)$, outputs 'yes' if $\{(W_{I_1} = \Omega_1)$, $(W_{I_2} = \Omega_2)$, $\ldots$, and $(W_{I_t} = \Omega_t)\}$ and 'no' otherwise.*

3.2 Security Definition

The original ICC (indistinguishability of ciphertext from ciphertext) is a security game defined in [7] for a secret key system. Since any user can make PECK in a public key system, we remove steps for encryption oracle queries. In this paper, ICC works as follows:

1. The challenger runs the $\mathsf{KeyGen}(1^k)$ algorithm to generate A_{pub} and A_{priv}. It gives A_{pub} to the attacker.
2. The attacker can adaptively ask the challenger for the trapdoor T_Q for any query Q of his choice.
3. At some point, the attacker $\mathcal{A}$ sends the challenger two documents D_0, D_1. The only restriction is that none of trapdoors asked previously in step 2 is distinguishing for D_0 and D_1. The challenger picks a random $b \in \{0, 1\}$ and gives the attacker $C = \mathsf{PECK}(A_{pub}, D_b)$. We refer C as the challenge PECK .
4. The attacker can continue to ask for trapdoor T_Q for any query Q of his choice unless T_Q can distinguish D_0 for D_1.
5. Eventually, the attacker $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$.

In other words, the attacker wins the game if he can correctly guess whether he was given the PECK for D_0 or D_1. We define $\mathcal{A}$'s advantage as

$$Adv_{\mathcal{A}}(1^k) = \left| Pr[b = b'] - \frac{1}{2} \right| .$$

Definition 4. *We say that a PECKS is semantically secure against an adaptive chosen keyword attack if for any polynomial time attacker $\mathcal{A}$ according to the game ICC we have that $Adv_{\mathcal{A}}(1^k)$ is a negligible function.*

To simplify our proofs, we define a variant of ICC, that is, ILCR (indistinguishability of limited ciphertext from random)[1]:

[1] A similar security game ICLR (indistinguishability of ciphertext from limited random) is defined in [7]. Our security game seems to be easier to understand and to apply to security proof than ICLR.

1. The challenger runs the $\mathsf{KeyGen}(1^k)$ algorithm to generate A_{pub} and A_{priv}. It gives A_{pub} to the attacker.
2. The attacker can adaptively ask the challenger for the trapdoor T_Q for any query Q of his choice.
3. At some point, the attacker $\mathcal{A}$ sends the challenger a keyword W and its position z in $[1, m]$. The only restriction is that the attacker did not previously ask for the trapdoor $T_{(z,W)}$. Then the challenger generates two documents D_0 and D_1. D_0 is a random document with restriction that z-th word of D_0 is W and D_1 is a random document in all positions. The challenger should select the two documents so that the previous trapdoors cannot distinguish D_0 for D_1. The challenger picks a random $b \in \{0, 1\}$ and gives the attacker $C = \mathsf{PECK}(A_{pub}, D_b)$ with two documents D_0 and D_1. We refer C as the challenge PECK .
4. The attacker can continue to ask for trapdoor T_Q for any query Q of his choice unless T_Q can distinguish D_0 for D_1.
5. Finally, the attacker $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$.

Theorem 1. *If there exists an adversary $\mathcal{A}$ that wins game ICC with advantage ϵ, then there exists an adversary $\mathcal{B}$ that wins game ILCR with advantage $\epsilon/2m^2$.*

Proof. We can prove this theorem by the standard hybrid argument. □

4 Proposed Scheme 1

4.1 Construction

The proposed scheme 1 requires just one pairing operation in Test step that is the most time critical step in general. Thus, the proposed scheme 1 is suitable for both searching on the stored data and routing in the email gateway. Observe that the scheme using a bilinear map in [4] also requires one pairing for a single keyword search. We need hash function $H : \{0, 1\}^* \to \mathbb{G}_1$. The proposed scheme 1 works as follows:

– $\mathsf{KeyGen}(1^k)$: The input security parameter determines the size, p, of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The algorithm chooses two random numbers $s_1, s_2 \in \mathbb{Z}_p$ and a generator P of $\mathbb{G}_1$. It outputs $A_{pub} = [P, Y_1 = s_1 P, Y_2 = s_2 P]$ and $A_{priv} = [s_1, s_2]$.

– $\mathsf{PECK}(A_{pub}, D)$: Select a random number $r \in \mathbb{Z}_p$ and obtain hash values. The $\mathsf{PECK}(A_{pub}, D)$ is

$$\big[\hat{e}(rH(W_1), Y_1), \hat{e}(rH(W_2), Y_1), \ldots, \hat{e}(rH(W_m), Y_1), rY_2, rP\big].$$

– $\mathsf{Trapdoor}(A_{priv}, Q)$: Select a random $u \in \mathbb{Z}_p$ and make $T_Q = [T_1, T_2, I_1, I_2, \ldots, \ldots, I_t]$ where

$$T_1 = \left(\frac{s_1}{s_2 + u} \bmod p\right)\Big(H(\Omega_1) + H(\Omega_2) + \ldots + H(\Omega_t)\Big),$$
$$T_2 = u,$$

and $I_1, I_2, \ldots, I_t$ are positions of the words come from Q.

– Test(A_{pub}, S, T_Q): Let $S = [A_1, A_2, \ldots, A_m, B, C]$. Check the equality,

$$A_{I_1} \times A_{I_2} \times \ldots \times A_{I_t} = \hat{e}(T_1, B + T_2C).$$

If so, output 'yes'; if not, output 'no'.

The equality of Test holds if $W_{I_i} = \Omega_i$ for $1 \leq i \leq t$. We can check as follows:

$$\begin{aligned} & A_{I_1} \times A_{I_2} \times \ldots \times A_{I_t} \\ & = \hat{e}(r(H(W_{I_1}) + H(W_{I_2}) + \ldots + H(W_{I_t})), Y_1). \\ & = \hat{e}(s_1(H(\Omega_1) + H(\Omega_2) + \ldots + H(\Omega_t)), rP). \\ & = \hat{e}(T_1, B + T_2C). \end{aligned}$$

4.2 Security Proof

Theorem 2. *The proposed scheme 1 is secure according to the game ILCR assuming DBDH is intractable.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the proposed scheme 1 under the security game ILCR. Suppose $\mathcal{A}$ makes at most q_T Trapdoor queries. We build an adversary $\mathcal{B}$ that solves the DBDH problem in $\mathbb{G}_1$ with probability at least $\epsilon' = \epsilon/(e^m q_T)$, where e is the base of the natural logarithm. The running time of $\mathcal{B}$ is approximately the same as $\mathcal{A}$'s.

On input $(P, \alpha P, \beta P, \gamma P, R)$ adversary $\mathcal{B}$'s goal is to output 1 if $R = \hat{e}(P, P)^{\alpha\beta\gamma}$ and 0 otherwise. Adversary $\mathcal{B}$ works by interacting with $\mathcal{A}$ in the ILCR game as follows:

KeyGen. Adversary $\mathcal{B}$ selects a random number $s \in \mathbb{Z}_p$, and gives $\mathcal{A}$ the public key $A_{pub} = [P, Y_1 = \alpha P, Y_2 = sP]$. The corresponding private key A_{priv} is $[\alpha, s]$ where the value α is unknown to $\mathcal{B}$.

Hash queries. At any time adversary $\mathcal{A}$ can query the random oracle H. To respond to H queries, $\mathcal{B}$ maintains a list of tuples $< W_i, h_i, a_i, c_i >$ called the H-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H at a point $W_i \in \{0,1\}^*$, Adversary $\mathcal{B}$ responds as follows:

1. If the query W_i already appears in the H-list in a tuple $< W_i, h_i, a_i, c_i >$, then adversary $\mathcal{B}$ responds with $H(W_i) = h_i$.
2. Otherwise, $\mathcal{B}$ generates a random coin $c_i \in \{0,1\}$ so that $Pr[c_i = 0] = 1/q_T$.
3. $\mathcal{B}$ selects a random $a_i \in \mathbb{Z}_p$. If $c_i = 0$, $\mathcal{B}$ makes $h_i \leftarrow a_i(\beta P)$. Otherwise, $\mathcal{B}$ computes $h_i \leftarrow a_i P$.
4. $\mathcal{B}$ adds the tuple $< W_i, h_i, a_i, c_i >$ to the H-list and responds with $H(W_i) = h_i$.

Trapdoor queries. When $\mathcal{A}$ issues a query for the trapdoor corresponding the the query $Q_i = (I_{i,1}, I_{i,2}, \ldots, I_{i,t}, \Omega_{i,1}, \Omega_{i,2}, \ldots, \Omega_{i,t})$, $\mathcal{A}$ responds as follows:

1. Adversary $\mathcal{B}$ executes the above algorithm for responding H queries to obtain an $h_{i,j}$'s such than $H(\Omega_{i,j}) = h_{i,j}$. Let $< \Omega_{i,j}, h_{i,j}, a_{i,j}, c_{i,j} >$ be the corresponding tuple on the H-list. Unless all $c_{i,j}$'s are 1, then $\mathcal{B}$ fails.

2. Otherwise, we know $c_{i,j} = 1$ and hence $h_{i,j} = a_{i,j}P$. Define $E_i = (a_{i,1} + a_{i,2} + \ldots + a_{i,t})Y_1$ and $F_i = (\frac{1}{s+u_i} \bmod p)E_i$ where u_i is a random integer in $\mathbb{Z}_p$. Observe that $F_i = \frac{\alpha}{s+u_i}(H(\Omega_{i,1})+H(\Omega_{i,2})+\ldots+H(\Omega_{i,t}))$. Therefore F_i is the correct trapdoor for the query Q under the public key $A_{pub} = [P, \alpha P, sP]$. Adversary $\mathcal{B}$ gives $[F_i, u_i, I_{i,1}, I_{i,2}, \ldots, I_{i,t}]$ to adversary $\mathcal{A}$.

Challenge. Eventually adversary $\mathcal{A}$ produces a keyword W and a position z that it wishes to be challenged on. $\mathcal{B}$ generates the challenge PECK as follows:

1. Adversary $\mathcal{B}$ executes the above algorithm for responding H queries to obtain an h such than $H(W) = h$. Let $< W, h, a, c >$ be the corresponding tuple on the H-list. If $c = 1$, then $\mathcal{B}$ fails.
2. Adversary $\mathcal{B}$ selects random words $W_{i,j}$'s for $i \in \{0,1\}$ and $1 \le j \le m$ except $W_{0,z}$ that becomes W. And $\mathcal{B}$ generates $D_i = (W_{i,1}, \ldots, W_{i,m})$ for $i \in \{0,1\}$. The restriction is that the previous trapdoors cannot distinguish D_0 for D_1. If so, $\mathcal{B}$ regenerates D_0 and D_1. Given $W_{0,j}$, let $< W_{0,j}, h_{0,j}, a_{0,j}, c_{0,j} >$ be the corresponding tuple on the H-list.
3. Adversary $\mathcal{B}$ responds with the challenge $[A_1, A_2, \ldots, A_m, B, C]$ and two documents D_0 and D_1. The values of challenge are computed as follows:
   – If $c_{0,j} = 0$, $A_j = R^{a_{0,j}}$. Otherwise, $A_j = \hat{e}(a_{0,j}(\gamma P), \alpha P)$.
   – $B = s(\gamma P)$.
   – $C = \gamma P$.

If R is $\hat{e}(P,P)^{\alpha\beta\gamma}$, the challenge is equivalent to

$$\big[\hat{e}(\gamma H(W_{0,1}), Y_1), \hat{e}(\gamma H(W_{0,2}), Y_1), \ldots, \hat{e}(\gamma H(W_{0,m}), Y_1), \gamma Y_2, \gamma P\big].$$

Thus this is a valid PECK for D_0 as required.

More queries. Adversary $\mathcal{B}$ responds to these queries as before. The only restriction is that no Trapdoor query distinguishes D_0 for D_1.

Output. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ indicating whether the challenge is the encryption of D_0 or D_1. Adversary concludes its own game by outputting a guess; if $b' = 0$ then $\mathcal{B}$ outputs 'yes' meaning $R = \hat{e}(P,P)^{\alpha\beta\gamma}$. Otherwise, it outputs 'no'.

This completes the description of adversary $\mathcal{B}$. Adversary $\mathcal{B}$ can fails in responding Trapdoor queries and preparing the challenge. We define two events:

– $\mathcal{E}_1$: $\mathcal{B}$ does not abort as result of any $\mathcal{A}$'s Trapdoor queries.
– $\mathcal{E}_2$: $\mathcal{B}$ does not abort during preparing the challenge for $\mathcal{A}$.

We can assume that q_T is sufficiently large, thus $(1 - 1/q_T)^{q_T} = 1/e$. $Pr[\mathcal{E}_1] \ge 1/e^m$ and $Pr[\mathcal{E}_2] = 1/q_T$. Thus, $\mathcal{B}$ breaks DBDH problem with the advantage $\epsilon' \ge \epsilon/(e^m q_T)$. □

Corollary 1. *The proposed scheme 1 is semantically secure against an adaptive chosen keyword attack assuming DBDH is intractable.*

4.3 Efficient Implementation

The proposed scheme 1 requires m pairings in PECK step. We show that the pairing operations are not a burden, by employing techniques of compressed pairing [13], preprocessing of fixed point [1], and combining final exponentiations.

Firstly, we review pairing operation very shortly. To simplify description, we consider the case of embedding degree 2, although our description can cover all even embedding degrees. A pairing operation consists of two parts: Miller's algorithm [10] maps $E(\mathbb{F}_q) \times E(\mathbb{F}_q)$ to $\mathbb{F}_{q^2}$ and a final exponentiation algorithm maps the field element to an element in $\mathbb{G}_2$. Let $x + iy \in \mathbb{F}_{q^2}$ is the output of Miller's algorithm where $x, y \in \mathbb{F}_q$, and $i^2 = \delta$ for some quadratic non-residue $\delta \in \mathbb{F}_q$. The final exponentiation raises the output of the Miller's algorithm to the power of $(q^2 - 1)/p = (q - 1)(q + 1)/p$. Then

$$(x + iy)^{q-1} = (x + iy)^q/(x + iy) = (x - iy)/(x + iy)$$

which is a simple computation. We define a (modified) pseudo-compressed pairing $\hat{\varepsilon}(U, V)$ as $a + bi = (x - iy)/(x + iy)$. Observe that $\hat{e}(U, V) = \hat{\varepsilon}(U, V)^{(q+1)/p}$. Given a, we can compute b up to a sign; the value is $\pm((a^2 - 1)^{1/2})/i$. Thus, if $s(a, b)$ is a single bit determines the sign of b, $a||s(a, b)$ is a sufficient information to reconstruct $a + bi$.

Now, the efficient implementation of the proposed scheme 1 is as follows:

- $\mathsf{KeyGen}(1^k)$: The same as the step in 4.1.
- $\mathsf{PECK}(A_{pub}, D)$: Select a random number $r \in \mathbb{Z}_p$ and obtain hash values. The $\mathsf{PECK}(A_{pub}, D)$ is

$$\big[a_1||s(a_1, b_1), a_2||s(a_2, b_2), \ldots, a_m||s(a_m, b_m), rY_2, rP\big],$$

  where a_i and b_i are computed from $\hat{\varepsilon}(rH(W_i), Y_1)$.
  Since all pairings use the same point Y_1, Miller's algorithm can be sped up by reusing the coefficients of line functions [1]. In addition, the pseudo-compressed pairing enable us to remove the computational cost of the final exponentiations and to reduce the size of PECK approximately by half.
- $\mathsf{Trapdoor}(A_{priv}, Q)$: The same as the step in 4.1.
- $\mathsf{Test}(A_{pub}, S, T_Q)$: Let $S = [A_1, A_2, \ldots, A_m, B, C]$. Recompute $a_i + b_i i$'s from A_i's, and $a_T + b_T i = \hat{\varepsilon}(T_1, B + T_2C)$. Check the equality,

$$\left(\frac{(a_{I_1} + b_{I_1} i) \times (a_{I_2} + b_{I_2} i) \times \ldots \times (a_{I_t} + b_{I_t} i)}{(a_T + b_T i)}\right)^{(q+1)/p} = 1.$$

  If so, output 'yes'; if not, output 'no'.
  Observe that this step requires only one final exponentiation, which is also required even if the pseudo-compressed pairing is not used. That is, there is no computational overhead in this step due to the pseudo-compressed pairing.

Using the above techniques, we can reduce a burden of pairing operations in PECK step without additional cost in other steps.

5 Proposed Scheme 2

5.1 Construction

The proposed scheme 1 is efficient, but employs *admissible encoding scheme* [5] in the H function. In a future, the admissible encoding scheme can be a big computational burden, because the encoding scheme requires $\log_2(q/p)$-bit scalar multiplication in $E(\mathbb{F}_q)$ where $\mathbb{F}_q$ is the field on which $\mathbb{G}_1$ is based and p is the size of group $\mathbb{G}_1, \mathbb{G}_2$. For example, if $\log_2 p$ is 512 and the embedding degree of pairing is 6, then $\log_2 q$ should be at least 2560 and thus $\log_2(q/p)$ is 2048 [5, 9]. This means that we should compute 2048-bit scalar multiplication to compute admissible encoding scheme, which is much harder than scalar multiplication in $\mathbb{G}_1$.

The proposed scheme 2 does not employ the admissible encoding scheme. And, this scheme is easier than scheme 1 in PECK step. We need hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{\log_2 p}$ and $H_2 : \{0,1\}^* \to \{0,1\}^{\log_2 p}$. The proposed scheme 2 works as follows:

– $\mathsf{KeyGen}(1^k)$: The input security parameter determines the size, p, of the group $\mathbb{G}_1$ and $\mathbb{G}_2$. The algorithm chooses random numbers $s_1, s_2, \ldots, s_m, s_{m+1}, s_{m+2} \in \mathbb{Z}_p$ and a generator P of $\mathbb{G}_1$. It outputs $A_{pub} = [P,\ Y_1 = s_1P, \ldots, Y_m = s_mP,\ Y_{m+1} = s_{m+1}P, Y_{m+2} = s_{m+2}P,\ g = \hat{e}(P,P)]$ and $A_{priv} = [s_1, \ldots, s_m,\ s_{m+1}, s_{m+2}]$.

– $\mathsf{PECK}(A_{pub}, D)$: Select random $r_0, r_1, \ldots, r_m \in \mathbb{Z}_p$ and obtain hash values. the $\mathsf{PECK}(A_{pub}, D)$ is

$$\big[r_0(Y_1 + H_1(W_1)P) + r_1P, \ldots, r_0(Y_m + H_1(W_m)P) + r_mP,\\ r_1Y_{m+1}, \ldots, r_mY_{m+1}, r_0Y_{m+2}, H_2(g^{r_0})\big].$$

– $\mathsf{Trapdoor}(A_{priv},Q)$: Select a random $u \in \mathbb{Z}_p$ and make $T_Q = [T_1,T_2,T_3,I_1,\ldots,\ldots,I_t]$ where

$$T_1 = \frac{1}{s_{I_1} + \ldots + s_{I_t} + H_1(\Omega_1) + \ldots + H_1(\Omega_t) + s_{m+2}u}P,$$
$$T_2 = \frac{1}{s_{m+1}}T_1,$$
$$T_3 = u$$

and $I_1, \ldots, I_t$ are positions of the words come from Q.

– $\mathsf{Test}(A_{pub}, S, T_Q)$: Let $S = [A_1, \ldots, A_m, B_1, \ldots, B_m, C, D]$. Check the equality,

$$H_2\left(\frac{\hat{e}(A_{I_1} + \ldots + A_{I_t} + T_3C, T_1)}{\hat{e}(B_{I_1} + \ldots + B_{I_t}, T_2)}\right) = D.$$

If so, output ‘yes’; if not, output ‘no’.

The equality of Test holds if $W_{I_i} = \Omega_i$ for $1 \leq i \leq t$. We can check as follows:

$$\begin{aligned}
&H_2\left(\frac{\hat{e}(A_{I_1} + \ldots + A_{I_t} + T_3C, T_1)}{\hat{e}(B_{I_1} + \ldots + B_{I_t}, T_2)}\right)\\
&= H_2\left(\frac{\hat{e}(A_{I_1} + \ldots + A_{I_t} + T_3C, T_1)}{\hat{e}(r_{I_1}P + \ldots + r_{I_t}P, T_1)}\right)\\
&= H_2\left(\hat{e}(r_0(Y_{I_1} + H_1(W_{I_1})P) + \ldots + r_0(Y_{I_t} + H_1(W_{I_t})P) + T_3C, T_1\right)\\
&= H_2\left(\hat{e}(r_0P, P)\right)\\
&= D.
\end{aligned}$$

5.2 Security Proof

Theorem 3. *The proposed scheme 2 is secure according to the game ILCR assuming DBDHI is intractable.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the proposed scheme 2 under the security game ILCR. Suppose $\mathcal{A}$ makes at most q_T Trapdoor queries. We build an adversary $\mathcal{B}$ that solves the $(q_T + 1)$-DBDHI problem in $\mathbb{G}_1$ with probability at least $\epsilon' = \epsilon/(emq_T)$. The running time of adversary $\mathcal{B}$ is approximately the same as $\mathcal{A}$'s.

On input $(P, xP, x^2P, \ldots, x^{q_T+1}P, R)$ adversary $\mathcal{B}$'s goal is to output 1 if $R = \hat{e}(P,P)^{1/x}$ and 0 otherwise. Adversary $\mathcal{B}$ works by interacting with $\mathcal{A}$ in the ILCR game as follows:

KeyGen. Adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ selects $\delta_1, \delta_2, \ldots, \delta_{q_T} \in \mathbb{Z}_p^*$ at random and let $f(z) = \prod_{j=1}^{q_T}(z + \delta_j)$.
2. Expand the terms of f to get $f(z) = \sum_{i=0}^{q_T} c_i z^i$. Compute $U = f(x)P = \sum_{i=0}^{q_T} c_i x^i P$ and $V = xU = \sum_{i=1}^{q_T+1} c_{i-1} x^i P$.
3. $\mathcal{B}$ computes $\frac{1}{x+\delta_i}U = (f(x)/(x+\delta_i))P = \sum_{j=0}^{q_T-1} d_j x^j P$ for $1 \leq i \leq q_T$, and stores the pairs $(\delta_i, \frac{1}{x+\delta_i}U)$'s.
4. $\mathcal{B}$ computes
$$R_U = R^{c_0^2} \times \hat{e}((c_0 + \sum_{i=0}^{q_T} c_i x^i)P, \sum_{i=0}^{q_T-1} c_{i+1} x^i P).$$
Observe that if $R = \hat{e}(P,P)^{1/x}$ then $R_U = \hat{e}(U,U)^{1/x}$.
5. $\mathcal{B}$ selects $\alpha_1, \ldots, \alpha_{m+2}, \beta_1, \ldots, \beta_m \in \mathbb{Z}_p$ at random, and computes $Y_i = \alpha_i V - \beta_i U$ for $1 \leq i \leq m$, $Y_{m+1} = \alpha_{m+1}U$, and $Y_{m+2} = \alpha_{m+2}V$. At last, $\mathcal{B}$ gives $\mathcal{A}$ the public key $A_{pub} = [U, Y_1, \ldots, Y_m, Y_{m+1}, Y_{m+2}, g = \hat{e}(U,U)]$. The corresponding private key A_{priv} is $[s_1 = \alpha_1 x - \beta_1, \ldots, s_m = \alpha_m x - \beta_m, s_{m+1} = \alpha_{m+1}, s_{m+2} = \alpha_{m+2}x]$ where the value $s_i, \ldots, s_m$ and s_{m+2} are unknown to $\mathcal{B}$.

***H_1* queries.** To respond to H_1 queries, $\mathcal{B}$ maintains a list of tuples $< W_i, h_i, c_i >$ called the H_1-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_1 at a point $W_i \in \{0,1\}^*$, Adversary $\mathcal{B}$ responds as follows:

1. If the query W_i already appears in the H_1-list in a tuple $< W_i, h_i, c_i >$, then adversary $\mathcal{B}$ responds with $H_1(W_i) = h_i$.
2. Otherwise, $\mathcal{B}$ generates a random $c_i \in [1, \ldots, mq_T]$ so that $Pr[c_i \leq m] = 1/q_T$.
3. If $c_i > m$, $\mathcal{B}$ selects a random $h_i \in \{0,1\}^{\log_2 p}$. Otherwise, $\mathcal{B}$ makes $h_i \leftarrow \beta_{c_i}$.
4. $\mathcal{B}$ adds the tuple $< W_i$,h_i,$c_i >$ to the H_1-list and responds with $H_1(W_i)$= h_i.

***H_2* queries.** To respond to H_2 queries, $\mathcal{B}$ maintains a list of tuples $< g_i, \gamma_i >$ called the H_2-list. The list is initially empty. When $\mathcal{A}$ queries the random oracle H_2 at a point $g_i \in \{0,1\}^*$, Adversary $\mathcal{B}$ responds as follows:

1. If the query g_i already appears in the H_2-list in a tuple $< g_i, \gamma_i >$, then adversary $\mathcal{B}$ responds with $H_2(g_i) = \gamma_i$.
2. $\mathcal{B}$ selects a random $\gamma_i \in \{0,1\}^{\log_2 p}$, and adds the tuple $< g_i, \gamma_i >$ to the H_2-list. Adversary $\mathcal{B}$ responds with $H_2(g_i) = \gamma_i$.

Trapdoor queries. When $\mathcal{A}$ issues a query for the trapdoor corresponding the the query $Q_i = (I_{i,1}, I_{i,2}, \ldots, I_{i,t}, \Omega_{i,1}, \Omega_{i,2}, \ldots, \Omega_{i,t})$, $\mathcal{A}$ responds as follows:

1. Adversary $\mathcal{B}$ executes the above algorithm for responding H_1 queries to obtain an $h_{i,j}$'s such than $H_1(\Omega_{i,j}) = h_{i,j}$. Let $< \Omega_{i,j}, h_{i,j}, c_{i,j} >$ be the corresponding tuple on the H_1-list. If $\forall j, c_{i,j} \leq m$, then $\mathcal{B}$ fails.
2. Otherwise, $\mathcal{B}$ obtains $h_{i,j}$'s. Define $E_i = s_{I_1} + \ldots + s_{I_t} + h_{i,1} + \ldots + h_{i,t}$, which can be expressed as $(\alpha_{I_1} + \ldots + \alpha_{I_t})x - (\beta_{I_1} + \ldots + \beta_{I_t}) + (h_{i,1} + \ldots + h_{i,t}) = \Gamma_i x + \Delta_i$. The Γ_i and Δ_i can be computed from the known values, α_{I_j}'s, β_{I_j}'s and $h_{i,j}$'s.
3. $\mathcal{B}$ picks i-th pair $(\delta_i, \frac{1}{x+\delta_i}U)$ in a storage. Find u_i, v_i satisfying an equality of $1/(x + \delta_i) = (v_i)/(\Gamma_i x + \Delta_i + \alpha_{m+2} x u_i)$. The u_i and v_i become $((\Delta_i/\delta_i - \Gamma_i)/\alpha_{m+2})$ and Δ_i/δ_i, respectively. Define $F_i = 1/(v_i(x + \delta_i))$. Observe that $F_i = 1/(\Gamma_i x + \Delta_i + \alpha_{m+2} x u_i) = 1/(s_{I_1} + \ldots + s_{I_t} + h_{i,1} + \ldots + h_{i,t} + s_{m+2} u_i)$. Therefore $(F_i, \frac{1}{s_{m+1}} F_i, u_i)$ is the correct trapdoor for the query Q. Adversary $\mathcal{B}$ gives $[F_i, \frac{1}{s_{m+1}} F_i, u_i, I_{i,1}, \ldots, I_{i,t}]$ to adversary $\mathcal{A}$.

Challenge. Eventually adversary $\mathcal{A}$ produces a keyword W and a position z that it wishes to be challenged on. $\mathcal{B}$ generates the challenge PECK as follows:

1. Adversary $\mathcal{B}$ executes the above algorithm for responding H_1 queries to obtain an h such than $H_1(W) = h$. Let $< W, h, c >$ be the corresponding tuple on the H_1-list. If $c \neq z$, then $\mathcal{B}$ fails.
2. Adversary $\mathcal{B}$ selects random words $W_{0,j}$'s for $1 \leq j \leq z - 1$ and $z + 1 \leq j \leq m$ so that $H_1(W_{0,j})$ becomes β_j by fabricating the H_1-list. Note that $W_{0,z}$ is equivalent to W. $\mathcal{B}$ selects random words $W_{1,j}$'s for $1 \leq j \leq m$. And $\mathcal{B}$ generates $D_i = (W_{i,1}, \ldots, W_{i,m})$ for $i \in \{0,1\}$. The restriction is that the previous trapdoors cannot distinguish D_0 for D_1. If so, $\mathcal{B}$ regenerates D_0 and D_1.
3. Adversary $\mathcal{B}$ select random $\rho, r_1, \ldots, r_m \in \mathbb{Z}_p$. Compute $A_i = \rho\alpha_i U + r_i U$ and $B_i = r_i Y_{m+1}$ for $1 \leq i \leq m$, $C = \rho a_{m+2} U$, and $D = H_2(R_U^{\rho})$.

$\mathcal{B}$ responds with the challenge $[A_1, \ldots, A_m, B_1, \ldots, B_m, C, D]$ and two documents D_0 and D_1. Observe that r_0 is equivalent to ρ/x.

If R is $\hat{e}(P, P)^{1/x}$, the challenge is equivalent to

$$\big[(\rho/x)(Y_1 + H_1(W_1)U) + r_1 U, \ldots, (\rho/x)(Y_m + H_1(W_m)U) + r_m U,$$
$$r_1 Y_{m+1}, \ldots, r_m Y_{m+1}, (\rho/x)Y_{m+2}, H_2(g^{(\rho/x)})\big].$$

Thus this is a valid PECK for D_0 as required.

More queries. Adversary $\mathcal{B}$ responds to these queries as before. The only restriction is that no Trapdoor query distinguishes D_0 for D_1.

Output. Finally, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ indicating whether the challenge is the encryption of D_0 or D_1. Adversary concludes its own game by outputting a guess; if $b' = 0$ then $\mathcal{B}$ outputs 'yes' meaning $R = \hat{e}(P, P)^{1/x}$. Otherwise, it outputs 'no'.

This completes the description of adversary $\mathcal{B}$. Adversary $\mathcal{B}$ can fails in responding Trapdoor queries and preparing the challenge. We define two events:

- $\mathcal{E}_1$: $\mathcal{B}$ does not abort as result of any $\mathcal{A}$'s Trapdoor queries.
- $\mathcal{E}_2$: $\mathcal{B}$ does not abort during preparing the challenge for $\mathcal{A}$.

We can assume that q_T is sufficiently large, thus $(1 - 1/q_T)^{q_T} = 1/e$. $Pr[\mathcal{E}_1] = \prod_{i=1}^{q_T}(1 - 1/(q_T)^{t_i}) \geq \prod_{i=1}^{q_T}(1 - 1/q_T) = 1/e$ and $Pr[\mathcal{E}_2] = 1/mq_T$. Thus, $\mathcal{B}$ breaks $(q_T + 1)$-DBDHI problem with the advantage $\epsilon' \geq \epsilon/(emq_T)$. □

Corollary 2. *The proposed scheme 2 is semantically secure against an adaptive chosen keyword attack assuming DBDHI are intractable.*

6 Conclusion

We refined the security model of a public key encryption with conjunctive field keyword search and gave two efficient schemes using a bilinear map. The proposed scheme 1 requires just one pairing operation in Test that is the most time critical step. Thus, the proposed scheme 1 is suitable for searching on the stored data. The security of this scheme relies on the intractability of DBDH problem. The proposed scheme 2 does not employ *admissible encoding scheme*, and is easier than scheme 1 in PECK step. The security of this scheme relies on the intractability of DBDHI problem.

Acknowledgement

The authors would like to thank anonymous refrees for their comments, especially on the redundancy of EFT (existential forgery of trapdoor). The comments were very helpful to improve the presentation of this paper.

References

1. P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott, "Efficient algorithm for pairing-based cryptosystems," *CRYPTO 2002,* LNCS 2442, pp. 354–369, Springer-Verlag, 2002.
2. D. Boneh and X. Boyen, "Efficient selective-ID secure identity based encryption without random oracle," *EUROCRYPT 2004,* LNCS 3027, pp. 223–238, Springer-Verlag, 2004.
3. D. Boneh and X. Boyen, "Short signatures without random oracles," *EUROCRYPT 2004,* LNCS 3027, pp. 56–73, Springer-Verlag, 2004.
4. D. Boneh, G. Di Crescenzo, R. Ostrovsky and G. Persiano, "Public key encryption with keyword search," *EUROCRYPT 2004,* LNCS 3027, pp. 506–522, Springer-Verlag, 2004.
5. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *CRYPTO 2001,* LNCS 2139, pp. 213–229, Springer-Verlag, 2001.
6. S. Galbraith, K. Harrison and D. Soldera, "Implementing the Tate pairing," *ANTS-V,* LNCS 2369, pp. 324-337, Springer-Verlag, 2002.
7. P. Golle, J. Staddon and B. Waters, "Secure conjunctive keyword search over encrypted data," *ACNS 2004,* LNCS 3089, pp. 31–45, Springer-Verlag, 2004.
8. A. Joux, "The Weil and Tate pairings as building blocks for public key cryptosystems," *ANTS-V*, LNCS 2369, pp. 20–32, Springer-Verlag, 2002.
9. A. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology*, Vol. 14, No. 4, pp. 255–293, Springer-Verlag, 2001.
10. V. S. Miller, "Short programs for functions on curves," *unpublished manuscript,* 1986. Available from `http://crypto.stanford.edu/miller/miller.pdf`.
11. S. Mitsunani, R. Sakai and M. Kasahara, "A new traitor tracing," *IEICE Trans. Fundamentals,* Vol. E85-A, No. 2, pp. 481–484, 2002.
12. D. J. Park, K. Kim and P. J. Lee, "Efficient searchable public key encryption scheme (in Korean)," *Proceedings of CICS-S04,* pp. 521–526, 2004.
13. M. Scott and P. S. L. M. Barreto, "Compressed pairing," *CRYPTO 2004,* LNCS 3152, pp. 140–156, Springer-Verlag, 2004.
14. F. Zhang, R. Safavi-Naini and W. Susilo, "An efficient signature scheme from bilinear pairing and its application," *PKC 2004,* LNCS 2947, pp. 277–290, Springer-Verlag, 2004.

A Probabilistic Method for Detecting Anomalous Program Behavior

Kohei Tatara[1], Toshihiro Tabata[2], and Kouichi Sakurai[2]

[1] Graduate School of Information Science and Electrical Engineering,
Kyushu University
tatara@itslab.csce.kyushu-u.ac.jp
http://itslab.csce.kyushu-u.ac.jp/

[2] Faculty of Information Science and Electrical Engineering,
Kyushu University, Japan
{tabata,sakurai}@csce.kyushu-u.ac.jp

Abstract. In this paper, we, as well as Eskin, Lee, Stolfo [7] propose a method of prediction model. In their method, the program was characterized with both the order and the kind of system calls. We focus on a non-sequential feature of system calls given from a program. We apply a Bayesian network to predicting the N-th system call from the sequence of system calls of the length $N-1$. In addition, we show that a correlation between several kinds of system calls can be expressed by using our method, and can characterize a program behavior.

Keywords: Intrusion detection, Anomaly detection, System call, Bayesian network

1 Introduction

The increase of computers connected to a network with the rapid spread of the Internet is being enhanced. Therefore, many cases of an unauthorized access to the computer by the malicious user are reported, and intrusion detection system which detects such intrusions is getting more and more important.

Many intrusions exploit a vulnerability which is inherent in a program, called buffer overflow. An attacker rewrites the return address of a function by overflowing an internal buffer. Thereby, the attacker can take over the control of the program and it is possible to execute arbitrary codes [1]. On the other hand, there are researches of anomaly detection system which detects the intrusion using buffer overflow by monitoring the control flow of a program [2–11, 14].

An anomaly detection consists of a learning period which learns behavior during the normal operation of a program, and a monitoring period which supervises the action of a program by comparing with the records of normal operations. Such an anomaly detection system has an advantage of possibility to detect an novel intrusion compared with the system which only detects a known intrusion based on the signature. However, the overhead of anomaly detection which supervises the execution of a program becomes comparatively large. So it is important to chose the data which characterize the operation of a program.

Forrest et al. [2] showed that it is possible for normal operation of a program to be characterized by the history of the sequences of system calls which a program emits. If an intrusion using buffer overflow happens, since some sequences of system calls which are not seen during the normal operation will be observed, it is detectable.

Using the method based on N-gram, we can break a sequence of system calls into the sub-sequences of fixed length N, and judge whether the operation of a program is normal or not by comparing with those. In order to make only N-gram applicable to comparison, it is greatly dependent on N whether incorrect detection takes place. Forrest et al. refered to the optimal value of N, namely 6. The 6 is experimentally optimal value on the trade-off between detection capability and efficiency. That is, although efficiency improves when the N is set as less value, detection capability declines a little. They did not touch on the reason why the value of 6 was drawn as optimal value, but was treated as the "magic number" for some years ahead. In order to abolish the necessity of asking for the optimal value of N from the training data, some researchers made attempts to generate a finite state machine [6]. And, other researchers also set N from the normal operation data automatically [7]. Lee et al. [14] did research which explains the mechanism of $N = 6$, from a standpoint of information theory using conditional entropy.

In recent years, the research which clarified the reason was reported by analyzing the data which Forrest et al. exhibited [12]. Kymie et al. proved that the condition in which at least one "Minimal foreign length" exists in the sequences of system calls collected when an intrusion happened is that the value of N is six or more. Minimal foreign length is the minimum value of N in which at least one unique sub-sequence exists, when the sequences of system calls are divided into the sub-sequences of length N [12]. Clearly from their report, in order to use system call as normal data of an anomaly detection, the features from the sequences of system calls used for learning need to differ from those observed from anomalous processes. Although the more than 200 kinds of system calls exist in the current version of Linux, about dozens of kinds of the system calls are actually emitted by the program at most. The total number of possible N-grams is too small to characterize the behavior of program by the system calls in the case of $N = 1$. If Σ kinds of system calls are published in the program, then there are Σ^N possible sequences of length N. Therefore, as Σ is large, the probability under which Minimal foreign length is found increases by leaps and bounds. Kymie et al. showed the method of calculating the optimal value of N, when the stide [2] and Hofmyer et al.'s method [3] were used for an anomaly detection system. However, when these methods are applied to other data sets, we do not affirm whether $N = 6$ may become the "magic number." When the optimal value of N is 7 in a certain data set, performing anomaly detection using the value 6 contains a possibility of overlooking abnormal sequences.

On the other hand, there is a research using the rule learning program, called RIPPER, which focuses on correlation of system calls [13]. In this paper, Lee et al. input those sequences which are attached the labels, "normal" or "abnormal" and generated some If-then rules, and tried to extract the feature which can classify the normal and abnormal operations The data which they used in their experiment was the same as the data of sendmail which Forrest et al. used. And the procedure of generating the data was described in [2]. The rule set which Lee et al. generated took form, such as "if $p_2 = 104$ and $p_7 = 112$ then the sequence is *normal*" (Here, $p_i = j$ means that the i-th system call number in a certain sequence of system calls is j.) Lee et al. showed that it is possible to detect anomalous behavior using this rule set. However, in order to generate this rule set, you have to attach the label of being normal or abnormal to each sequence of training data. They also aimed to learn the correlation between each system calls. That is, they predicted the N-th system calls or the middle system calls in sequences of length N and showed as a result that detection accuracy depends on the value of N.

Eskin et al. [7] introduced a method of calculating the optimal value of N from training data [7]. They obtained it by substituting the conditional probability of transition between each system calls for the formula of the conditional entropy of Shannon and chosen sequence length with minimal entropy. This value is the most efficient value calculated information-theoretically. Then, they also proposed a "prediction model" which expects a N-th system call from the sequence of the length $N-1$ using those conditional probabilities and succeeded in increasing accuracy of anomaly detection. As shown above, we can calculate the optimal value of N. But we do not know what propety in the conventional methods based on N-gram affects detection accuracy. In order to get more efficiency, we need to clarify such a property.

In this paper, we propose an alternative method of prediction model. We apply a Bayesian network for predicting the N-th system call from the sequence of system calls of the length $N-1$ as well as Eskin et al. We show that a correlation between several kinds of system calls can be expressed by using our method, and can characterize a program behavior. Then we can decrease the number of kinds of sequences to use in anomaly detection. The composition of this paper is as follows. In Section 2, we compare our method with previous result. The difference between Eskin et al.'s and our method is specifically clarified, and our contribution is described. In Section 3, we describe the algorithm of our method. In Section 4, we also present the result of experiment. Then, in Section 5, we consider a validity of our method. We conclude in Section 6.

2 Non-sequential Model

Eskin et al. [7] proposed the model using sparse Markov transducers based on sparse prediction tree [7]. In this method, they replaced the overlapping symbol with the wild card in the sparse Markov tree obtained from a set of sequences of system calls, and reduced the number of branches. They showed that the probabilistic threshold outperformed mismatch threshold empirically.

The advantage of using a Bayesian network is that sequence of length $N-1$ correlating with a N-th system call can be treated as non-sequential one. For

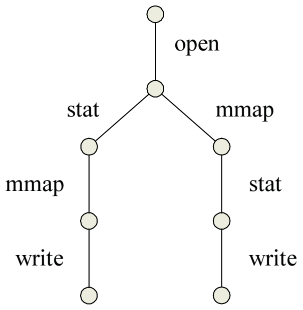


Fig. 1. The example of sparse Markov tree (The sequence length N is 4).

example, it is assumed that we observe the sequence of length 4 such as {open mmap stat write} and {open stat mmap write}. By using sparse Markov transducers, we can obtain the sparse Markov tree, shown as Figure 1.

This tree structure has branched at the portion of open system call. On the other hand, when a Bayesian network is used for predicting the N-th system call from the preceding sequence of length $N-1$, the $(N-i)$-th system call only correlate with N-th. That is, the sequence {open mmap stat write} and {open stat mmap write} is treated as the same one. When some system calls are classified according to the kind, we consider correlation between those sets.

3 Our Proposal

This section explains outline and concrete procedure of our proposal.

3.1 Outline of Our Proposal

In a learning period, a Bayesian network is formed from the sequences of system calls which an application program emits. A system call is a function for an application program to use the function which an operating system offers, and the number corresponding to a name is assigned, respectively. Henceforth, expression called a system call S_i designate the system call whose system call number is i.

A Bayesian network [18] can express the qualitative dependency between random variables with non-circulating directed graph, and is suitable for expressing a phenomenon including uncertainty, and causal relation. In the background of the our proposal, there is an idea that causal relation exists between each system calls in the sequence of system calls from a program. Therefore, the validity of our method is shown experimentally to the last. The relation between system call S_i and S_j is designated by the directed link $S_i \rightarrow S_j$ in a Bayesian network. S_i is called parent node, and S_j is called child node. The quantitive causal relation between S_i and S_j is expressed with the conditional probability $P(S_j|S_i)$. When there are two or more parent nodes, set of the parent nodes of S_j is designated by $\pi(S_j)$. The set of conditional probabilities, provided that all the values of parent nodes exists, is called Conditional Probability Table (CPT), and is used as normal operation data in our method.

In a monitoring period, using the learned bayesian network, the validity of the sequence of system calls which a program publishes is verified. When a program publishes a system call S_i, conditional probability $P(S_j|\pi(S_j))$ is calculated by using CPT obtained in the learning period. If the intrusion using buffer overflow occurs, some sequences of system calls which are not seen during the normal operation will be observed [2, 3]. It is expected from this character that the value of conditional probability takes a low value during anomaly operation, and a high value during normal operation continuously.

On the other hand, when sets of the parent nodes $\pi(S_i)$ are equal as for two or more system calls, it is thought that those conditional probabilities become small. When judging the height of conditional probability and the validity of issue, we are anxious about the number of incorrect detections increasing. We try to solve the problem by using the statistical technique, called Mann-Whitney U-test [19]. Mann-Whitney U-test is useful in case it tests whether there is a difference between the median values of two groups.

Hereafter, the concrete procedure in a leaning period and a monitoring period is explained.

3.2 Formation of a Bayesian Network

In a learning period, the procedure which forms a Bayesian network from sequences of system calls is as follows.

1. In the sequences of system calls $X = \{X_1, \ldots, X_l, \ldots\}$, when it is assumed that a causal relation exists between the l-th system call X_l and the set of system calls $\{X_{l-1}, \ldots, X_{l-D}\}$ published for the past D times, it is set to $\pi(X_l) = \{X_{l-1}, \ldots, X_{l-D}\}$. The value of D (here, it stand for the degree of dependence) can be set up arbitrarily.
2. Suppose that the element of $\pi(X_l)$ be the node of a Bayesian network. Then all pair of two nodes if there exist a causal relation are connected by directed links.
3. The procedures (1) and (2) are repeated for all sequences of system calls.
4. $P(X_l|\pi(X_l))$ is calculated for all X_l.

Figure 2 is the example which actually complete the above-mentioned procedure from the sequences of system calls which the ftp program publishs, and form a Bayesian network. The number in parenthesis in Figure 2 expresses the system call number. In this case, a set of the parent nodes of a socketcall system call and a gettimeofday system call is equal. Therefore, as shown above, the conditional probability of these system calls may take a low value.

3.3 The Procedure of the Anomaly Detection in a Monitoring Period

The procedure of performing anomaly detection using the learned Bayesian network is as follows.

child node	parent node
...	...
stat	close, open, read
mmap	open, read, stat
close	read, stat, mmap
mprotect	stat, mmap, close
munmap	mmap, close, mprotect
socketcall	close, mprotect, munmap
gettimeofday	mprotect, munmap, socketcall
write	munmap, socketcall, gettimeofday
...	...

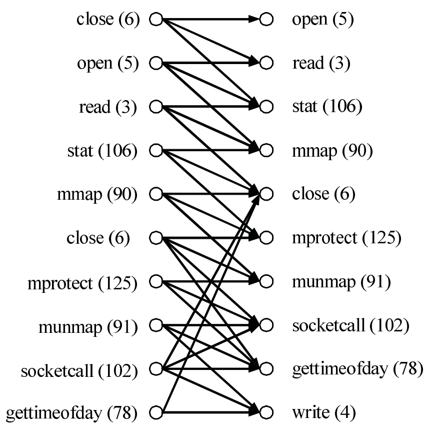


Fig. 2. The example of Bayesian network when the degree of dependency D is 3 (The number in the parenthesis expresses the system call number).

1. In the sequences of system calls $X = \{X_1, \ldots, X_i, \ldots\}$ a program emits, when it is assumed that a causal relation exists between the ith system call X_i and the set of system calls $\{X_{i-1}, \ldots, X_{i-D}\}$ published for the past D times, it is set to $\pi(X_i) = \{X_{i-1}, \ldots, X_{i-D}\}$. The value of D can be set up arbitrarily.
2. $A_i = P(X_i|\pi(X_i))$ is calculated from the CPT obtained during a learning period. Moreover, j is selected for $B_i = P(S_j|\pi(X_i))$ so as to takes the highest value.
3. Mann-Whitney U-test [19] is performed with the two groups $\{A_i, \ldots, \ldots, A_{i-I+1}\}$ and $\{B_i, \ldots, B_{i-I+1}\}$. A null hypothesis presupposes 'There is no difference between the median values of the two groups'. A significant level can be set up arbitrary.
4. When the U statistics calculated in (3) takes the value below a critical value, it rejects null hypothesis. That is, since there is a difference during the median values of two groups, the sequence is flagged as anomalous. If the number of anomalous sequences exceeds a threshold, it is judged that an intrusion occurs.

Importantly, we calculate $P(X_i|\pi(X_i))$ from the CPT, as described below. First we take the set of conditional probabilies of X_i, given $\pi(X)$ from which the hamming distance to $\pi(X_i)$ take a minimum value in the CPT. Then, we select the maximum conditional probability as $P(X_i|\pi(X_i))$ in the set. Needless to say, the hamming distance between two sequences affects the detection capability.

4 Experiment

This section describes the result of experiment according to the method which was stated in Section 3. A specific purpose is to prove that our method is able

Table 1. Details of data sets.

Program	# of seq.	# of seq. for training	# of seq. for testing	# of proc.
ftp	181,663	10,000	171,663	5
xlock	9,230,067	895,924	8,334,143	2
ps	8,589	4,112	4,477	11
login	13,739	6,128	7,611	13
sendmail	143,109	73,491	69,618	34

to detect the intrusion using buffer overflow and to compare the accuracy and performance of our method with the other method.

4.1 Data Sets

Forrest et al. [2, 3] showed that the short sequences of system calls issued by a program during its normal executions can characterize it. Those sequences are also different from the sequences of its anomalous executions as well as the executions of other programs. The many different methods are compared each other in the past research [5]. The data sets used in this research are publicly available at http://www.cs.unm.edu/˜immsec/data-sets.htm. Therefore, we also experiment using these data sets.

These data consist of "live" data which are recorded by the normal use and "synthetic" data which are recorded in the environments where the program options are chosen carefully. Although it was a somewhat rude way, in order to investigate the number of false positives and false negatives, we chose a suitable number of sequences of system calls from all these data at random, and used for training and monitoring. Table 1 describes the details of data sets.

4.2 Comparison

Many researchers have proposed the methods based on N-gram. tide, stide [5], and Hofmyer's method [3] are mentioned as an example which are successful empirically. We decided to use the Hofmyer et al. 's method [3] as comparison. Hofmyer et al. calculated the hamming distance between the sub-sequences of system calls of length N. When this hamming distance exceeds a threshold, that sequence was judged as abnormal. In their method, the program was characterized with both the order and the kind of system calls. So, we are interested in whether we obtained the same result or not, even with only non-sequential sequences of system calls.

4.3 Experimental Results

We measured the accuracy and performance of anomaly detection using our method. In order to evaluate the anomaly detection system, we can observe the number of true positives and false positives. True positive rate shows the percentage of sequences which are flagged as abnormal in the set of sequences issued by an anomalous process. And false positive rate expresses the percentage

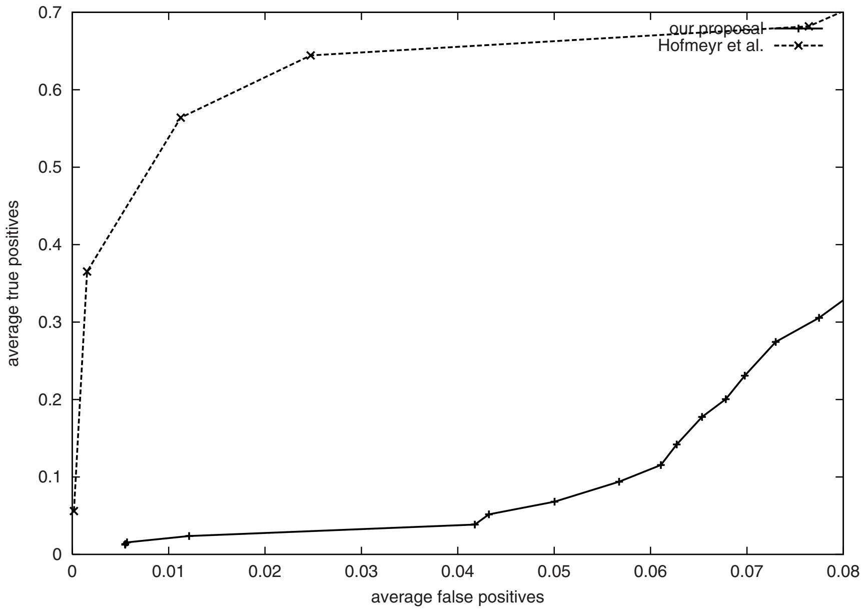


Fig. 3. ftp ($N = 6$).

of sequences accidentally judged to be anomalous in the set of sequences of system calls issued by a normal process. The result which we obtained is shown in Figure 3, 4, 5, 6, 7.

The experiment using ftp program resulted in the regrettable score shown in Figure 3. It is easy to check that the rate of true positives is remarkably small and the further analysis of the sequences of system calls in this program is required. In other programs, the accuracy of the anomaly detection of Hofmyer et al. and our proposal are dependent on some parameters. These parameters involve the threshold C in Hofmyer et al. 's method which limits the minimal hammming distance to decide whether a sequence of system calls is anomalous or not, N which is initial parameter [3], the critical value in our method and so on. They depend on the context as well as the sequence length N. That is, it means that the algorithm which finds out optimal values of them from normal data is required. Needless to say, when we select the appropriate parameter, we can achieve the results exceeding the Hofmyer et al. 's method. We selected the 6 as the sequence length N conventionally used in the past researches. The ROC curve in Figure 3, 4, 5, 6, 7 is obtained by trying the various parameters described above. The ROC curve shows that even if only the kind of system call is extracted as the feature, we can characterize the operation of program.

5 Consideration

In this paper, we showed experimentally that by paying attention to the correlation between several kinds of system calls and the non-sequential feature of

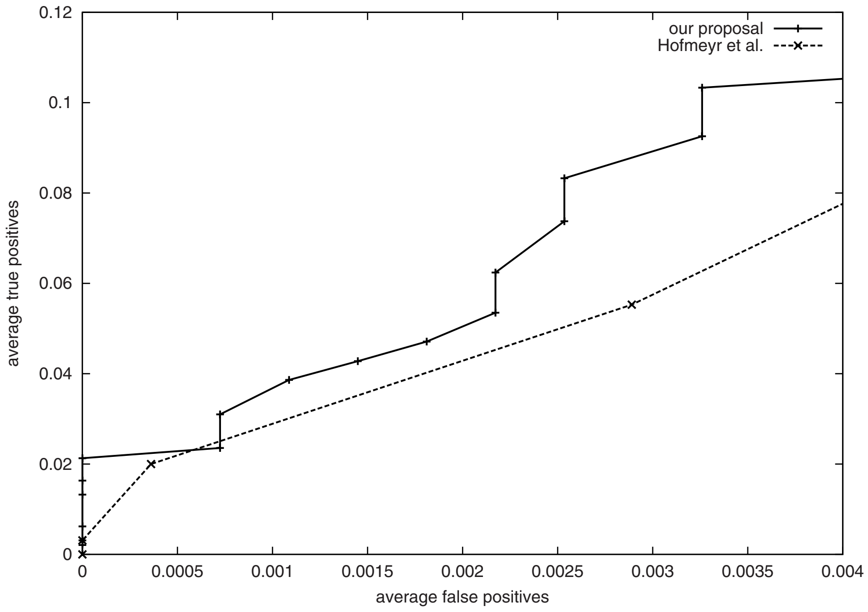


Fig. 4. login ($N = 6$).

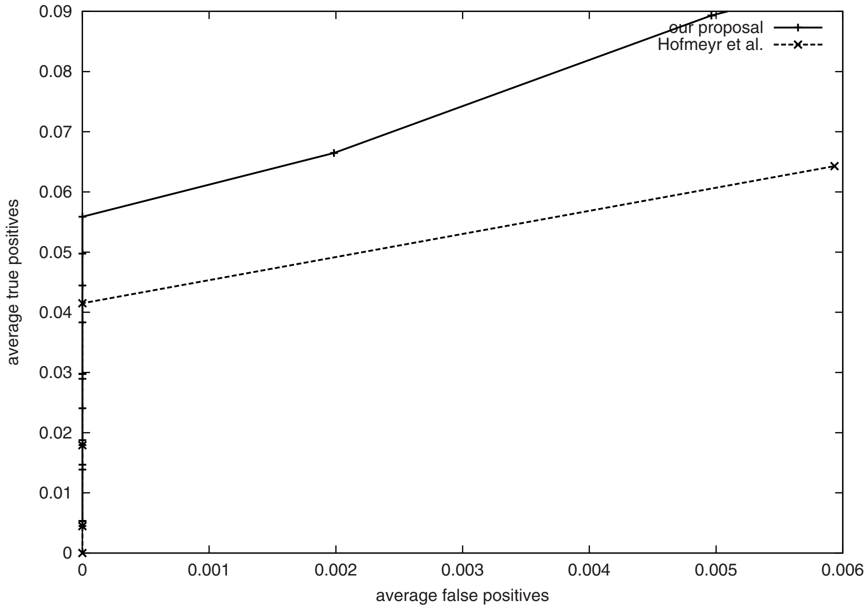


Fig. 5. ps ($N = 6$).

those, we can characterize the operation of program. On the other hand, there are many researches which observe sequential feature of the sequences. The reason for this trend is due to the size of space. When the Σ kinds of system calls

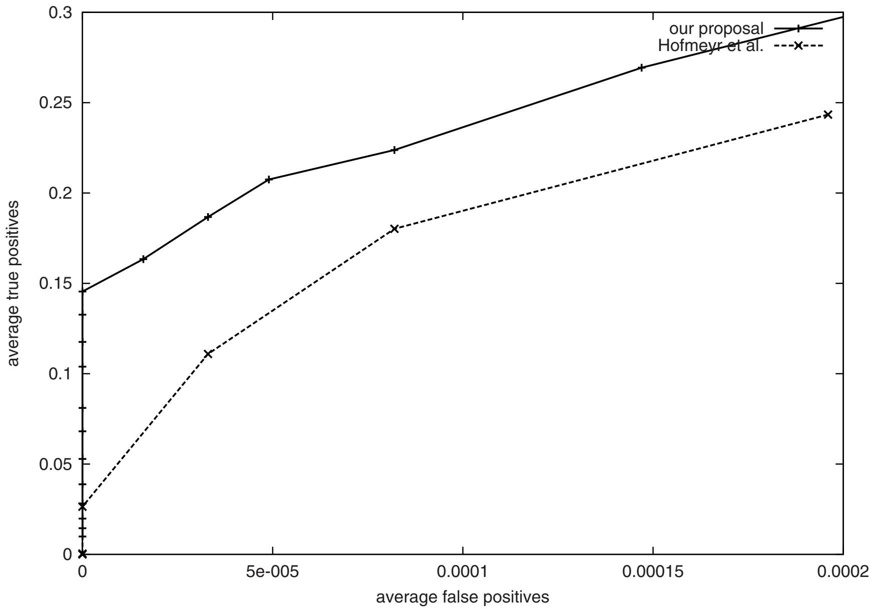


Fig. 6. sendmail ($N = 6$).

exist in an environment, the number of possible sequences in order to catch the feature of program is $|\Sigma|^N$. But, our method has only $N \cdot \frac{(|\Sigma|+N-2)!}{(N-1)!\cdot(|\Sigma|-1)!}$ kinds of possible sequences. This means that the more varieties of control flows in the program, the less number of unique sequences of system calls we can discriminate from all the other sequences. That is, this may drop the accuracy of the anomaly detection and increase the number of false positives. Fortunately, we can prove that non-sequential feature is large enough to distinguish between normal and abnormal operation. But, because the research which shows that it is possible to avoid an anomaly detection system by camouflaging sequences of system calls skillfully is also reported [17], it is desirable that we can express many features of programs to enhance the accuracy, while holding overhead.

On the other hand, there is a research that does not only regard a sequence of system calls as a feature, but also describes the action of a program in higher dimensions with other information about it. Wagner et al. performed static analysis of a source code and created the non-deterministic pushdown automaton expressing control flow of a program [15]. In recent years, Oyama et al. developed the method, added inspection of stack information, and created a state transition diagram leading to no false positives [16]. In these anomaly detection, since there are many features which should be inspected, more amounts of calculation are needed, so we are anxious about the overhead. Our ultimate goal is an anomaly detection system as an intrusion detection system. For that purpose, it is necessary to consider not only the capability of anomaly detection in real time but the amount of calculation and usability.

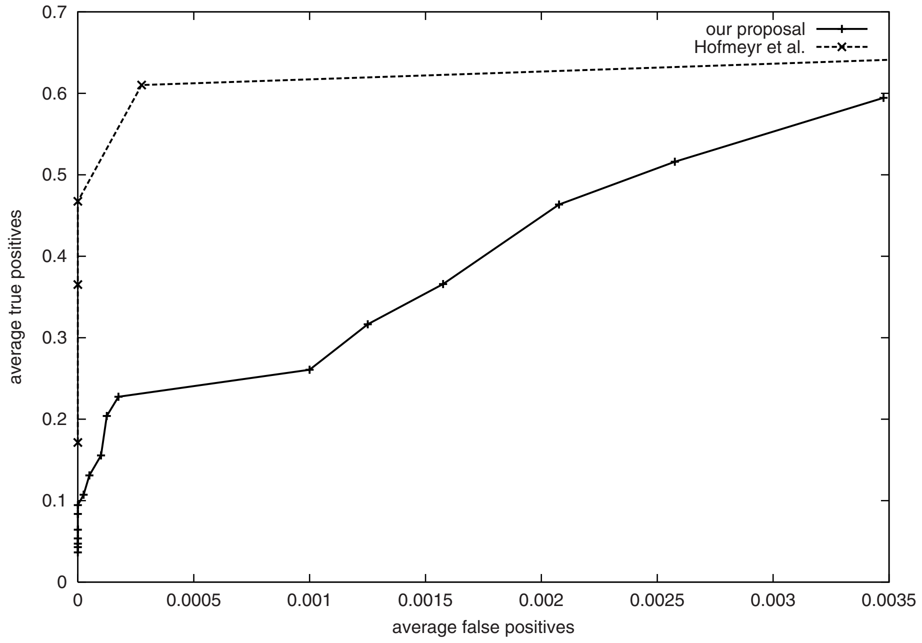


Fig. 7. xlock ($N = 6$).

6 Conclusion

In this paper, we proposed a method of anomaly detection based on N-gram. Since our proposal has put the basis on correlation being between system calls from the standpoint of information theory as well as Eskin et al. [7]. Moreover, we apply Bayesian network to concretely describe the correlation between several kinds of system calls. Thereby, we can clarified the non-sequential correlation between system calls experimentally. Future work is investigation of the validity of the proposal system in more programs.

References

1. Beyond-Security's SecuriTeam.com. Writing Buffer Overflow Exploits - a Tutorial for Beginners. http://www.securiteam.com/securityreviews/5OP0B006UQ.html (accessed 2003-09-05).
2. S. Forrest, S. A. Hofmeyr, A. Somayaji, T.A. Longstaff. A sense of self for Unix processes. In the *1996 IEEE Symposium on Computer Security and Privacy.*
3. S. Forrest, S. A. Hofmeyr, and A. Somayaji, Intrusion detection using sequences of system calls. *Journal of Computer Security*, Vol.6, pp. 151–180, 1998.
4. G. Helmer, J. Wong, V. Honavar, and L. Miller. Intelligent agents for intrusion detection. In *IEEE Information Technology Conference*, pp. 121–124, Sep. 1998.
5. C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy,*

6. C. Marceau. Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the New Security Paradigms Workshop 2000*, 2000.
7. E. Eskin, W. Lee, and S. Stolfo, Modeling system call for intrusion detection using dynamic window sizes. In *Proceedings of the 2001 DARPA Information Survivability Conference & Exposition*, Anaheim, CA, June 2001.
8. S. Li, A. Jones. Temporal Signatures for Intrusion Detection. *17th Annual Computer Security Applications Conference* , Dec. 10–14, 2001
9. A. P. Kosoresow, S. A. Hofmeyr, Intrusion Detection via System Call Traces, IEEE Software, vol. 14, pp. 24–42, 1997.
10. R. Sekar, M. Bendre, P. Bollineni and D. Dhurjati, A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors, In *Proceedings of the IEEE Symposium on Security and Privacy*, 2001.
11. Y. Liao, V. Rao Vemuri, Using Text Categorization Techniques for Intrusion Detection, In *Proceedings of the 11th USENIX Security Symposium.* Aug. 2002.
12. K. M. C. Tan, R. A. Maxion, "Why 6?" Defining the Operational Limits of stide, an Anomaly-Based Intrusion Detector. In *Proceedings of IEEE Symposium on Security & Privacy*, pp. 188–201, 2002.
13. W. Lee, S. Stolfo, and P. Chan, Learning Patterns from Unix Process Execution Traces for Intrusion Detection, In *Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management*, 50-56, 1997.
14. W. Lee and D. Xiang, Information-Theoretic Measures for Anomaly Detection, In *Proceedings of The 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.
15. D. Wagner, D. Dean, Intrusion Detection via Static Analysis, In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, 2001.
16. M. Oka, H. Abe, Y. Oyama, K. Kato, Intrusion Detection System Based on Static Analysis and Dynamic Detection, In *Proceedings of Forum on Information Technology (FIT 2003)*, Japan, Sep. 2003.
17. D. Wagner, P. Soto, Mimicry Attacks on HostBased Intrusion Detection Systems. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, Nov. 2002.
18. Y. Motomura, I. Hara, User Model Construction System using Probabilistic Networks. http://staff.aist.go.jp/y.motomura/ipa/ (accessed 2003-09-05)
19. W. J. Conover, Practical Nonparametric Statistics, John Wiley & Sons, Inc., New York, 1971.

Service Discrimination and Audit File Reduction for Effective Intrusion Detection*

Fernando Godínez[1], Dieter Hutter[2], and Raúl Monroy[3]

[1] Centre for Intelligent Systems, ITESM–Monterrey
Eugenio Garza Sada 2501, Monterrey, 64849, México
fgodinez@itesm.mx
[2] DFKI, Saarbrücken University
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
hutter@dfki.de
[3] Department of Computer Science, ITESM–Estado de México
Carr. Lago de Guadalupe, Km. 3.5, Estado de México, 52926, México
raulm@itesm.mx

Abstract. Current IDSs can be easily overwhelmed by the the amount of information they ought to analyse. By pre-processing the information, this paper aims both to alleviate the computational overhead involved in intrusion detection and to make IDSs scalable. Regardless whether it is a sequence of network packets or a sequence of system calls, the information an IDS analyses is often redundant in at least two respects: first, every entry in the sequence may contain spurious information; second, any sequence may contain redundant subsequences. By using Rough Sets we have identified key attributes for every entry eliminating spurious information, without missing chief details. Using n-gram theory we have identified the most redundant subsequences within a sequence, and then substituting them with a fresh tag, resulting in a sequence length reduction. To make an IDS scalable we have proposed to structure the IDS as a collection of sensors, each of which is specialised to a particular service, i.e. `telnet`, `smtp`, etc. To approach service selection, we suggest the use of Hidden Markov Models, trained to detect an specific service described by a family of n-grams. Our results are encouraging, we have obtained an average reduction factor of 12. Using the service discriminator we have also written a simple, yet effective, misuse IDS. The impact over detection and false alarm ratio using reduced sequences is negligible.

Keywords: Dimensionality Reduction, Object Reduction, Intrusion Detection, Misuse Detection

1 Introduction

Intrusion detection is concerned with the timely discovery of any activity that threatens the integrity, availability or the confidentiality of an IT system. It often amounts to detecting an known pattern of computer misuse or an deviation to

* This research is supported by three research grants CONACyT 33337-A, CONACyT-DLR J200.324/2003 and ITESM CCEM-0302-05.

expected user behaviour. Regardless of the approach, however, current Intrusion Detection Systems (IDSs) get easily overwhelmed for the amount of information they ought to analyse. This paper suggests an pre-processing mechanism that aims both to make IDSs tractable and scalable.

To make intrusion detection more tractable, we compact the information to be analysed without losing key information. This input information, whether it being a sequence of network commands or a sequence of system calls, is often redundant in at least two respects. First, any entry in either of these sequences, called *object* for short, may contain spurious information. And second, any sequence may contain a number of irrelevant objects. Using rough sets [1], we have successfully identified the object attributes that best characterise the object information without missing chief details [2]. Using n-gram theory [3], we have successfully identified those subsequences of objects that most frequently occur within a session and then folded them up with a fresh tag, thus reducing the session length.

To make it scalable, we suggest to structure intrusion detection as a collection of sensors, each of which is specialised to a particular service. To approach service discrimination, we suggest to use a Hidden Markov model (HMM) trained to pinpoint what n-gram family (characterising a service) the given header of an input sequence is likely to belong to.

While our mechanism is applicable in most contexts, we have focused on a sequence of system calls and used, for testing purposes, BSM log files borrowed from the DARPA repository [4, 5]. Our results are encouraging. In average, we can shrink an input sequence by a factor of 12. Moreover, using an `ftp`, `smtp` and `telnet` discriminator, we have written an simple, yet effective intrusion detector. It consists of the parallel composition of 3 independent IDSs, each of which deals with an separate service discriminator IDS.

This paper is organised as follows: §2 describes an novel architecture to an intrusion detection system. §3 and §4 respectively describe our methodology for object normalisation and session shrinking. Then §5 introduces an HMM-based service discriminator, with which we separate multiplex input sequences according to the service each of which belongs to. §6 shows the impact of our reduction techniques on intrusion detection. §7 contrasts related work and §8 presents suggestions for further work and the conclusions we have drawn from our experiments.

2 An Architecture for an Intrusion Detection System

In this section we present a novel architecture for an intrusion detection mechanism. The architecture is novel in two respects: i) it incorporates an service discriminator, with which our IDS separates multiplexed input sequences in terms of the service each of which belongs to. And ii), it incorporates both an misuse detector (MIDS) and anomaly detector (AIDS) [6], probabilistically combining their output in order to increase the detection ratio and reduce the false-positive one. The IDS simultaneously checks the input for a misuse and an anomaly. Fig. 1 shows the IDS architecture.

The first element of the architecture is an attribute reducer which filters out superfluous attributes from a sequence of system calls. The system calls with the normalised number of attributes is then shrinked by substituting repetitive sequences (in the form of n-grams) with a fresh tag. Then the service discriminator uses the first system calls from each session to select the service it belongs to. Each service the system is monitoring has its own discriminator. Once the session is categorised and reduced, it is parsed by the MIDS and AIDS modules. The output of both modules is then used to estimate the probability of an intrusion.

Now we will briefly describe the role of each IDS component.

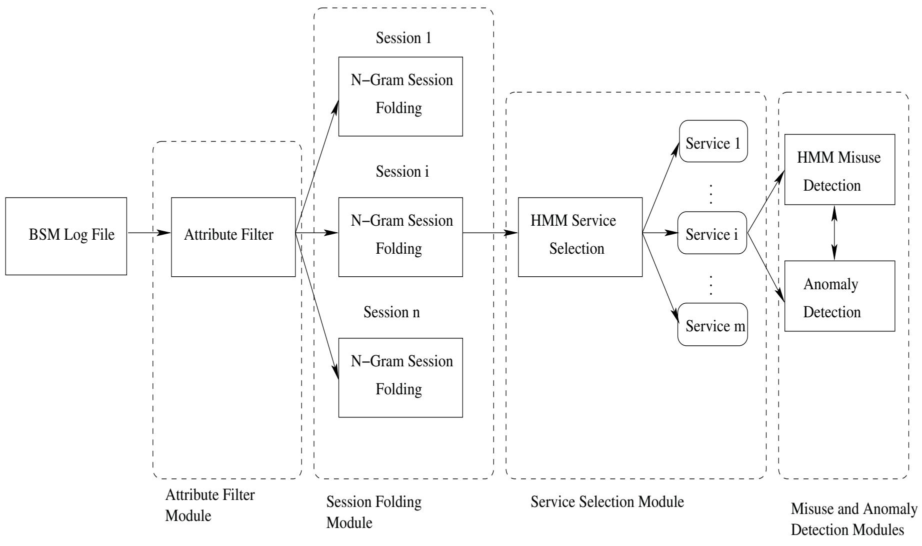


Fig. 1. IDS Architecture.

3 Attribute Reduction Using Rough Sets

This section describes the first of two serial methods that are used to shrink the information to be analysed by the IDS. As mentioned above, we restrict ourselves to the inspection of sequences of system calls.

A system call consists of an mnemonic and a number of arguments. The number of arguments varies from one system call to other and there might be a few dozens of argument types in a typical operating system. Thus, if we were to use an sequence of system calls as the input to an intrusion detector, then we will have to consider every possible system call as well as each of its associated arguments. Since this would make the detection intractable, IDS often consider only the system call mnemonic, dropping out all of the arguments, hence missing possible key information. To get around this problem, we suggest to normalise every system call such that it contains the least number of arguments with respect to a given target for information coverage [7]. We approach system call normalisation using rough sets [1].

Rough sets are used to identify superfluous arguments of a collection of system calls. They allow for flexible element classification by using a rough approximation to set membership, c.f. the upper and lower limits that define a rough boundary. Rough sets estimate the relevance of an element by computing the element dependencies with respect to a given decision class. It achieves attribute set covering by imposing a discernibility relation. Rough set's output, purged and consistent data, can be used to define decision rules.

3.1 Rough Sets for System Call Normalisation

Normalising the arguments of a system call corresponds to finding an attribute reduct. So, to achieve system call normalisation, ideally we would just need to collect together as many as possible sequences of system calls and make our favourite rough set tool, e.g. Rosetta [8], to process them. However this is not practical. So we ran a number of separate reduction experiments, each of which considers an independent sequence of system calls, and then merged together the associated reducts. Then, we find what we called a *minimum common reduct* performing an statistical analysis that removed those call arguments that appeared least frequently.

3.2 Reduct Extraction

In our reduct extraction experiments, we used audit borrowed from the DARPA repository. In total, we considered 8 logs, randomly chosen from the existing 25, in 1998. Then the logs were put together into a large, single log which was evenly divided in segments of 25,000 system calls, yielding 365 segments. For each partial log file, we made Rosetta extract the associated reduct using Johnson's algorithm, selecting 100% information converage. Then the resulting reducts were sampled using an frequency-based discriminant, and out of that analysis we constructed a minimum common reduct (MCR). This MCR keeps most information of the original data but minimises the number of attributes. The largest reduct in the original set has 15 attributes and our minimum common reduct has 18 attributes. This is still a 66.66% reduction in the number of attributes. The 18 chief attributes are shown below:

Access-Mode	device-ID	exec-arg-1	Effective-Group-ID
Owner	arg-value-1	Socket-Type	Process-ID
Owner-Group	arg-string-1	Remote-IP	System-Call
File-System-ID	arg-value-2	Audit-ID	
inode-ID	arg-value-3	Effective-User-ID	

3.3 Reduct Validation

To validate the performance of the reduct we appeal to so called association patterns. An *association pattern* is basically a pattern that, with the help of wild-cards, matches part of an input information. Given both a reduct and a log file, the corresponding association patterns are extracted by overlaying the reduct over that log file, and reading off the values. Then the association patterns are compared against another log to compute how well they cover that log file information. Thus, our validation test consists of checking the quality of the association patterns generated by our output reduct, considering two log files. The rationale behind it is that the more information about the system the reduct comprehends the higher the matching ratio the association patterns of that reduct will have.

The reduct was validated using log files from the 1998 and 1999 DARPA repositories. The results we got showed that we need less than a third of the 51 original attributes to describe an audit file with minimum loss of information. A complete account of our experiments can be found in [2].

4 Session Folding Using N-Gram Models

We now turn our attention to the second method used for reducing the information to be used for inspection. The method shrinks an input sequence by substituting repetitive subsequences. We used an n-gram model to identify key repetitive subsequences that largely reduce an given sequence of system calls. Our results show that we can reduce an input sequence by an factor of 4, using an only 19 n-gram model.

4.1 N-Gram Theory

To identify the most repetitive session subsequences, we use n-gram theory [3]. N-gram theory has been largely used in the context of language prediction. It has also been used in anomaly detection by Maxion [9, 10], Marceau [11], Wespi [12], and Forrest [13]. Let an *n-gram* be a sequence of n symbols, system calls in our case. Then, an *n-gram language model* is used to calculate the probability that a system call s will appear at position n, given the occurrence of an $(n-1)$-gram; that is, a sequence of *n-1* system calls. So the n-gram language model enables us to estimate the likelihood of appearance of a given n-gram along a larger sequence. By using n-gram theory, we identified the n-grams with the highest rate of occurrence among different services, namely: i) `ftp`, ii) `telnet` and iii) `smtp`, and the ones general to all services. This section shows a collection of n-grams that are very likely to be found repetitively in any computer session.

4.2 N-Gram Extraction

N-Gram extraction consists of the application of a blind, exhaustive procedure. As a result, we obtained the n-grams that occur most frequently in the training sessions. Although in theory n-gram model creation should consider all possible n-grams, in practice only n-grams that exist within the training data are used. For example for a 10-gram with a vocabulary of 200 tokens, the possible number of sequences should be $200^{10} \times 199$. However, in our experiments, we found only 2291. N-Gram extraction prunes all sequences with 0 occurrences. For the final language model a low probability is assigned to pruned sequences. In our calculations, we considered the n-grams that were pruned from the entire log file as well as those pruned from different services within that file.

The n-grams were extracted from a set of all concatenated sessions from the same day (i.e. a daily log file). The sessions were concatenated because we wanted to extract the n-grams with higher frequency or probability regardless the session they belong to. Every n-gram used was assigned a priority to decide the order of substitution.

If an n-gram is present in the training log files an occurrence frequency is assigned to it, if it is not present then an occurrence probability is assigned. Using the n-gram count and the language model, we identified the n-grams with higher occurrence frequency or probability. We considered different n-gram sizes to find n-grams that provided a high reduction rate.

Using both n-gram occurrence frequency and probability, we estimated a reduction ratio later used as a priority Pr for every selected n-gram. Not only does this ratio consider large n-grams with a high frequency, but it also considers the total number of system calls N on the training sequence sessions from which the n-grams were extracted. We calculated a reduction percentage for every selected n-gram, i.e. how much will a given n-gram reduce a log file.

If we use n-gram frequency f the n-gram priority Pr is given by:

$$Pr = \frac{n \times (f+1)}{N} \tag{1}$$

By contrast if we use n-gram probability P the n-gram priority Pr is given by:

$$Pr = P \times n \tag{2}$$

In both cases n is the size of the n-gram. Both equations provide a reduction ratio for the input n-gram. Using this ratio we choose the sequences that provide a high reduction rate. We will now describe the priority assignment for service exclusive n-grams.

To avoid overlapping when making an n-gram substitution we used a priority queue approach to select the n-gram to substitute. The queue was used to substitute high ratio n-grams. We created a window of size equal to the largest n-gram in the queue. Every time the window was full, we tested its contents against all n-grams in the queue. The order of the priority queue is given by the ratio defined in equations (1) and (2). By substituting n-grams with higher ratio we warranty that, even if there is overlapping, only the n-grams that provide maximum reduction are used. Then by substituting an n-gram with a new symbol we avoided further substitution on that segment resulting in overlapping elimination. We will now explain how the priority queue was applied to log file reduction.

4.3 N-Grams for Session Folding

We used three dimensional histograms to analyse the amount of n-grams with a frequency similar to a multiple of the session number in the training data. If there are m sessions and the frequency of an n-gram is a multiple of m, then that n-gram is more likely to be common among every session. The same histograms can be used to know in advance how many n-grams to look for when making the frequency based selection.

Based on this frequency analysis we identify the n-grams with a desired frequency. This analysis makes the extraction of such n-grams much easier and faster. From the extraction we chose 100 n-grams for the reduction. These n-grams are mostly ones with a frequency equal to a multiple of the number of sessions. Also based on the language model probabilities we selected about 50 n-grams with a probability above 98%.

After extracting n-grams for every log file a language model is generated. One option that needs explaining is the selection of the *discounting strategy*. The one provided with the software we used is the good-Turing estimator. According to the literature [3] it is a great method to avoid elimination of unseen n-grams without imposing considerable probability reduction of existing n-grams.

After selecting n-grams with high frequency or high probability we calculated the reduction ratio. The reduction ratio will sort selected n-grams in a priority queue for subsequent replacement of such n-grams in the log files. Using the priority queue we can substitute or fold a given session. Prior to the substitution we load any abbreviation dictionary that we have previously used in order to avoid repetitive abbreviations in sub-sequent reductions. Now we will describe how the validation is done using the resulting set of n-grams.

4.4 Reduction Validation

Extracted n-grams provide an average reduction of 74% within the training sessions. We also used the n-grams to reduce unseen sessions from 5 different log files. As input we have an unseen log file and as output we provide the reduced log file. In table 1 we show the reduction ratio over the validation data. The table columns are: log file id, original number of system calls, compressed number of system calls, and number of n-grams used in the reduction. Last row of the table shows the results of applying the reduction to a file with only `telnet` sessions.

Table 1. Validation Results.

Log File	Original Object #	Compressed Object #	Reduction %	Used N-grams
1	776,000	270,000	65.3%	7
2	1,800,000	486,000	73%	12
3	1,150,000	344,000	70.1%	5
4	801,000	175,000	78.2%	9
5	1,158,000	392,000	66.2%	5
Telnet	209,000	48,000	77.1%	5

Our results show that we can reduce session length, up to a fourth of its original size with minimal information loss. We will show in the following section how n-grams can be grouped and used for service discrimination.

5 Hidden Markov Models for Service Selection

Every service such as `telnet`, `ftp`, and `smtp` has a distinctive session header. Just as we showed in the previous section, n-grams can be used to identify such repetitive session headers. However there is a catch; from session to session there are small variations in session headers. This variations of similar n-grams will be regarded as different and therefore will not be considered by our folding method. With the use of hidden Markov Models (HMM) to group similar n-grams, we group such n-grams as a family.

We will begin the section by pointing out the advantages of having a service discrimination module.

5.1 Benefits of Service Discrimination for Intrusion Detection

According to Axelsson [14], building a profile for anomaly detection requires a staggering amount of information. All these information can easily overwhelm any IDS. To get around this situation either we reduce the information, or we split the profile in smaller parts, e.g. a user profile divided in services. Both solutions are not mutually exclusive since reduced information can also be split up. Since we have already reduced the information, now we reduce the workload for our IDS by discriminating services. With this selection we advocate the detection to a single service. The main advantages of using a service selection are shown below:

Flexibility, the configuration for the service we are monitoring can change, i.e. different port number. Service selection will not be affected since we are analysing behaviour and not a specific configuration.

Efficiency, the search space is reduced whether we make misuse or anomaly detection.(e.g. only attacks for the selected service are verified).

Scalability, in order to monitor a new service, we only need to add a discriminator for that service, and train the misuse and anomaly detection modules accordingly.

By using the service discriminator, our IDS is flexible, scalable, and efficient, it uses a reduced search space to look for an intrusion. These characteristics have been of interest for an IDS since Denning's paper [15], and have been pointed out by Forrest et al. [16], Zamboni et al. [17] and Mell et al. [18, 19]. An ideal IDS has more characteristics but ours complies with the ones mentioned above. Now we will describe the process to select service selection n-grams.

5.2 N-Gram Selection

The methodology to extract n-grams representing session headers is similar to the one proposed in §4 only up to the frequency analysis. We want to find the n-grams that characterise the beginning of every session of the same service. Services that need authentication might have two different n-grams; one for successful, and one for unsuccessful authentication. Such n-grams will allow us to identify the service a session belongs to.

The analysis is made to the sequence of system calls previous to the call `fork` which usually indicates the beginning of user interaction process. Wespi et al. in [20] consider sequences that begin with a `fork` and end with `exit`. By using our service selection method, when a `fork` is reached the service is already selected.

To identify the sequence of system calls prior to the `fork` call, we begin with an n-gram frequency analysis. We know every session of the same service has the same header but only to a certain point. After such point, headers begin to show small variations. These variations are repetitive among sessions. The same variation will show on many sessions. That is why we only keep the n-grams with a frequency equal to the number of sessions of a given service. We repeat the frequency analysis incrementing the size of the n-grams until no n-gram has a frequency equal to the number of sessions. Up to this point we have a large n-gram header that will repeat among every session. Posterior n-grams are not guaranteed to be equal for every session. Now we will show how to collect these different n-grams and treat them as one.

5.3 Service Specific Hidden Markov Models

Detecting the biggest n-gram common to all headers is only the first step. After its detection, we need to identify how sessions continue the service initialisation process. We also need to know how many different system calls exist between different headers, the length of the largest header, and the end of the header.

By extracting the largest header n-gram and the closest n-gram to the first `fork` we can identify the entire header. After an identification of the headers for every session, we proceed to isolate them.

After isolation, we concatenate every header corresponding to the same service. This concatenation will create a file containing headers for a given service. We call these concatenations n-gram families. To be able to identify the members of these families we used a probabilistic sequence identifier. The identifier is known as a hidden Markov model (HMM) and is described in detail in the book of Manning and Shütze [3]. The theory behind HMMs is well known and will not be discussed in this paper.

One way to train an HMM with multiple sequences is to concatenate them, just as we did by grouping the n-gram families. HMM training is a demanding process. In figures 2 and 3 we show training times in seconds for sessions of different sizes. In each figure a regular session and its folded counterpart are contrasted. The times needed to train an HMM are exponential depending on the number of states defined for the HMM. The bigger the number of states the better the classification, in practice the increment in performance stops at a certain number of states. There is no theoretical way to determine the number of states in advance, however empirically we found that for header n-gram families 30 states are enough to make a proper discrimination, i.e. to be able to distinguish between two different services.

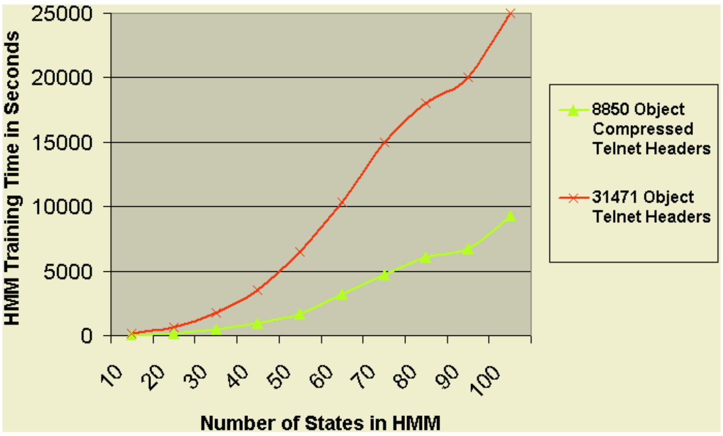


Fig. 2. Telnet Service HMM Training Times.

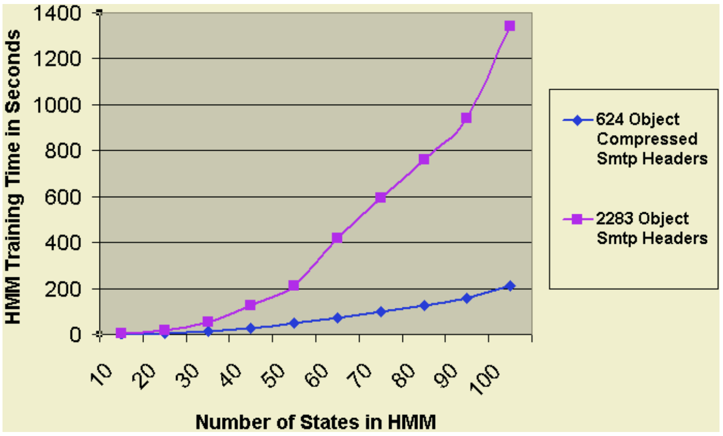


Fig. 3. Smtp Service HMM Training Times.

The times shown in figures 2 and 3 were calculated in a Pentium IV @ 2.6 GHz, 1GB of RAM, and running Mandrake© Linux 9.0.

5.4 Service Selection Validation

We generated HMMs for each of these services: `telnet`, `smtp`, `ftp`, `finger`. In the case of `telnet` and `ftp` we separated each of them in two HMMs; one for successful login and one for unsuccessful. We then tested the generated HMMs against headers from the same service and from other services. The results are summarised in table 2.

The first column in table 2 represents the HMMs for each service. The percentage is the number of correctly classified headers for the diagonal (same service discriminator and session). For an HMM of a service different from the testing session services (non diagonal elements), the percentage represents in-

Table 2. % of Correct Service Discrimination.

HMMs	`telnet` headers	`smtp` headers	`ftp` headers	`finger` headers
`telnet`	100%	2%	2%	0%
`smtp`	1%	100%	2%	100%
`ftp`	0%	1%	100%	0%
`finger`	0%	0%	0%	100%

correct classifications or false positives. The classification is a high probability if the header belongs to the HMM or low probability if it does not belong. Is worth mentioning that we monitor sessions generated at host level. When an `smtp`, `ftp`, or `finger` session is started for example from within a `telnet` session, those sessions are considered part of the `telnet` session. We only monitor services in the same machine the IDS is in. A session where the service is located in another host is monitored in that host and not locally.

6 Intrusion Detection Using Reduced Audit Files

In order to make a sound analysis of our misuse IDS we need to define our scenario. In this section we will first explain the attacks used to test our IDS. Then we will describe results obtained using HMMs with and without reduced audit files.

All the attacks used in our experimentation are present in the 1998 and 1999 DARPA repositories. These attacks are publicly available as security advisories. All of the attacks are described in detail in Kendall's thesis [21]. The attacks we used for our experiments are 9: i) eject, buffer overflow (U2R), ii) ffb, buffer overflow (U2R), iii) load-module, shell as root (U2R), iv) format, buffer overflow (U2R), v) ftp-write, R2L, vi) dictionary, R2L, vii) warez-client, unauthorised software, viii) satan, probe, and ix) ip-sweep, probe. Attacks labelled as U2R mean user to root attacks: this means the attacker is able to gain root privileges. Attacks labelled as R2L means remote to local attacks: the attacker can obtain some user privileges.

We used 20 instances of each attack to train the HMMs. The tests were conducted against entire sessions of different services, both reduced and non-reduced sessions were used. We tested against 800 `telnet`, 1000 `smtp`, 50 `ftp` and 150 `finger` sessions. All the sessions were randomly picked from the 1998 and 1999 DARPA repositories.

The results of the attack tests show that even though the detection ratio is about 97%, the false alarm ratio is also high. From every 200 attacks detected, approximately 21 were false positives. This false positive rate is very high. If we reduce the detection threshold, false positive ratio also lowers. Initially we used a 90% similarity measure, i.e., to be labelled as an attack, the tested sequence should be 90% similar to the training sequence. When increased to 95%, false positives were reduced from 21 to 13 out of 200 detections. Detection ratio was also lowered to 92%.

By using reduced sessions we obtained a 98% detection ratio and the false positives were 23 out of 200 detected attacks. With a higher similarity measure, 95%, false positives lowered from 23 to 14 and the detection ratio also lowered but only to 94%. We tested the same attacks for both scenarios. The difference in false positives was found in short attacks as `eject`. Most of the false positives were normal sessions labelled as one of these short attacks. Nevertheless, higher detection ratio is present in variations of these same short attacks. In the next section we will contrast existing audit reduction methods with ours.

7 Related Work

Lane and Brodley have also addressed the problem of object reduction [22, 23]. They have proposed two main clustering methods, one based on K-centres and the other on greedy clustering. Upon application, each cluster contains a collection of objects sequences, which is then collapsed into only two objects: the cluster centre and the associated mean. The other object sequences are simply disregarded. Lane and Brodley's methods may yield a huge reduction ratio (e.g. 500 different sequences might be shrunk to only 30 ones or even less); however, eager sequence elimination inevitably leads to poor or incomplete profiles and therefore to an increase in the false-alarm detection rate.

Apart from the two methods mentioned above, Lane and Brodley have also explored two heuristic pruning techniques: least recently used (LRU) and least frequently used (LFU). In both techniques the reduction ratio is defined *a priori* and hence a predetermined number of sequences ought to be eliminated from the input session. Both techniques will of course produce a reduction ratio as high as indicated, but at the expense of loss of chief information. Lane and Brodley report on an increment in the false-positive and false-negative ratio ($> 20\%$ and $> 16\%$ respectively)[1]. By comparison, even though our technique yields a lower reduction ratio, it does not lose any information and therefore there should be no increment in the false/positive-alarm ratio.

Wespi et al. propose an IDS along with a reduction method [20]. The reduction technique they use makes the same assumption as ours. Sequences of a certain size have a high rate of repetition. They also analyse header sequences as the `ftp login` where small variations occur from session to session. They reduce the sequences by eliminating repetitive actions and instead of replacing frequent sequences like we do they aggregate consecutive audit events. They claim that replacing a repetitive sequence with a new virtual event enriches the vocabulary but in our experimental experience we saw that the added vocabulary is minimal in contrast with the reduction rate.

Statistical methods as Hidden Markov Models (HMMs), have been successfully used for intrusion detection [24–26]. But as reported in these papers, HMMs take a large amount of time for training. Experimentally our results can help improve the times reported in the results by Forrest et al. and Ciao et al. Wagner also has a paper describing the disadvantages of using only short sequences as

[1] Notice these figures represent an increment, not the actual percentage of false-alarms.

the detection base using HMMs [27]. The size of the sequences can be greatly increased by the use of our methodology. We will now present some conclusions and future work.

8 Conclusions and Future Work

We have been able to show how two techniques, rough sets for attribute reduction and n-gram models for session folding, can be used to reduce the size of the audit files. Both reduction processes yield a information loss that is negligible. The attribute reduction method provides a 64% reduction of the original attributes. The session folding method provides a 74% reduction in session length. We also showed that the impact of the reduction on intrusion detection is negligible, with a small overhead in false positives but with an improvement on the detection rate.

By using the service discrimination module, we can scale IDSs by adding any number of services. There is added flexibility because a service only needs to keep the same header behaviour in order to be selected. The detection module also benefits from the service discrimination by reducing the search space for both misuse and anomaly detection.

As future work we need to test the actual benefit of reducing the search space for anomaly detection. Test of the system running without discrimination and with discrimination. Also new HMMs for services like `ssh`, `sftp`, and `rpc` should be added to the IDS. All the results from this research can be downloaded from our website at `http://webdia.cem.itesm.mx/ac/raulm/ids/`.

References

1. Komorowski, J., Polkowski, L., Skowron, A.: Rough sets: A tutorial. In: Rough-Fuzzy Hybridization: A New Method for Decision Making. Springer-Verlag (1998)
2. Godínez, F., Hutter, D., Monroy, R.: Attribute reduction for effective intrusion detection. In Favela, J., Menasalvas, E., Chávez, E., eds.: Proceedings of the 2004 Atlantic Web Intelligence Conference, AWIC'04. Number 3034 in Lecture Notes in Artificial Intelligence, Springer-Verlag (2004) 74–83
3. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Massachusets Institute of Technology, Cambridge, Massachusets 02142 (1999)
4. Lippman, R.P., Cunningham, R.K., Fried, D.J., Graf, I., Kendall, K.R., Webster, S.E., Zissman, M.A.: Results of the DARPA 1998 offline intrusion detection evaluation. slides presented at RAID 1999 Conference (1999)
5. Haines, J.W., Lippmann, R.P., Fried, D.J., Tran, E., Boswell, S., Zissman, M.A.: 1999 DARPA intrusion detection system evaluation: Design and procedures. Technical Report 1062, Lincoln Laboratory, Massachusetts Institute of Technology (2001)
6. Debar, H., Dacier, M., Wespi, A.: A revised taxonomy for intrusion detection systems. Technical report, Zurich Research Laboratory, IBM Research Division (1999)

7. Lee, W., Xiang, D.: Information-theoretic measures for anomaly detection. In: Preceedings of the 2001 IEEE Symposium on Security and Privacy. (2001) 130–143
8. Øhrn, A., Komorowski, J.: ROSETTA: A Rough Set Toolkit for Analysis of Data. In Wong, P., ed.: Proceedings of the Third International Joint Conference on Information Sciences. Volume 3., Durham, NC, USA, Department of Electrical and Computer Engineering, Duke University (1997) 403–407
9. Maxion, R.A., Tan, K.M.: Anomaly detection in embedded systems. IEEE Transactions on Computers **51** (2002) 108–120
10. Maxion, R.A., Tan, K.M.: Benchmarking anomaly-based detection systems. In: Proceedings of the 1st International Conference on Dependable Systems and Networks, New York, New York, USA, IEEE Computer Society Press (2000) 623–630
11. Marceau, C.: Characterizing the behavior of a program using multiple-length n-grams. In: Proceedings of the 2000 workshop on New security paradigms, ACM Press (2000) 101–110
12. Wespi, A., Dacier, M., Debar, H.: An intrusion-detection system based on the teiresias pattern-discovery algorithm. In Gattiker, U.E., Pedersen, P., Petersen, K., eds.: Proceceedings of EICAR '99. (1999)
13. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for unix processes. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, Los Alamitos, CA, IEEE Computer Society Press (1996) 120–128
14. Axelsson, S.: Aspects of the modelling and performance of intrusion detection. Department of Computer Engineering, Chalmers University of Technology (2000) Thesis for the degree of Licentiate of Engineering.
15. Denning, D.E., G., N.P.: Requirements and model for IDES-A real-time intrusion detection system. Technical report, Coputer Science Laboratory, SRI International, Menlo Park, CA (1985)
16. Hofmeyr, S., Forrest, S., Somayaji, A.: Intrusion Detection using Sequences of System Calls. Technical report, Dept. of Computer Science, University of New Mexico, Albuquerque, NM (1997)
17. Balasubramaniyan, J., Garcia-Fernandez, J., Isacoff, D., Spafford, E., Zamboni, D.: An Architecture for Intrusion Detection Using Autonomous Agents. Technical Report 98/05, Purdue University, COAST Laboratory, West Lafayete, Indiana, USA (1998)
18. Mell, P., McLarnon, M.: Mobile Agent Attack Resistant Distributed Hierarchical Intrusion Detection Systems. In: RAID Recent Advances in Intrusion Detection, Second International Workshop, West Lafayete, Indiana, USA (1999) online proceedings: http://www.raid-symposium.org/raid99/.
19. Jansen, W., Mell, P., Karygiannis, T., Marks, D.: Applying mobile agents to intrusion detection and response. NIST Interim Report (IR) 6416, NIST National Institute of Standards and Technology, Computer Security Division (1999)
20. Wespi, A., Dacier, M., Debar, H.: An intrusion-detection system based on the teiresias pattern-discovery algorithm. Technical Report RZ3103, Zurich Research Laboratory, IBM Research Division (1999)
21. Kendall, K.: A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology (1998)
22. Lane, T., Brodley, C.E.: Temporal Sequence Learning and Data Reduction for Anomaly Detection. ACM Transactions on Information and System Security **2** (1999) 295–331

23. Lane, T., Brodley, C.E.: Data Reduction Techniques for Instance-Based Learning from Human/Computer Interface Data. In: Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann (2000) 519–526
24. Warrender, C., Forrest, S., Pearlmutter, B.: Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on security and Privacy, IEEE Computer Society Press (1999)
25. Qiao, Y., Xin, X., Bin, Y., Ge, S.: Anomaly intrusion detection method based on hmm. ELECTRONIC LETTERS **38** (2002) 663–664
26. Yeung, D.Y., Ding, Y.: Host-based intrusion detection using dynamic and static behavioral models. Pattern Recognition **Vol. 36** (2003) pp. 229–243
27. Wagner, D., Soto, P.: Mimicry attacks on host based intrusion detection systems. In: Ninth ACM Conference on Computer and Communications Security, Washington, DC, USA, ACM (2002) 255–265

IDS False Alarm Filtering Using KNN Classifier

Kwok Ho Law and Lam For Kwok

Department of Computer Science,
City University of Hong Kong, Kowloon, Hong Kong
khlaw@cs.cityu.edu.hk, cslfkwok@cityu.edu.hk

Abstract. Intrusion detection is one of the important aspects in computer security. Many commercial intrusion detection systems (IDSs) are available and are widely used by organizations. However, most of them suffer from the problem of high false alarm rate, which added heavy workload to security officers who are responsible for handling the alarms. In this paper, we propose a new method to reduce the number of false alarms. We model the normal alarm patterns of IDSs and detect anomaly from incoming alarm streams using k-nearest-neighbor classifier. Preliminary experiments show that our approach successfully reduces up to 93% of false alarms generated by a famous IDS.

1 Introduction

Information security has been one of the major issues concerned by computer professions in recent years. While human daily life is more and more dependent on computers, the number of cyber crimes, as well as the impact caused by the cyber crimes, is growing in incredible rates. According to the statistics collected by the CERT Coordination Center (CERT/CC) in the United States, the number of reported incidents related to computer security in a year raised from 1,334 to 137,529 over the last decade [1]. This is the result of the rapid growth of information technology applications and it shows the importance to protect our information assets from attacks and damages.

In response to the security threats, many technologies have been developed to guard valuable information assets against unauthorized disclosures, illegal modifications and unpredicted service interruptions. Besides common protection mechanisms (e.g. cryptography, firewalls and authentication) that prevent attacks from happening, intrusion detection systems (IDSs) are invented to uncover attacks and to alert administrators for countermeasures. Intrusion detection is one of the essential elements in computer security because prompt reactions to intrusive activities can greatly reduce harm to the systems and loss due to the attacks.

1.1 Intrusion Detection

Intrusion detection is the process of monitoring computer systems or networks and searching for signs of attacks [2]. An intrusion detection system consists of sensor(s) that listen to the activities within a computer system or network and generates alarms to administrators if it finds that suspicious activities (may or may not be an intrusion) occur. Over the years, computer scientists have developed numerous techniques to detect intrusions. IDSs are also widely used as second-line defense of computer systems, as supplement to front-line defense mechanisms (e.g. authentication and firewalls).

Techniques of detecting intrusions fall into two categories, signature-based detection and anomaly-based detection. The former depends on some known attack patterns (signatures). Signature-based IDSs examine audit data (including, but not limited to, network traffic, system call logs, resources utilization, and users' activity logs) and compares it with the signatures. If there is a match, an alarm is generated to warn security officers for further investigation. Most commercial IDSs depend on signature-based detection.

Anomaly-based detection attempts to find suspicious activities occur in the target systems. It is based on the assumption that attack scenarios are rare and in some ways attacking activities possess different characteristics from normal behaviors. To find out abnormal events, normal profiles of protected systems are modeled first and are learned by the IDSs. Incoming audit data are then classified to see if it conforms to the normal model. If not, there may be attacks and alarms are raised.

Some research works have suggested combination uses of the signature-based and anomaly-based techniques [4]. The hybrid systems take advantages from both detection schemes to achieve faster speed and ability to uncover new attacks.

1.2 Problems of Intrusion Detection Systems

Although IDSs have been used for years and have demonstrated their values to organizations' security, most of them suffer from the problem of high false alarm rate and of having difficulties in fine-tuning. Practitioners often complaint that commercial off-the-shelf (COTS) IDSs trigger tons of alarms, but most alarms are actually false. The number of undesirable false alarms generated by commercial IDSs in a site can be as high as thousands per day! Identifying real alarms from huge volume of alarms is a frustrating task for security staff. Even worse, when security officers receive huge amount of false alarms everyday and treat them as a norm, they may oversee the importance of incoming alerts when real attacks occur [3]. Therefore, reducing false alarms is a critical issue in enhancing efficiency and usability of IDSs.

IDSs can be fine-tuned to suppress false alarm generation. However, it is not that easy because improper configuration may degrade security. A signature-based IDS depends on a set of rules to separate intrusive behaviors from streams of audit data. For example, a telnet connection to a UNIX machine with *root* privileges may be dangerous, so IDSs will trigger alarms when they see such kind of connection. It is obvious that the tighter the rule set, the stronger the security can be achieved. However, a tight rule set always induces more alarms, while many of them are actually not intrusive. Relaxing some rules can reduce the number of false alarms, but this action is risky, causing the IDSs unable to detect certain noteworthy incidents. The tuning problem is actually to search for a balance of reducing false alarm rate and maintaining system security.

1.3 An Overview of Our Approach

This paper reports our research that tries to reduce the number of false alarms without sacrificing security. The objectives are to reduce false alarm rate and to maintain the level of security achieved. Our approach is to let the false alarms being issued as they are and then detect any abnormal patterns from them using data mining techniques. We believe that when an attack is taking place, the alarms generated by the IDSs will have different patterns from that in an attack-free environment. Detecting these abnormal patterns can find out suspicious incidents from tones of false alarms. Those alarms which are classified as normal can be ignored. In this sense the security officers' headache, high false alarm rate and hard configuration of IDSs, can be released.

It is worth noting that our work is to reduce the number of false alarms, hence the usability of IDSs is improved. We do maintain the security level offered by intrusion detection algorithms, but we are not going to enhance it. More specifically, alerted attacks detected by the IDSs should not be filtered out by the false alarm reduction process; however, attacks that the IDSs missed will remain undetected. Enhancing detection ability of IDSs is beyond the scope of this paper.

The remaining parts of this paper will review research works on alarms handling for IDSs; propose an approach to filter false alarms; and describe experiments to evaluate our idea with results.

2 IDS Alarm Handling Using Data Mining

Mining IDS false alarms is not a new research area. Manganaris *et al.* analyzed alarm streams to find association rules [5]. He suggested an alarm handling framework that filters false alarms for IBM's network operations center, which provides real-time intrusion detection services to customers. His approach is to characterize IDS normal behavior in terms of frequent alarm sets with minimum occurrence in bursts of alarms. Then incoming alarms were classified to look for non-frequent alarm sets which are considered to be suspicious.

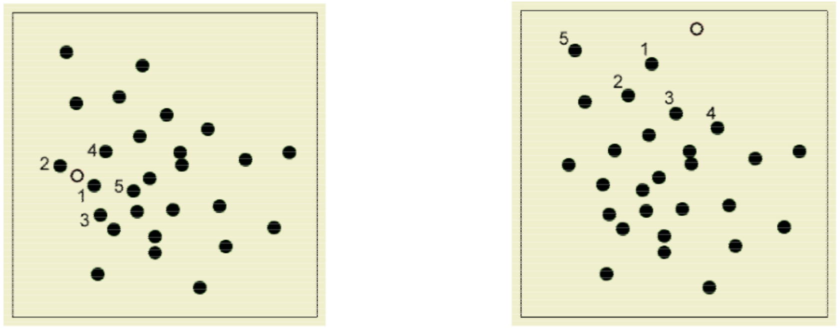
Julisch, also from IBM, proposed to find alarm clusters and generalized forms of false alarms to identify root causes [6]. He observed that 90% of false alarms are correspondent to a small number of root causes. Knowing the root causes, human expertise can adjust the IDSs regularly or remove the root causes to reduce the number of false alarms by 82% as shown in Julisch's experiments. He also mined IDS alarms for episode rules [7], which try to predict the set of alarms follows when a specific set of alarms occurs. He believed that the rules are useful because with the knowledge of such alarm patterns representing legitimate uses of the protected systems, highly similar alarms (which are supposed to be legitimate, too) can be filtered easily in the future. However, in his report, the episode rules offered only 1% of alarm reduction rate and 99% of alarms were left for manual processing.

Data mining technologies have shown their capabilities to reduce more than a half of false alarms. Our approach described in this paper further reduces the false alarm rate using KNN classifier.

3 Suggested False Alarm Filtering Using KNN Classifier

In this section, we suggest a filtering method for IDS alarms that significantly reduce false alarm load. We believe that when a network is under attack, the IDS will issue alarms in a way that is more or less different from usual situation. For example, the IDS may trigger an alarm of the type that is absent in normal situation or issue more alarms of certain types, depending on the attacks it is experiencing. Our method is to determine whether the incoming alarm sequences are deviated from normal situations. If that is the case, it may be a sign of attacks and further investigation is needed. On the other hand, if the incoming alarms possess very similar patterns as in an attack-free situation, the risk of being attacked is low. Below we will describe how to model normal alarm patterns and how to detect deviations from the normal patterns.

Given a large set of alarms generated by an IDS under an attack-free environment with N distinct alarm types, we model the normal alarm patterns with an N-dimensional space. A data point P (with N attributes $<A_1, A_2, A_3 \ldots A_N>$) in the space represents the counts of alarms of different types within a time window of size W. We define these points as "*normal*" because they are created from alarms for normal audit data experiencing no attacks. These alarms are said to be "safe" and are treated as false alarms. Now we have a set of data points telling us how the IDS behaves when there is no intrusion attempt.



(a) A normal point **(b)** An abnormal point

Fig. 1. Example of normal and abnormal points in the false alarm model. Numbered points are the 5 nearest normal (black) points from the new (white) point

To detect abnormal patterns from newly arrived alarms, new data points are created for the new alarms in the same way. The new points' distances from the normal points indicate their deviances from normal patterns. If a new point is close to the normal points, it is considered as normal too. Then we say in the time period that the point represents, there is nothing special in alarm distribution and alarms generated in that period are false alarms. Figure 1a shows a simple example of the model and a new normal point (Note that for the ease of illustration it is a 2-D example. In real situations the dimension is much larger than two). On the other hand, we consider a new point as abnormal if it satisfies any one of the following cases:

1. it lies far apart from the normal points (see Figure 1b)
2. it consists of new alarm type(s) that is absent in normal points

An abnormal point tells administrators that in the corresponding time period, the alarm distribution is strange and some intrusive activities may be happening.

We use k-nearest-neighbor (KNN) classifier to classify whether a data point is normal or abnormal. KNN classifier has been first used in intrusion detection area for anomaly detection to learn program behaviors and uncover intrusions from audit data [8]. Here we try to extend its use in false alarm reduction. The KNN classifier measures the distance between two data points P and Q by Euclidean distance (Formula 1). The distance actually represents their similarity. The shorter the distance between them, the more similar they are.

$$distance(P,Q)=\sqrt{\sum_{i=0}^{N}(p_i-q_i)^2} \tag{1}$$

where p_i and q_i are the values of the i^{th} attribute of points P and Q respectively. The final similarity score of a data point being classified is the average of its Euclidean distances from the closest k normal points. If the similarity score is higher than a threshold T, the point is said to be abnormal and the alarms corresponding to it are noteworthy (case 1 as mentioned above). On the other hand, alarms corresponding to low-score data points are false and are filtered.

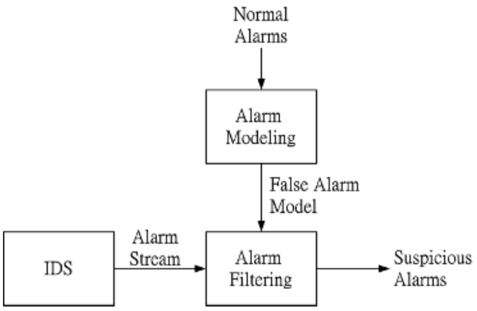


Fig. 2. Relationship between IDS and the proposed alarm reduction processes

Our suggested modeling and filtering processes are independent from the intrusion detection process (Figure 2). Therefore we believe that our model can be applied to most commercial IDSs in use nowadays without changing the existing detection configuration. The alarm modeling process makes use of normal alarms, i.e. alarms raised by IDSs under no attacks, to construct false alarm models as described earlier in this section. The normal models are used by the alarm filtering procedure in which continuously incoming alarms are filtered. Only the alarms left out needed to be investigated. The entire reduction procedure can be seen as a plug-in to IDSs.

4 Experiments

To evaluate the applicability of our approach, we conducted preliminary experiments with DARPA 1999 dataset. The DARPA Intrusion Detection Evaluation [9] program evaluated intrusion detection technologies with sufficiently large sample dataset which contains network traffic embedded with marked attacks. What we used in our experiments are TCPDUMP data collected from an Air Force Local Area Network in 1999. It was actually the captured traffic over the network. There are three weeks (5 days a week) of training data, with crafted attacks in week 2 only and no attack in week 1 and week 3.

4.1 False Alarms from Snort

We examined the network traffic with an open-source signature-based IDS, namely Snort [10]. Snort is a well-known network intrusion detection freeware available on the Internet. It looks for signs of attacks from the network traffic by comparing its own set of attack signatures and incoming packets, and generates an alarm when it finds a match. The Snort recorded each incident to database with the exact timestamp, alarm type, packet header and other relevant information.

Table 1. Top 10 frequent alarm types

Alarm Type	Occurrence
SNMP public access udp	54498
ICMP PING NMAP	7000
FTP CWD	5426
TELNET access	1799
SCAN FIN	1027
ATTACK-RESPONSES 403 Forbidden	726
TELNET login incorrect	554
WEB-MISC /doc/ access	551
WEB-CGI redirect access	542
WEB-MISC apache DOS attempt	480

We observed that the Snort, with default configuration, had issued more than 75000 alarms when processing the 3 week data, with more than 5000 per day on average. There were 76 different alarm types. For the attacks in week 2, Snort was able to detect 18 out of 43 attacks. From intrusion detection's points of view, it is not satisfactory. It may be improved with correct configurations, but improving detection ability is beyond the scope of this study. Among the alarms, we observed that 4 alarm types contributed for more than 68000 (91%) alarms. This observation matches with Julisch's one [6] as mentioned in Section 2. The 10 most frequent alarms types are listed in Table 1.

4.2 False Alarm Model

Since the DARPA 1999 dataset contains no intrusive traffic in week 1 and week 3, alarms triggered from dataset of these two weeks are false alarms. There are totally 60549 false alarms in these two weeks. The modeling of false alarm patterns is described in the previous section. A data point represents the alarm distribution of different alarm types in a time period. Since there are 76 different types of alarms, there are 76 attributes for each data point. Each attribute value equals to the count of the corresponding alarm type within the time period. The length of the time window was determined arbitrarily. Obviously, a shorter time window would tell administrators more precisely the time when a noteworthy data point got identified. Here we set the time window size to be 2 minutes. The order of magnitude of different alarm types may be different in nature and it will affect the accuracy of the classifier, so the attribute values of the points are normalized to eliminate this effect.

Another parameter to determine is the sampling rate, i.e. how frequent we construct data points with the 2-minute window size. The time interval of sampling between two successive data points should be shorter than the size of time window that a single data point represents. Otherwise, there will be gaps between data points and some alarms will be missed in the model. So we set the time interval to be 1 minute.

The data points of week 1 and week 3 alarms represent how the IDS behaves under attack-free situation and we call them normal points. If a new data point lies far apart from these normal points, we consider it as abnormal and further investigations of alarms accountable for the point are needed. Classifying data points is done by the KNN classifier.

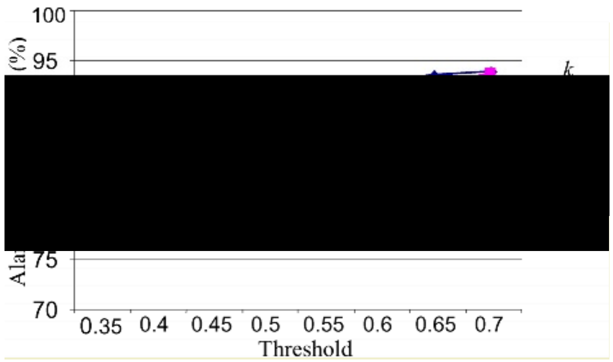
4.3 Testing the KNN Classifier

To verify our proposed method of false alarm filtering, a simple KNN classifier was implemented using Euclidean distance as the measurement of similarity between two data points. We conducted some preliminary experiments with different parameters to find an optimal setting. The two parameter values to be tested are the *k* value and the threshold *T*.

The value *k* means the number of closest normal data points taken in account when classifying a new data point. This value has an important effect on the performance of the KNN classifier and is determined empirically. We tested for different *k* values, from 5 to 20. For each value of *k*, we worked out the alarm reduction rates with different threshold *T*. The larger the value of *k*, the longer the average distance between the classifying point and the normal points. It is because more points are included. Thus the different threshold value should be tested for each value of *k*.

The second parameter *T* plays an important role when judging whether a data point is abnormal. When classifying a new data point, the Euclidean distances of the point with normal points are calculated. The average of *k* shortest distances is taken as the similarity score of the new point. If the score is larger than *T*, the new point is classified as abnormal. Obviously, a high threshold would produce less abnormal points, but security would be looser and some noteworthy alarms may be missed. On the other hand, lower threshold would offer tighter security but remain to have more false alarms. Since we believe that alarm reduction is desirable only if security level is maintained, we do not accept a too high *T* value which leads to filtering of true alerts.

The experimental results are presented in Figure 3. The results are encouraging. We found that our approach is able to reduce up to 93% of alarms without filtering out any true alarms detected by Snort. The reduction rates go up as the threshold *T* increases because more data points are classified as normal and the alarms corresponding to these points are filtered. For k=10, about 80% of the false alarms in week 2 are filtered when *T*=0.35. The rate reaches 93.8% when the threshold equals to 0.7. We did not further increase the threshold because it began to miss out some true alarms.



[redacted]

Fig. 3. False alarms filtered using KNN classifier with varying *k*

5 Conclusion and Future Work

In this paper, we have proposed a method to reduce the number of IDS false alarms. We modeled the normal alarm patterns with an N-dimensional space where each dimension corresponds to one alarm type. Each data point in the model represents alarm distribution for a time period while each of its attribute value is the number of alarms of specific type in that time period. In our study, KNN classifier is used to classify new data points into normal or abnormal based on the Euclidean distances.

To prove our idea, we carried out preliminary experiments with different k and T value combinations. The results show that over 93% of alarms were filtered without ignoring true attacks. It means that security officers can spend much less effort in handling IDS alarms.

We plan to further develop our model and apply it on live data to prove that our idea is applicable to existing commercial IDSs under real life environments. We will also focus on improving our approach to adapt to the changes in the environment.

References

1. CERT/CC Statistics 1988-2003: CERT Coordination Centre, Carnegie Mellon University, http://www.cert.org/stats/cert_stats.html
2. R. Bace: *Intrusion Detection*, Macmillan Technical Publishing, 2000.
3. K. Julisch: Dealing with False Positives in Intrusion Detection, *3rd Intl. Workshop on the Recent Advances in Intrusion Detection*, 2000.
4. A. Seleznyov and S. Puuronen: HIDSUR: a hybrid intrusion detection system based on real-time user recognition, *Proceedings of 11th International Workshop on Database and Expert Systems Applications*, 2000, pp. 41-45.
5. S. Manganaris, M. Christensen, D. Zerkle and K. Hermiz: A data mining analysis of RTID alarms, *Computer Networks*, Vol. 34, Issue 4, 2000, pp. 571-577.
6. K. Julisch: Mining Alarm Clusters to Improve Alarm Handling Efficiency, *Proceedings of the 17th Annual Conference on Computer Security Applications*, 2001, pp. 12-21.
7. K. Julisch and M. Dacier: Mining Intrusion Detection Alarms for Actionable Knowledge, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 266-375.
8. Y. Liao and V. R. Vemuri: Use of K-Nearest Neighbor classifier for intrusion detection, *Computers and Security*, Vol. 21, Issue 5, 2002, pp. 439-448.
9. DARPA Intrusion Detection Evaluation, MIT Lincoln Laboratory, http://www.ll.mit.edu/IST/ideval/
10. Snort, http://www.snort.org/

Content-Based Synchronization Using the Local Invariant Feature for Robust Watermarking

Hae-Yeoun Lee[1], Jong-Tae Kim[1], Heung-Kyu Lee[1], and Young-Ho Suh[2]

[1] Department of EECS, Korea Advanced Institute of Science and Technology,
Guseong-dong, Yuseong-gu, Daejeon, Republic of Korea
hytoiy@casaturn.kaist.ac.kr
[2] Digital Content Research Division, Electronics and Telecommunications Research Institute,
Gajeong-dong, Yuseong-gu, Daejeon, Republic of Korea

Abstract. This paper addresses the problem of content-based synchronization for robust watermarking. Synchronization is a process of extracting the location to embed and detect the signature, copyright information, so that it is crucial for the robustness of the watermarking system. In this paper, we will review representative content-based approaches and propose a new synchronization method based on the scale invariant feature transform. In content-based synchronization approaches, it is important to extract robust features even with image distortions and we suspect that the consideration of local image characteristics will be helpful. The scale invariant feature transform regards these characteristics and is invariant to noise, spatial filtering, geometric distortions, and illumination changes of the image. Through experiments, we will compare the proposed method with representative content-based approaches and show the appropriateness for robust watermarking.

1 Introduction

The rapid growth of network and computing technology has opened the ubiquitous era and digital multimedia has been widely used and accessed everywhere. However, digital multimedia can be illegally copied, manipulated, and reproduced without any protection. Digital watermarking is an efficient technique to prove the ownership by inserting copyright information into the contents itself.

Since Cox *et al.* [1] proposed a novel watermarking strategy using spread-spectrum technique, there have been many researches inspired by methods of image coding and compression. These works are robust to image noise and spatial filtering, but show severe problems to geometric distortions.

In order to counter geometric distortions, synchronization, a process of identifying the location in contents for watermark embedding and detection, is required. These techniques can be classified into four categories: (1) the use of periodical sequence, (2) the use of templates, (3) the use of invariant transforms, and (4) the use of media contents.

Kutter [2] proposed a robust synchronization approach using periodical sequence. The signature is embedded multiple times in the image at different spatial locations. The peak pattern corresponding to the location of the embedded signature is acquired by the auto-correlation function and used to restore geometric distortions by which watermarked images have undergone. Pereira and Pun [3] described a template-based approach that inserted templates into media contents. Accurate and efficient recovery of geometric transformations is possible by detecting these templates and estimating distortions. Lin and Cox [4] introduced an approach that exploited invariant properties of the Fourier transform to cyclic translation using a log-polar mapping, called as the Fourier-Mellin transform. This transform is mathematically well defined and invariant to rotation, scaling, and translation of the image. However, the severe fidelity loss during the inversion of a log-polar mapping makes it difficult to implement practically this approach. The last category is based on media contents and our approach belongs to this category. Details of content-based approaches will be explained in section 2.

In this paper, we will review previous content-based synchronization approaches and propose a new content-based synchronization method based on the scale invariant feature transform. In content-based synchronization approaches, feature extraction, or analysis, is important for the robustness of the watermarking system and we suspect that the consideration of local image characteristics will be helpful to extract features robustly even with image distortions. The scale invariant feature transform may be one of the solutions based on local image characteristics and is invariant to rotation, scaling, translation, and illumination changes of the image. Therefore, we adopt and modify this transform for the watermarking purpose. In experiments, we will compare the performance of the proposed method with that of other content-based approaches by applying various attacks such as lossy compression, spatial filtering, and geometric distortions. The results show the appropriateness of the proposed method for robust watermarking.

In the following section, we will describe previous content-based synchronization approaches. Section 3 will propose a new synchronization method based on the scale invariant feature transform. Experiment results and discussions are shown in section 4 and we will make a conclusion.

2 Previous Content-Based Synchronization Approaches

In content-based synchronization approaches, media contents represent an invariant reference for geometric transformations so that referring contents can solve the problem of synchronization, i.e. the location of the signature is not related to image coordinates, but image semantics. If we fail to detect the exact location where the signature is embedded, it is impossible or severely difficult to retrieve the signature correctly and the performance of the watermarking system will be decreased after all. Therefore, extracting the location, called as the *patch*, for watermark embedding and detection is very important and carefully designed. In this section, we will review representative content-based synchronization approaches to calculate the patch.

2.1 Bas *et al.*'s Approach

Bas *et al.* [5] proposed a feature-based synchronization approach based on salient feature points and their Delaunay tessellation. In order to formulate patches for watermark embedding and detection, they first extract feature points by applying the Harris corner detector that uses the differential features of the image. The set of extracted feature points are decomposed into a set of disjoint triangles through Delaunay tessellation. If the set of extracted feature points in the original image and distorted images is identical, Delaunay tessellation will be an efficient method to divide the image and invariant to spatial filtering and geometric distortions of the image. They use each triangle as the patch and embed the signature into the patch by applying a classical additive watermarking method on the spatial domain.

Drawbacks of this method are that the Harris corner detector is sensitive to image modifications, i.e. the set of extracted feature points is different even with small image modification and Delaunay tessellation of these points is also severely different from that of the original image. Therefore, it is difficult to robustly extract the triangle, the patch, and the robustness of the watermarking system will be eventually decreased. Furthermore, geometric manipulations that modify the relative position of feature points or remove feature points, for example aspect ratio changes and cropping of the image, may result in different tessellation and the patch do not more correspond. Fig. 1 shows the results of different tessellation in image modifications.

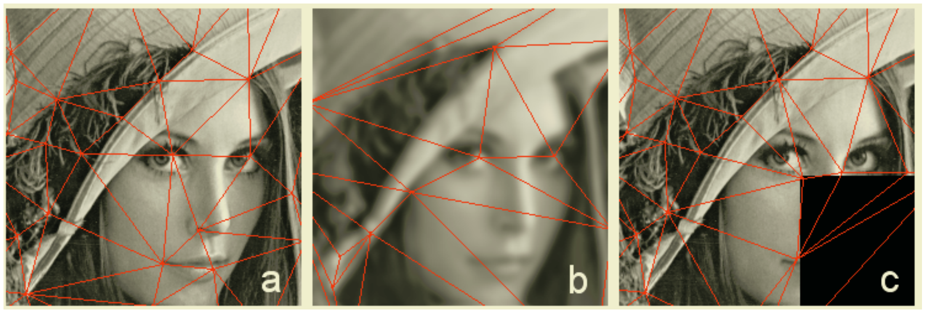


Fig. 1. Feature points and their Delaunay tessellation: (a) the original image, (b) the gaussian blurred image, and (c) the cropped image

2.2 Nikolaidis and Pitas' Approach

Nikolaidis and Pitas [6] described an image segmentation-based synchronization approach. In general, image segmentation is a useful tool in image processing and segmented regions are expected to be invariant to image noise and spatial filtering. Moreover, each region will be affected by geometric manipulations as the whole image.

In order to extract the patch, they apply an adaptive k-mean clustering technique and retrieve several largest regions. These regions are then fit by ellipsoids and their bounding rectangles are used as the patch to embed or detect the signature.

Problems with this method are that image segmentation is dependent on image contents, objects and textures, etc., and severely sensitive to image modifications that remove image parts, for example cropping and translation of the image. Fig. 2 shows the original image and its segmented regions in Baboon and Airplane images. For convenience, we represent only boundaries of segmented regions. In the Baboon image, we can easily select the largest and efficient regions for the patch, pointed by an arrow, but in the Airplane image, it is difficult to select the region for the patch.

In our experiments, for the analysis of the segmentation-based synchronization approach, we first adopt an adaptive k-mean clustering technique to segment the image, calculate the center of gravity, centroid, of segmented regions whose size is over predefined thresholds to extract robust feature points, and then these points are decomposed into triangles, patches, by Delaunay tessellation. The results with this method will be described in section 4.

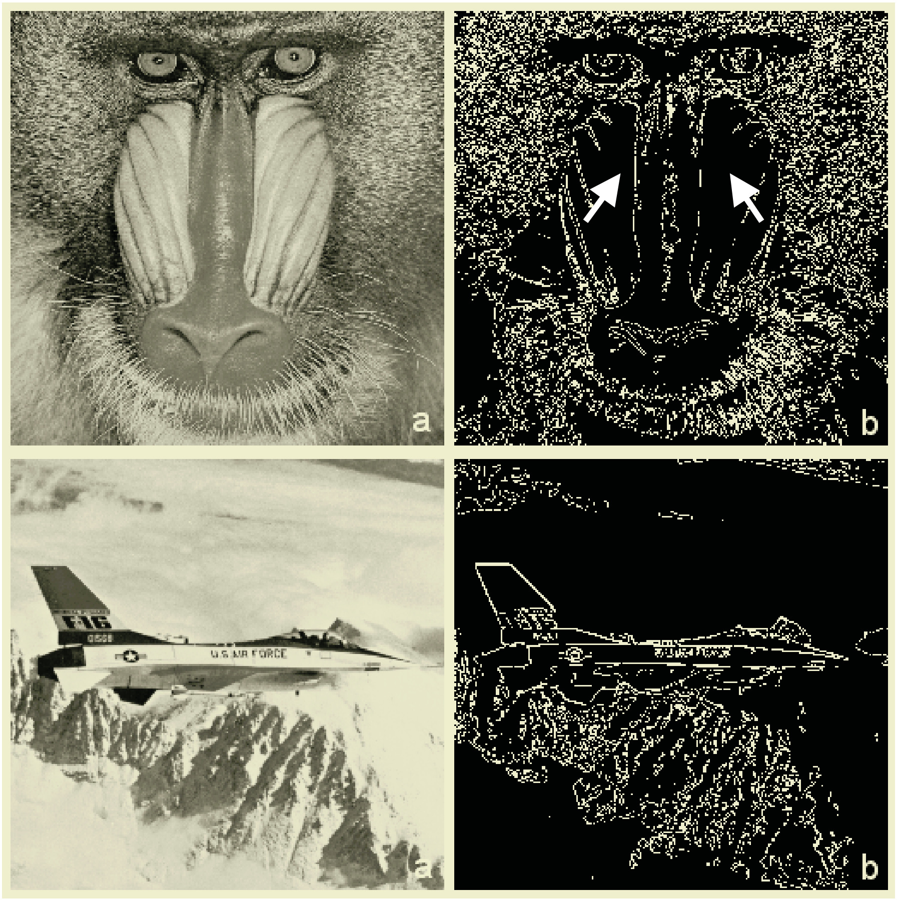


Fig. 2. (a) The original image and (b) its segmented image in Baboon and Airplane images

2.3 Tang and Hang's Approach

Tang and Hang [7] introduced a synchronization approach using the intensity-based feature extractor and image normalization. In general, the objects in the normalized image are invariant to small image modifications and this approach focuses on this fact. In order to extract feature points, they use a method called as Mexican hat wavelet scale interaction. It determines feature points by identifying the intensity change of the image and is more robust to spatial distortions. Then, the disks of the fixed radius *R*, whose center is each extracted feature point, are normalized so that they can be invariant to rotation, translation, and partly spatial filtering of the image. They use these normalized disks as patches for watermark embedding and detection.

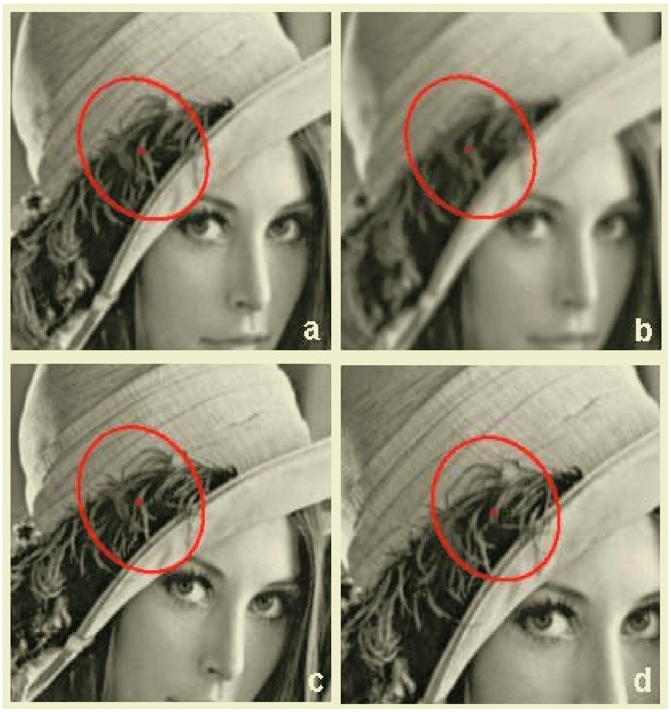


Fig. 3. The normalized disks: (a) the original image, (b) the blurred image, (c) the 10º rotated image, and (d) the 1.2× scaled image

However, they fix the radius of the disks and an image normalization technique is sensitive to image contents used for normalization so that this approach shows severe weakness to scaling distortion. In fact, it is not easy to determine the radius of the disks efficiently. Fig. 3 shows the shape of the normalized disks with image distortions. We can find that the normalized disks is robustly extracted with spatial filtering and rotation of the image, but has problems in the scaling distortion of the image, i.e. the normalized disk from the scale image is not matched with that from the original image (see Fig. 3d).

For the analysis of this approach, we first calculate the normalized disks and acquire six affine-transformation parameters. These parameters are used to formulate the normalized rectangle, the patch, for watermark embedding and detection. The results will be shown in section 4.

3 Proposed Synchronization Approach

In object recognition and image retrieval applications, affine-invariant features have been recently researched. Lowe [8] proposed a scale invariant feature transform that is based on the local maximum or minimum of the scale-space. Mikolajczyk and Schmid [9] suggested an affine-invariant interest point extractor by considering the local textures, and Tuytelaars and Gool [10] described a local image descriptor that extracted interest points and searched their near-by edges, contours, for affine-invariant regions. These affine-invariant features are highly distinctive and matched with high probability against a large case of image distortions, for example viewpoint changes, illumination changes, partial visibility, and noise of images.

In content-based synchronization approaches, the extraction of the patch is very important for the robustness of the watermarking system and we suspect that the consideration of local image characteristics will be helpful for the robust extraction of the patch. In this section, we propose a new synchronization method based on the scale invariant feature transform.

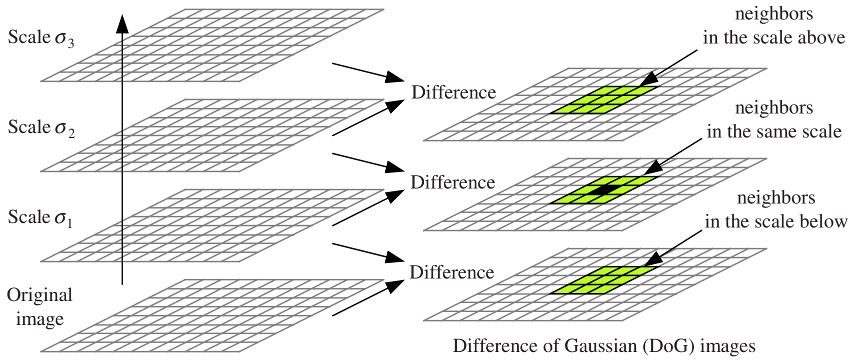

Scale σ_3
Scale σ_2
Scale σ_1
Original image
Difference
Difference
Difference
neighbors in the scale above
neighbors in the same scale
neighbors in the scale below
Difference of Gaussian (DoG) images

Fig. 4. The scale-space by using the difference of gaussian function and the closest neighbors of a pixel (filled a black color)

3.1 Scale Invariant Feature Transform

The scale invariant feature transform, called as SIFT descriptor, has been proposed by Lowe [8] and proved to be invariant to image rotation, scaling, translation, partly illumination changes, and projective transform. This descriptor extracts feature points by considering local image characteristics and describes the properties of each feature point such as the location, scale, and orientation. The basic idea of SIFT descriptor is detecting feature points efficiently through a staged filtering approach that identifies stable points in the scale-space.

SIFT descriptor can extract local feature points from following steps: (1) select candidates for feature points by searching peaks in the scale-space from a difference of gaussian (DoG) function, (2) localize feature points using the measures of their stability, (3) assign orientations based on local image properties, and (4) calculate feature descriptors which represent local shape distortions and illumination changes.

In order to extract candidate locations for feature points, they first acquire the scale-space by using a difference of gaussian function and retrieve all local maximum and minimum in the scale-space by checking eight closest neighbors in the same scale and nine neighbors in the scale above and below. These locations are invariant to scale changes of the image (see Fig. 4).

After candidate locations have been found, they perform a detail fitting to the nearby data for the location, edge response, and peak magnitude. Then, candidate points that have a low contrast or are poorly localized are removed by measuring the stability of each feature point at the location and scale using a 2 by 2 Hessian matrix, *H*, as follows.

$$Stability = \frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - D_{xy}^2} < \frac{(r+1)^2}{r}, \text{ where } H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{1}$$

The r value is the ratio between the largest and smallest eigen values and used to control the stability. In experiment, they use a value of $r = 10$.

To achieve invariance to image rotation, they assign a consistent orientation to each feature point based on local image properties and describe it relative to this orientation. In order to assign an orientation, the gradient magnitude m and orientation θ are computed by using the pixel difference as follows.

$$\begin{aligned} m &= \sqrt{(L_{x+1,y} - L_{x-1,y})^2 + (L_{x,y+1} - L_{x,y-1})^2} \\ \theta &= \tan^{-1}\left(\frac{L_{x,y+1} - L_{x,y-1}}{L_{x+1,y} - L_{x-1,y}}\right) \end{aligned} . \tag{2}$$

where L is the gaussian smoothed image with the closest scale where feature points are found. The histogram of orientations is formed from the gradient orientation at all sample points within a circular window of a feature point. Peaks in this histogram correspond to the dominant directions of each feature point.

For illumination condition invariance, they define 8 orientation planes, make the gradient magnitude m and orientation θ smooth by applying a gaussian filter, and then sample over a 4 by 4 grids of locations with 8 orientation planes. This feature vector, 4x4x8 elements, is normalized by dividing the square root of the sum of squared components to reduce the effect of illumination changes.

Local feature points obtained through SIFT descriptor are invariant to rotation, scaling, translation, and partly illumination changes of the image.

3.2 Modification for the Watermarking Purpose

The number and distribution of local feature points from SIFT descriptor is dependent on image contents and textures. Moreover, SIFT descriptor is originally devised for image matching applications so that it extracts many feature points densely distributed to over the whole image. In order to use this local invariant descriptor for the watermarking purpose, we adjust the number, distribution and scale of feature points and remove points whose possibility to be detected, matched, with image distortions is low through experiments. Finally, the patch for watermark embedding and detection is formulated using this descriptor.

SIFT descriptor represents the properties of each feature point such as its location (t_1, t_2), scale (s), and orientation (θ), etc. Therefore, for watermark embedding and detection, we can make the patch that is invariant to rotation, scaling, and translation of the image by the following affine-transform equation. Through this transformation, we can convert the signature whose shape is a rectangle into the shape of patches in the image or vice-versa.

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = s\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x_o \\ y_o \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}. \tag{3}$$

Images from natural scenes have many noise factors that affect feature extraction and we can decrease the interference of noise by applying gaussian filtering before feature extraction.

In order to control the distribution of local feature points, we apply a circular neighborhood constraint used by Bas *et al.* [5]. The neighborhood size D is dependent on the image dimension and quantized by the r value as follows.

$$D = \frac{width + height}{r}. \tag{4}$$

The *width* and *height* represent the width and height of the image, respectively. The r value is a constant that control the size and set as 24 similar to Bas *et al.* However the value from the difference of gaussian function is used to measure the strength of each feature point. The neighborhood size must be carefully designed. If the size is small, the patch will be severely overlapped and if the size is large, the number of the patch will not be enough.

SIFT descriptor considers image contents and hence the shape of the patch is rotated and scaled dependently on image contents (see Fig. 7d). For embedding and detection of the signature into the patch, interpolation is necessarily required to transform the rectangular signature to be matched with the shape of the patch or vice-versa. In order to minimize the distortion of the signature through interpolation, the size of the patch must be set near to that of the rectangular signature. For adjusting the size of local features, we divide the scale of feature points into the range and apply magnification factors determined experimentally on the assumption that the size of watermarked images will not be excessively changed.

The scale of feature points from SIFT descriptor is also related to the scale factor of a gaussian function in the scale-space. In our analysis, feature points whose scale is small have the low probability to be detected because they are easily disappeared when image contents are modified. Local feature points whose scale is large also have the low probability to be detected in distorted images because their locations easily move to other locations. Moreover, it means overlapping with other patches and it will degrade the perceptual quality of the image when the signature is inserted. Therefore, we remove feature points whose scale is below 2 or over 10, these values are experimentally determined.

Fig. 5 shows the patch from our proposed method for watermark embedding and detection. For convenience, we represent only one patch. We can find that the patch is formulated robustly even with spatial filtering, rotation, and scaling of the image.

4 Experiment Results and Discussions

In this section, we will compare the performance of the proposed method with that of three representative content-based synchronization approaches described in section 2. Method 1 is an approach proposed by Bas *et al.* [5], method 2 is a segmentation-based approach similar to Nikolaidis and Pitas' approach [6], and method 3 is an approach described by Tang and Hang [7]. For all methods, we applied a circular neighborhood constraint of Bas *et al.* [5] to obtain the homogeneous distribution of feature points.

For experiments, we used five 512 by 512 pixel images: Lena, Baboon, Pepper, Airplane, and Lake images widely used in image processing applications (see Fig. 6). Each image is distorted by applying spatial filtering attacks such as mean filtering, median filtering, gaussian noise, and JPEG compression and geometric distortions such as rotation, scaling, translation, and cropping of the image.

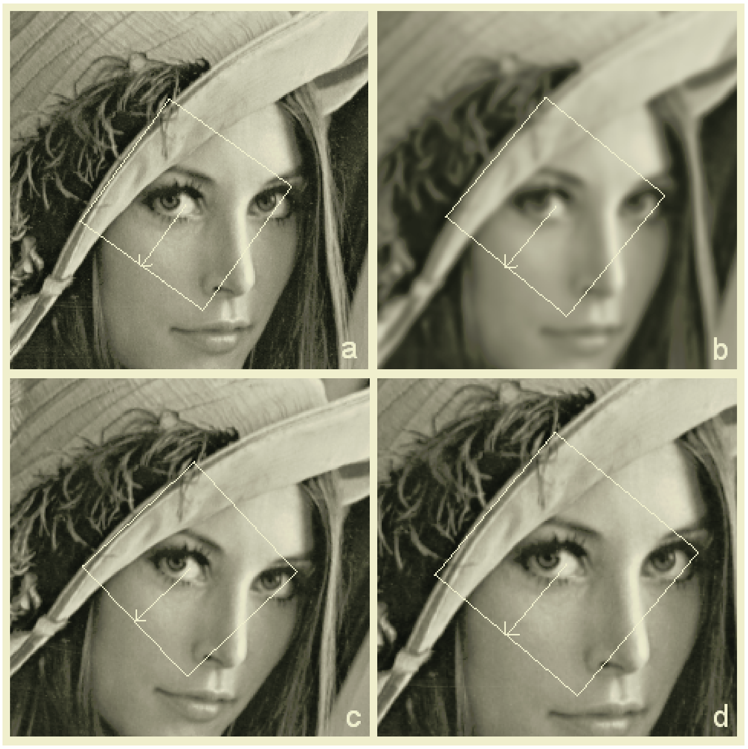


Fig. 5. The affine-invariant patch from the scale invariant feature transform: (a) the original image, (b) the blurred image, (c) the 10° rotated image, and (d) the 1.2× scaled image (the arrow represents the scale and orientation of the feature point)

The robustness of the patch is measured by matching the patch from the original image with those from attacked images. If the pixel difference between the location of the patch from the original image and that from attacked images is less than two pixels, we considered it as the correctly matched patch. These small miss-alignments can be compensated by searching some pixels around the location of the patch originally found when we retrieve the embedded signature, prove the ownership, from the watermarked patch. In particular, if images are attacked by geometric distortions, we transform the coordinates of the patch from attacked images into the coordinates of the original image by calculating their inverse transform.

Table 1 shows experiment results. The data in an "Original image" row is the number of the patch extracted from the original image. The data in other rows is the number of matching patches between the patch from the original image and those from attacked images. The root mean square (RMS) errors of the pixel difference, miss-alignments of matching patches, are represented in parenthesis (in pixels). Each item in table is the averaged value from five images.

Fig. 6. (a) the Lena image, (b) the Baboon image, (c) the Pepper image, (d) the Airplane image, and (e) the Lake image

Table 1. The number of matching patches from the original image and attacked images. RMS errors of the pixel difference are represented in parenthesis

	Method 1	Method 2	Method 3	Proposed method
Original image	45.5	57	31	43.6
Mean 3×3	17.8 (0.412)	14.8 (0.569)	30.2 (0.234)	25.6 (0.664)
Median 3×3	19.2 (0.379)	12.4 (0.664)	21.8 (0.491)	25.4 (0.664)
80% JPEG compression	25.6 (0.294)	21.4 (0.442)	26.2 (0.557)	33.0 (0.457)
Gaussian noise	16.6 (0.369)	16.2 (0.546)	22.4 (0.658)	21 (0.748)
Rotation (10°)	11.4 (0.441)	0 (0.000)	5.6 (0.683)	15.8 (0.661)
Scaling (1.2×)	12.8 (0.502)	13 (0.673)	0 (0.000)	5.2 (1.325)
Translation (30×30)	24 (0.000)	0 (0.000)	7.6 (0.443)	31.2 (0.311)
Cropping (1/4)	27.6 (0.000)	0.6 (0.067)	8.2 (0.387)	31.6 (0.143)

Figure 7 shows the shape of patches by each content-based synchronization approach in the Pepper image. The shape of patches from method 1 and method 2 is a triangle and the shape of patches from method 3 and our method is a rectangle. The background image (Fig. 7b) of method 2 is the boundary of segmented regions. The background image (Fig. 7c) of method 3 represents the residual image between two images scaled by different factors in Mexican hat wavelet scale interaction.

As mentioned in section 2, method 1 is a synchronization approach based on feature points by the Harris corner detector and their Delaunay tessellation. The Harris corner detector is considerably sensitive to small image modifications and the triangles of Delaunay tessellation from these feature points do not more correspond. Therefore, the results with method 1 show that the number of matching patches is decreased in attacks. Especially, in cropping and translation distortions of images, although the intensity of images is not modified, the matching patches, triangles, from Delaunay tessellation are severely different. The results with this method however were overall acceptable for the watermarking purpose. In general, the watermarking system can prove the ownership of contents if it can retrieve the embedded signature correctly from at least one patch.

Method 2 is a synchronization approach using image segmentation and Delaunay tessellation. Although this approach shows high performance in image scaling, the performance with other attacks was poorer than other approaches. Moreover, when images were cropped, rotated, and translated, the center of gravity of segmented regions easily moved to other locations and hence it was very difficult to formulate the patch robustly.

Method 3 is based on an intensity-based feature extractor called as Mexican hat wavelet scale interaction and image normalization. As explained in section 2, the objects in the normalized image are invariant to small image modifications and rotation. The results with this method show considerable robustness to spatial filtering attacks than other approaches. However the performance with geometric attacks is relatively low and especially when images are scaled, they failed to extract the matching patches.

The overall performance of the proposed method is satisfactory. Our method can extract the patch more robust than method 1 in spatial filtering, translation, and cropping attacks because SIFT descriptor only considers local image characteristics and is invariant to illumination changes of the image. Our method is also more robust than method 3 in geometric distortions. We expect that the watermarking system using the proposed synchronization method will be resilient to spatial filtering and geometric distortions of the image.

However, our method shows relatively a low performance in scaling distortion but it is acceptable for the watermarking purpose because the watermarking system can prove the ownerships of contents if it can retrieve the embedded signature correctly from at least one patch. If we can retrieve the signature from more patches, the reliability and robustness of the watermarking system can be increased. Our on-going research is focused on increasing the robustness in scaling distortion and we expect to acquire more high performance.

5 Conclusion and Future Works

Synchronization is a process of identifying the location to embed and detect the signature and crucial for the robustness of the watermarking system. The content-based

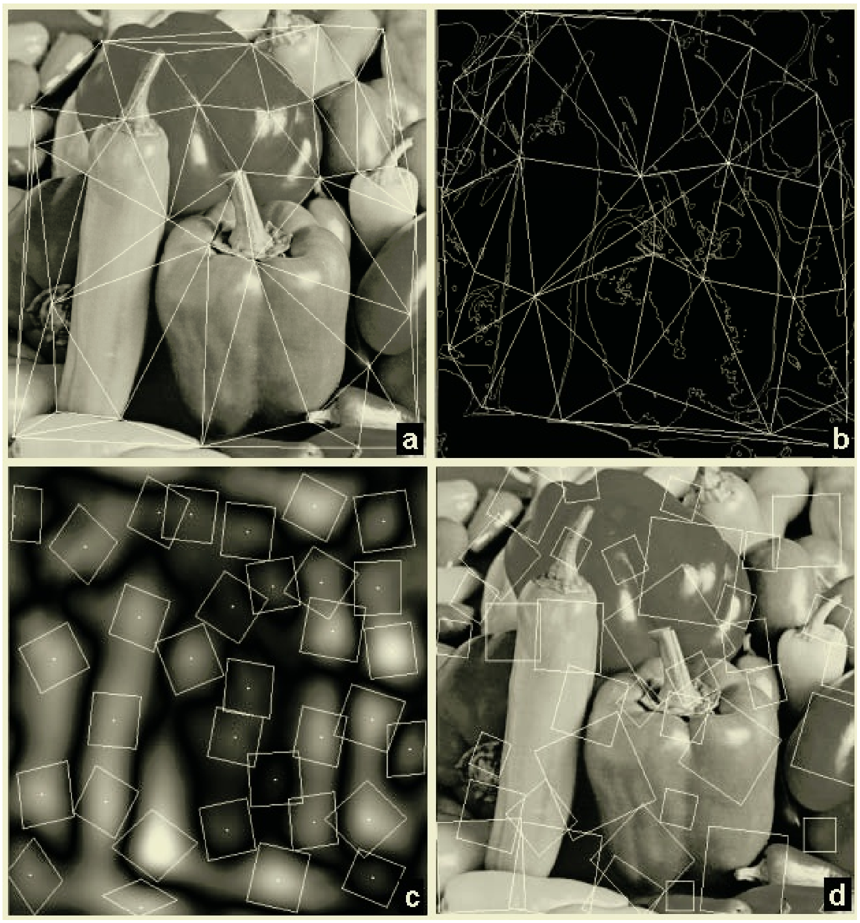


Fig. 7. The shape of patches by each content-based synchronization approach. (a) Method 1, (b) Method 2, (c) Method 3, and (4) Proposed method

synchronization approach is one of the solutions and our on-going research is focused on this approach. In this paper, we reviewed representative content-based synchronization approaches and proposed a new synchronization approach based on the scale invariant feature transform, called as SIFT descriptor. We suspect that the consideration of local image characteristics will be helpful to extract features with robustness. The scale invariant feature transform considers these local image properties and is invariant to rotation, scaling, translation, and illumination changes of the image. We modified this descriptor for the watermarking purpose.

In experiments, we applied various image distortions, attacks, such as spatial filtering and geometric distortions and compared the performance of our approach with that of representative content-based synchronization approaches. The results support that the consideration of local invariant features is helpful in designing a robust watermarking system and our synchronization approach is one of the efficient methods to solve the problem of synchronization. Our future research is focused on increasing the robustness with geometric attacks and applying the patch for watermark embedding and detection. We believe that our watermarking approach using this local invariant feature will be also robust.

References

1. I.J. Cox, J. Kilian, T. Shamoon: Secure spread spectrum watermarking for multimedia. IEEE Trans. on Image Processing, Vol. 6 (1997) 1673-1678
2. M. Kutter: Watermarking resisting to translation, rotation and scaling. Proc. of SPIE, Vol. 3528 (1998) 423-431
3. S. Pereira, T. Pun: Robust template matching for affine resistant image watermark. IEEE Trans. on Image Processing, Vol. 9 (2000) 1123-1129
4. C. Lin, I.J. Cox: Rotation, scale and translation resilient watermarking for images. IEEE Trans. on Image Processing, Vol. 10 (2001) 767-782
5. P. Bas, J-M. Chassery, B. Macq: Geometrically invariant watermarking using feature points. IEEE Trans. on Image Processing, Vol. 11 (2002) 1014-1028
6. A. Nikolaidis, I. Pitas: Region-based image watermarking. IEEE Trans. on Image Processing, Vol. 10 (2001) 1726-1740
7. C.W. Tang, H-M. Hang: A feature-based robust digital image watermarking scheme. IEEE Trans. on Signal Processing, Vol. 51 (2003) 950-959
8. D.G. Lowe: Object recognition from local scale-invariant features. Proc. of International Conference on Computer Vision, (1999) 1150-1157
9. K. Mikolajczyk, C. Schmid: An affine invariant interest point detector. Proc. of European Conference on Computer Vision, (2002) 128-142
10. T. Tuytelaars, L.V. Gool: Wide baseline stereo matching based on local affinely invariant regions. Proc. of British Machine Vision Conference, (2000) 412-422

Some Fitting of Naive Bayesian Spam Filtering for Japanese Environment

Manabu Iwanaga[1], Toshihiro Tabata[2], and Kouichi Sakurai[2]

[1] Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan
`iwanaga@itslab.csce.kyushu-u.ac.jp`

[2] Faculty of Information Science and Electrical Engineering, Kyushu University, Japan
`{tabata,sakurai}@csce.kyushu-u.ac.jp`
`http://itslab.csce.kyushu-u.ac.jp/`

Abstract. Bayesian filtering is one of the most famous anti-spam measures. However, there is no standard implementation for treatment of Japanese emails by Bayesian filtering. In this paper, we compare several conceivable ways to treat Japanese emails about tokenizing and corpus separation. In addition, we give experimental results and some knowledge obtained by the experiments.

1 Introduction

Recent years, spam is rapidly increasing, because email is much cheaper method to send out some information than other advertising methods such as direct mail or telemarketing. Nowadays spam becomes a terrible obstacle to email communication, hence it is important for users and postmasters to keep spam out of their maildrop.

In order to keep spam out of maildrop, not only users but Internet Service Providers (ISPs) and Mail User Agents (MUAs) have applied anti-spam features into their products and services. For example, Mozilla [1], Eudora [2] and Outlook [3] have applied spam-detection as standard features.

One of the most famous anti-spam measures is Bayesian filtering. It has become famous in the past several years. Many implementations of Bayesian filtering have been developed, motivated by and based on Graham's essay [4]. Since these implementations work alone as a proxy or via other program like procmail[5], they can be applied easily. However, there is no standard implementation to process Japanese emails in Bayesian filtering.

Spam causes a social issue also in Japan (In the case of Japan, spam sent to email address assigned to cell phone is a big problem). Some Japanese email users receive spams written in Japanese, others receive spams written in English, and some unfortunate users receive both of them. For example, authors exchange both Japanese and English emails with an identical email address, and also receive both Japanese and English spams by the address.

In this paper, we consider and make experiments in several conceivable points to treat Japanese emails. First, we focus on how to extract tokens from Japanese sentence. It is important to extract tokens, because the probability that received email is a spam is calculated from probabilities for tokens, that the email with the token is a spam. Second, we consider how to separate corpuses according to language written in email. While most of Japanese implementations of Bayesian filtering separate corpuses between Japanese emails and non-Japanese emails, choosing corpus for each token, not for each email, will bring more accuracy. A part of the result may be useful for users who treat both English emails and non-English emails, not only Japanese.

Corpus is a collection of words appeared in previous spams and nonspams. In detail, it contains two data:

– number of learned spams and nonspams, respectively,
– frequency of each word in spams and nonspams, respectively.

From above two data, each word is given a probability that an email which contains the word is a spam.

2 Related Work

In order to avoid spam, various methods have been proposed and used. Most of proposed methods are classified into following.

– Watching behavior of SMTP connection (e.g. Greylisting [6])
– IP Blacklisting (e.g. ORDB [7])
– Domain authentication (e.g. SPF [8], senderID [9])
– Content filtering: rule-based (e.g. Spamassassin [10]), statistical (e.g. naive Bayes [4, 11, 18–20], Markovian [12]), collaborative (e.g. Razor [13])
– Challenge/response (e.g. [14–17])

Watching behavior of SMTP connection, IP Blacklisting, and domain authentication are mainly used in mail servers. It is pretty hard for users to apply these methods on their PCs, because a mail server can omit some information of sender and a user cannot intervene in transaction when it delivers to users' maildrop (For example, a user cannot apply greylisting without control of his/her mail server). On the other hand, users can easily apply content filtering and challenge/response, without modifying servers and SMTP (needless to say, these methods can work on a server for convenience of users.)

Nowadays, statistical filtering methods have been applied broadly to avoid spam, especially Bayesian filtering (naive Bayes) which has been popular since Graham's essay [4] came out. Bayesian filter calculates the probability for every word that a randomly chosen email containing the word will be a spam, according to past spams and nonspams. Furthermore, Bayesian filter can add tokens appeared in the email to its corpus, according to judgment of the Bayesian filter itself. Therefore, a user only has to train his/her filter in case his/her filter makes a mistake. Even if spammers use obfuscated words, Bayesian filter also learns these words and obfuscated words are used as obvious evidence. There are many available implementations, for example bsfilter [18], scbayes [19], bogofilter [20] and popfile [21].

3 Consideration in Bayesian Filtering for Japanese Environment

As Bayesian filtering is used around the world, it is also used in Japanese environment, which treats both Japanese email and English email. In order to apply Bayesian filtering in Japanese environment, some modification on the filtering scheme for improving efficiency of Bayesian filtering should be considered.

There are many Bayesian filtering implementations intended to be used in Japanese environment. These implementations have several features specialized for Japanese environment. We focus on these features, especially about extracting tokens from a sentence and separating corpus according to language written in an email.

3.1 Method for Extracting Tokens

Because Japanese does not have a blank to separate a sentence into tokens, it is not so easy to extract grammatical words from a sentence. In Japanese, the most popular way to extract tokens until now is variants of bigram which utilize the cayegory of Japanese characters (hiragana, katakana and kanji). Basically, bigram is a method that all pairs of consecutive two characters are extracted as tokens. In fact, bigram is usually adapted with some modification, because kanji, hiragana and katakana are used with different way in Japanese. It is also a simple and easy way to implement that regarding each adjacent characters in same category (kanji, hiragana, and katakana) as a token. Bigram seems fairly effective for Bayesian filtering because Japanese has many phrases composed of two, three or four kanjis.

If a user sticks to grammatical word, he/she can also use external tools like KAKASI [22], ChaSen [23] or MeCab [24]. While these tools are originally intended for morphological analysis or kanji-to-hiragana conversion, tokenizing is now one of the most well-known usages of these tools.

Now we give several samples. Bsfilter [18], one of Bayesian filtering implementations written in ruby, has following rules to extract tokens from a Japanese email.

– For a sequence of kanjies, adjoining two kanjies is respectively extracted as a token. If kanji stands alone or continues only two characters, the whole is extracted as one token.
– All contiguous katakanas are extracted as one token as the whole.
– Hiraganas are not extracted as any token.

Scbayes [19], another implementation written in Scheme[1], has following rules.

[1] Scheme is one of varieties of the Lisp programming language.

– Basically, contiguous two characters are extracted as a token.
– However, combination of hiragana and kanji in that order is ignored.

Some implementations like bogofilter [20] do not support extracting tokens from Japanese sentences, so some tool must convert Japanese email to tokenized text which an implementation can extract tokens from. These tokenizing methods are depicted as Figure 1.

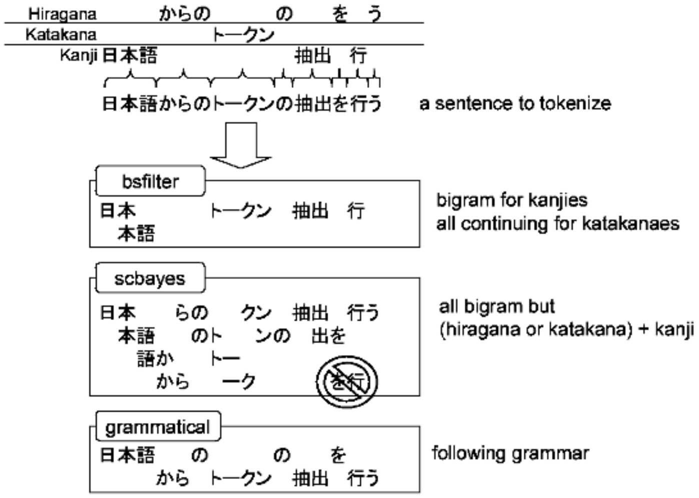


Fig. 1. Three tokenizing methods.

3.2 Method of Separating Corpus

Relative proportions of spams and nonspams are not same between in one's Japanese emails and in one's English emails. For example, one of the authors' email addresses receives many Japanese spams but only a few English spams, while another receives many English spams but a few Japanese spams. It is thought that the former address was gathered by and exchanged among Japanese spammers, and the latter one by/among spammers in English-speaking market. On the other hand, we can receive important English emails by the latter email address, for example, acceptance letter from the international conference and confirmation of hotel reservation. This situation causes high spam probabilities for whole words of particular language. It is undesirable that this bias causes more false-positives (FP) in emails of particular language, because Bayesian filtering is based on the assumption that we do not want a classification to be affected by the relative proportion of spams and nonspams.

To cope with this situation, most of Japanese implementations separate corpuses for Japanese emails and non-Japanese emails. In these implementations, filters first make a decision whether an email is written in Japanese or not. That is to say, the implementation distinguishes language written in an email, and

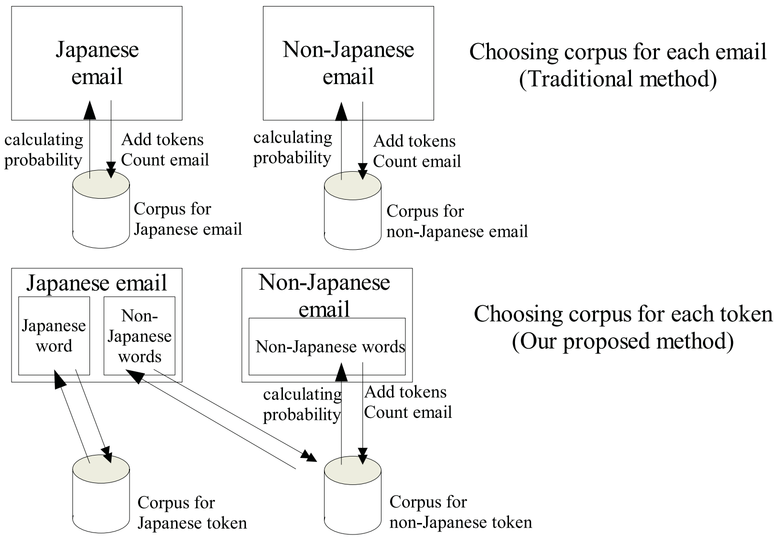

Japanese email
Non-Japanese email
Choosing corpus for each email (Traditional method)
calculating probability
Add tokens Count email
Corpus for Japanese email
Corpus for non-Japanese email
Japanese email
Japanese word
Non-Japanese words
Non-Japanese email
Non-Japanese words
Choosing corpus for each token (Our proposed method)
Corpus for Japanese token
Corpus for non-Japanese token

Fig. 2. Our Proposal for separating corpus.

then a spam probability is calculated with one corpus corresponding to the language. It is reasonable to reduce false-positives arising from language in which email is written. It is easy to distinguish Japanese emails from non-Japanese emails, reading "charset" header field [25] or checking character code.

However, we wanted to detect language-independent evidence, which is contained in header, etc. To realize this, we tried to distinguish language of each token, not each email. This method is expected to reduce both false-positives and false-negatives (FN). This method can also deal with an email which contains two or more languages as a message body.

Our method is represented as Figure 2. Upper one shows traditional separation for each email and lower one shows our proposed separation for each token. In our method, non-Japanese tokens in Japanese emails are learned by a corpus for non-Japanese tokens. On the other hand, corpus should also count number of learned spams and nonspams. Because tokens in Japanese emails are learned by either Japanese and non-Japanese corpuses, Japanese email should be counted by both of the corpuses. In this experiment, we leave this matter open and use simple way. We count an email as both Japanese and English email, in proportion to ratio between Japanese tokens and English tokens. For example, an email which contains 300 Japanese tokens and 100 English tokens is counted as 0.75 Japanese emails and 0.25 English emails.

4 Experiment

We performed experiments on techniques concerning about treating Japanese emails mentioned above. In every experiment, emails were divided randomly into ones for training and ones for testing. We call the proportion of emails for training on all emails in the experiment initial training ratio. For example, we have 1000 nonspams and 600 spams, and initial training ratio is 1/5, we train 200 nonspams and 120 spams. We changed initial training ratio to several values to know characteristic of our method associated with amount of training. At first, emails for training are learned as spams and nonspams by a Bayesian filter, respectively. Then emails for testing are classified to spam or nonspam by the filter, and we count numbers of false-negatives and false-positives. Because corpuses are cleared at the end of each testing, each experiment is independent.

Performance is evaluated by rate of false-positives and it of false-negatives. The former is an error that a nonspam is wrongly classified as a spam, and the latter is an error that a spam is wrongly classified as a nonspam.

In the following experiments, we used bsfilter (Revision 1.35.4.3, [18]) as an implementation of Bayesian filtering, with Graham's method of calculation and 0.9 as threshold, and then simulated the other methods when needed. It is because we want to look at core differences between methods and omit the other differences.

4.1 Method for Extracting Tokens

We performed experiment with three tokenizing methods, bsfilter-styled bigram, scbayes-styled bigram (simulated on bsfilter) and grammatical tokenizing. We used ChaSen [23] as a grammatical tokenizer.

The number of emails used for this experiment is shown in Table 1. Condition 1 assumes that there are spams as many as nonspams, and Condition 2 assumes that there are more nonspams than spams. The emails are collected from spams / nonspams we received and spams we caught by a honeypot email account. Note that emails used in this experiment are all Japanese emails, because tokenizing methods are intended only for Japanese.

Table 2 shows the result of experiment between tokenizing methods. In case there are spams as many as nonspams (Condition 1), there were about 0.34% of false-positives and 1.70% of false-negatives on an average, and difference between tokenizing methods was little (Roughly speaking, difference between tokenizing methods is only ±0.07% of false-positives and ±0.41% of false-negatives from average). In this case, difference between tokenizing methods was little.

In case there are more nonspams than spams, false-positive decreased and false-negative increased (Condition 2) from above case, and difference of false-negatives between tokenizing methods was not negligible (difference of false-

Table 1. Number of emails which is used to experiment about extracting tokens.

initial training	nonspam	spam
	Japanese	Japanese
Condition 1	293	293
Condition 2	1659	293

Table 2. Comparison between token-extracting methods.

initial training	method	Condition 1		Condition 2	
		FP	FN	FP	FN
3/4	**bsfilter**	0.41%	1.42%	0.00%	6.69%
	ChaSen	0.27%	1.94%	0.00%	8.72%
	scbayes	0.27%	1.42%	0.00%	15.88%
1/2	**bsfilter**	0.27%	1.39%	0.02%	11.22%
	ChaSen	0.27%	1.94%	0.02%	14.18%
	scbayes	0.34%	2.11%	0.00%	23.64%

negative reached about at ±5% from average.) Bsfilter-styled bigram had the best performance with 6.69% and 11.22% of false-negatives; ChaSen had second-best with 8.71% and 14.18%. However, scbayes-styled bigram made the most errors, about twice as many as bsfilter. From this result, we can say that grammatical tokenizing does not always yield good performance in Japanese.

4.2 Method of Separating Corpus

Next, we performed experiments to compare between two methods of separating corpus. One is a way that selects a corpus for each email, and the other is a way that selects corpus for each token. The number of emails used for this experiment is shown in Table 3. Condition 3 represents a case that there are spams as many as nonspams, and Condition 4 represents a case that there are more nonspams than spams. Similar to the experiment in Section 4.1, the emails are collected from spams / nonspams we received and spams we caught by a honeypot email account. However, this time we included non-Japanese spams / nonspams.

Table 4 shows the result of comparison between corpus separation styles. Through Condition 3 and 4, the result shows that our method, the way that selects a corpus for each token, got less false-positives with almost same amount of false-negatives. We think it is because spam has more characteristics which are language-independent than nonspam.

Because false-positive is serious problem on using Bayesian filtering, it is significant that our method can decrease false-positives without increase of false-negatives.

Table 3. Number of emails for comparison about corpus separation.

Init. training	nonspam			spam		
	JA	non-JA	Total	JA	non-JA	Total
Condition 3	1167	55	1222	293	929	1222
Condition 4	1680	257	1937	75	164	239

JA: Japanese emails, non-JA: non-Japanese emails.

Table 4. Experiment about separating corpus.

Number of emails	Initial training	**for each email** FP	**for each email** FN	**for each token** FP	**for each token** FN
Condition 3	4/5	0.99%	3.84%	0.48%	3.86%
	3/5	0.97%	4.35%	0.55%	4.28%
	2/5	1.45%	5.26%	0.81%	5.49%
	1/5	2.29%	7.10%	1.46%	6.84%
Condition 4	4/5	0.32%	4.59%	0.24%	4.75%
	3/5	0.37%	6.09%	0.23%	5.42%
	2/5	0.47%	7.39%	0.25%	7.87%
	1/5	1.03%	11.68%	0.24%	12.63%

5 Conclusion

In this paper, the authors have explained issues on Bayesian filtering for Japanese. To adapt Bayesian filtering into Japanese email environment, some modification is usually made to Bayesian filtering. The authors have looked at two factors and experimented on several methods which have applied by existing implementation or which the authors have proposed.

First, we have focused on methods of extracting tokens from sentence. Grammatical tokenizer does not always yield the best performance for Japanese emails.

Second, we have focused on methods of separating language-specified corpuses. False-positives can be decreased by choosing language-specified corpus for each token, not for each email, without increase of false-negatives. Because false-positive is serious problem on using Bayesian filtering, it is significant that choosing corpus for each token can decrease false-positives without increasing false-negatives.

While the authors have obtained good result by choosing corpus for each token, modification for counting email learned by Bayesian filter is very intuitive. More sophisticated way for counting emails may yield better performance than our way.

Acknowledgement

This research was partly supported from the grant of Secom Science and Technology Fundation.

References

1. Mozilla 1.3 Release Notes, modified February 2004, http://www.mozilla.org/releases/mozilla1.3/.
2. QUALCOMM Releases Eudora(R) 6.0 - Significant Version Upgrade with New Advanced Time-Saving Tools, September 2003, http://www.eudora.com/press/2003/09_04_03.html.

3. Help Prevent Junk E-Mail Messages with Outlook 2003, April 2003, http://www.microsoft.com/office/editions/prodinfo/junkmail.mspx.
4. P. Graham, "A Plan for Spam," http://paulgraham.com/spam.html.
5. Procmail, http://www.procmail.org/.
6. E. Harris, The Next Step in the Spam Control War: Greylisting, 2003, http://projects.puremagic.com/greylisting/.
7. Open relay database, http://www.ordb.org/.
8. Sender Policy Framework, http://spf.pobox.com/.
9. MTA Authentication Records in DNS, *Internet-Draft*, May 2004, http://xml.coverpages.org/draft-ietf-marid-core-01.txt.
10. Spamassassin, http://spamassassin.org/.
11. P. Graham, Better Bayesian Filtering, *Spam conference*, Boston, USA, January 2003, http://paulgraham.com/better.html.
12. W. Yerazunis, The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It., *2004 Spam Conference*, Boston, USA, January 2004, http://crm114.sourceforge.net/Plateau_Paper.pdf
13. Vipul Razor, http://razor.sourceforge.net/.
14. E. Gabber, M. Jakobsson, Y. Matias, A. Mayer, Curbing Junk Email via secure Classification, *Financial Cryptography '98*, Anguilla, British West Indies, 1998, 198–213.
15. R. J. Hall, Channels: Avoiding unwanted electronic mail, *the 1996 DIMACS Symposium on Network Threats*, Piscataway, USA, 1996, 85–103.
16. Mailblocks, http://about.mailblocks.com/.
17. M. Jakobsson, J. Linn, J. Algesheimer, "How to Protect Against a Militant Spammer," *Cryptology ePrint archive*, report 2003/071, 2003.
18. bsfilter, http://www.h2.dion.ne.jp/˜nabeken/bsfilter/.
19. scbayes, http://www.shiro.dreamhost.com/scheme/wiliki/wiliki.cgi?Gauche%3A SpamFilter&l=jp.
20. bogofilter, http://bogofilter.sourceforge.net/.
21. POPFile, http://popfile.sourceforge.net/.
22. KAKASI, http://kakasi.namazu.org/.
23. ChaSen, http://chasen.aist-nara.ac.jp/.
24. MeCab, http://cl.aist-nara.ac.jp/˜taku-ku/software/mecab/.
25. Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples, *RFC2049*, Nov. 1996, http://www.ietf.org/rfc/rfc2049.txt

Efficient Authenticated Key Agreement Protocol for Dynamic Groups

Kui Ren[1], Hyunrok Lee[1], Kwangjo Kim[1], and Taewhan Yoo[2]

[1] IRIS, Information and Communications University, Daejon, Korea 305-714
{kren, tank, kkj}@icu.ac.kr

[2] Electronics and Telecommunications Research Institute, Daejon, Korea 305-350
twyoo@etri.re.kr

Abstract. Group key management presents a fundamental challenge in secure dynamic group communications. In this paper, we propose an efficient group authenticated key agreement protocol (EGAKA), which is designed to be fully distributed and fault-tolerant, provides efficient dynamic group membership management, mutual authentication among group members and is secure against both passive and active attacks. The features of EGAKA are as follows: Firstly, EGAKA can be built on any general two-party key exchange protocol without relying on a particular one. EGAKA achieves scalability and robustness in heterogenous environments by allowing members to use any available two-party protocol in common and deliberately designed fault-tolerant mechanism in dynamic membership management. Secondly, EGAKA provides extremely efficient member join services in terms of both communication and computation costs which are constant to the group size. This is a very useful property in the scenarios with frequent member addition.

1 Introduction

In recent years, more and more applications rely on peer-to-peer group communications. Examples include teleconferences, replicated servers, command and control systems, and communications in *ad hoc* networks. Providing ubiquitous and reliable security services is very important in these environments and is considered as an open research challenge [2, 23]. The basic requirement for secure group communications is the availability of a common secret group key among members. Therefore, key management, as the corner stone of most other security services, is of the primary security concern. Key management schemes can be classified into two flavors: centralized key distribution and distributed key agreement. Key distribution protocols aren't suitable for dynamic peer groups, because of many inherent drawbacks and limitations [13]. Many key agreement protocols in the open literature are the extensions of two-party Diffie-Hellman (DH) key exchange protocol [2–6, 10, 13, 16, 18, 20, 21, 25, 26, 29], except for some recently proposed protocols based on Weil pairing [17, 22]. All these protocols fall into two different categories: One deals with static groups; while the other deals with dynamic groups.

1.1 Related Works

In this subsection we summarize related works on group key agreement protocol. Most group key agreement protocols are based on generalizations of the two-party DH key exchange protocol. These protocols usually rely on certificates not only to perform entity authentication but also to resist active attacks such as man-in-the-middle attack, under the assumption of the deployment of public key infrastructure. But this may not be always true in dynamic groups formed in the heterogeneous environments.

The following protocols focus mainly on efficiency in terms of computation and communication costs. Burmester *et al.* [9] proposed a protocol which takes only two rounds and three modular exponentiations per member to generate a group key. However, the communication cost is significant, requiring $2n$ (n: group size) broadcast messages, and this protocol is only secure against passive attacks. Steiner *et al.* [25] addressed dynamic membership issues in the developing of Group Diffie-Hellman (GDH) protocol. GDH protocol is fairly computation-intensive, requiring $\mathcal{O}(n)$ exponentiations, but bandwidth efficient. A-GDH and SA-GDH were proposed by Ateniese *et al.* [2] based on GDH. Two protocols are, however, proved to be vulnerable to a number of potential attacks [19]. The computation and communication costs of both protocols are high, each requiring n rounds and $\mathcal{O}(n^2)$ exponentiations. TGDH, a tree-based key agreement protocol proposed by Kim *et al.* [13], is another modified version of GDH. TGDH combines a binary tree structure with the GDH technique and is efficient in terms of computation as most membership changes require $\mathcal{O}(\log n)$ exponentiations. Note that key establishment and authentication issues are not explicitly discussed in TGDH. Another protocol by Yang *et al.* [29] is an ID-based authenticated group key agreement protocol. However, dynamic membership management in this protocol is not clear. Key agreement based on group shared password can be found in [1]. There are also some three-party key agreement protocols based on Weil pairing [17, 22]. Many other protocols focus mainly on the security itself. These protocols are typically of high inefficiency. The protocol proposed by Bresson *et al.* [4–6] is the first provably secure one. It is based on GDH protocols by adding authentication function. The entity authentication is done via signatures on all the message to frustrate active attacks. Katz *et al.* [11] proposed another provably secure protocol, which is based on Burmester's protocol by introducing signature operations for authentication. Some conference key establishment protocols with security proofs can be found in [3, 26].

This paper proposes an efficient group authenticated key agreement protocol (EGAKA). Except for common functionalities, EGAKA distinguishes itself from other existing protocols as follows: Firstly, EGAKA can be built on any two-party protocols without relying on a particular one. Therefore, EGAKA achieves scalability and robustness in heterogenous environments by allowing members to use any available two-party protocol in common. Secondly, EGAKA provides extremely efficient member join service in terms of both communication and computation costs which are constant regardless of the group size. This is a very useful property in the scenarios with frequent member addition. The remainder of the paper is organized as follows. We brief the notation, necessary terminology, and some primitives in Section 2. Then in Section 3, we discuss the goals and assumptions of EGAKA. This is followed by the description of EGAKA in Section 4. In Section 5, we compare the complexity of EGAKA with those of other proposed protocols. Security issues of EGAKA is then discussed in Section 6. Finally, the conclusion is given in Section 7.

2 Notation and Primitives

The notation as used throughout the paper is shown below:

$\{\cdot\}_K$	symmetric encryption algorithm using key K
$h(\cdot)$	one way hash function
K_G	group secret key
S_{ij}	shared secret by M_i and M_j, e.g. $\alpha^{x_i x_j}$
K_{ij}	peer-to-peer session key between M_i and M_j
B_{ij}	blinded S_{ij}, *i.e.*, $B_{ij} = h(S_{ij})$
d	height of a key tree
$M_{\hat{i}}$	M_i's sibling in the key tree
N_{lj}	tree node j at level l
E_i	the partners set of M_i

We also use the following definitions and cryptographic primitives:

Key Tree is used in the past for centralized group key distribution systems. The logical key hierarchy (LKH) method [27, 28] is the first approach. Almost all the later group key management protocols adopt such kind of binary key tree structure because of its inherent efficiency. TGDH and ELK are such examples [13, 20, 21]. The structure of key tree used in EGAKA is depicted in Figure 1. There are 3 types of nodes: root node, leaf node and interior node. A leaf node is also called an isolated leaf node, if his sibling is an interior node. For example, N_{22} is such a node. Each leaf node is associated with a group member. Every node in the key tree has a key pair: a secret key and the corresponding blinded key. The secret key is shared only by the members whose corresponding nodes belong to the subtree (if any) rooted in this node and thus for secure subgroup communication. For example, the left node at level 1 has a secret key K_{135} and a blinded key $B_{135} = h(K_{135})$, and K_{135} is shared only by M_1, M_3 and M_5. The blinded key is for group key computing. How to securely and efficiently assign the appropriate subset of these intermediate keys to each group member is always the most challenging problem in the protocol design. Note that in EGAKA, neither secret key nor the blinded key is transmitted in plaintext.

The group key is computed as: $K_G = K_{123456} = h(B_{135}||B_{246}); B_{135} = h(K_{135}) = h(h(B_{15}||B_3)); B_{246} = h(K_{246}) = h(h(B_{24}||B_6)); B_{15} = h(K_{15}); B_3 = h(K_3); B_{26} = h(K_{26}); B_4 = h(K_4)$, where $||$ denotes message concatenation. In the later description, we do not distinguish between group member and its corresponding leaf node. To simplify our subsequent description, we use the term *key-path*, denoted as KP_i^*, which is a set of nodes along the path of M_i from itself to the root node (except for the root node). We also use the term *co-path* as defined in [13], denoted as CP_i^*, which is the set of siblings of each node in the key path of M_i. For example, the KP_5^* and CP_5^* of member M_5 are the two sets of nodes $\{N_{32}, N_{21}, N_{11}\}$ and $\{N_{31}, N_{22}, N_{12}\}$, respectively. The cardinalities of both CP_i^* and KP_i^* depend on M_i's position in the key tree and equal to its level. For M_5 at level 3, the cardinalities of KP_2^* and CP_2^* are both 3. Therefore, every member derives the group key from all the blinded keys of its co-path nodes and its own secret share. A *partner* of M_i is defined as the member who shares a peer-to-peer session key with M_i. We use E_i denote the partners set of M_i. In Figure 1, E_1 consists of M_2, M_3 and M_5.

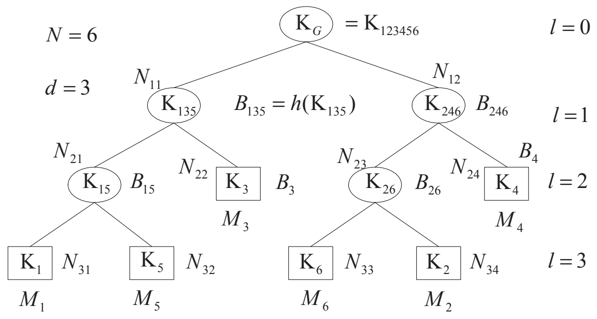


Fig. 1. Notation for key tree.

Two-party authenticated key agreement protocol of any kind can be used in EGAKA, if only it provides explicit key authentication and entity authentication, perfect forward secrecy, resistance to known-key attacks. A typical protocol is the one proposed by Ateniese *et al.*, which is a provable secure two-party generalized DH authenticated key exchange protocol (A-DH) [2]. The security of A-DH is directly based on the well known two-party Decisional Diffie-Hellman (DDH) problem [25]. The A-DH is a two-round protocol and provides implicit key authentication and entity authentication without requiring a *priori* knowledge of the long term public key of the parties involved. And the certificates can be piggy-backed onto existing protocol messages [2]. By adding some additional key confirmation messages, A-DH can provide explicit key authentication instead of implicit key authentication. Other qualified protocols include password based two-party key agreements protocols such as AMP *etc.* [12, 14, 15].

3 Goals and Assumptions of EGAKA

In the design of EGAKA, we bear the following goals in mind. Firstly, EGAKA should provide flexible and efficient member join/leave services in terms of communication and computation costs. In particular, we emphasize on the scenarios with frequent member additions. Such scenarios include many multicast applications. In member leave service, we focus on fault-tolerant property to achieve robustness. Secondly, EGAKA should provide entity authentication. Every member should be authenticated when joins the group, and thus frustrates masquerading and eavesdropping. The trust model in EGAKA is that any single current member can authenticate the new members and accept them. This is assumed because we do not consider insider attacks as our focus is on the secrecy of group keys and the integrity of group membership. The latter means the inability to spoof authenticated membership. Consequently, insider attack is not relevant in this context since a malicious insider can always reveal the group key or its own private key, thus allowing for fraudulent membership. Thus, by definition, a new member is said to authenticated only if it was authenticated at least once by any current group member. Thirdly, EGAKA should be resistant to known-key attack, while providing forward secrecy, backward secrecy and key independence [2, 13, 21]. In the design of EGAKA, we also address an important principle: the protocol should be fully distributed, which means no centralized KDC should be involved during both key establish and key update processes and the secret keying information should only be generated, computed and transmitted by group member itself. The existence of centralized third party violates the nature of key agreement protocol and is also impractical in many scenarios [2, 13].

We assume the size of dynamic peer groups to be less than 200 (empirically), because large groups are likely to have very frequent membership changes and much diluted trust. The former will cause lots of overhead and the latter negates the need for contributory group key agreement. In dynamic groups, groups are usually formed on-the-fly, and therefore, members tend to have different deployments of security primitives. And different primitives demand different assumptions. For example, in order to resist man-in-the-middle-attack, both of two parties in DH key exchange protocol must have certificates issued by some CA to certify their public key; while in password-based key exchange protocols, shared password must exist between the two parties. In order to adapt to these heterogenous environments, EGAKA is designed to work with any two-party authenticated key agreement protocol in common among groups members, that is, group members can choose any desired two-party protocol available to use by negotiation (*e.g.,* either DH protocol or password based key agreement protocol, *etc.*); group key can then be established contributorily based on the chosen protocol. Thus, the robustness and flexibility is achieved in EGAKA.

4 EGAKA Protocol

EGAKA consists of two basic sub-protocol suites: key establishment protocol (EGAKA-KE) and key update protocol (EGAKA-KU).

4.1 EGAKA-KE

EGAKA-KE includes two phases: Phase I is to complete group entity authentication by applying any chosen two-party authenticated key agreement protocol; Phase II is the group key generation process.

Phase I: Entity Authentication. In Phase I, group members first negotiate the two-party authenticated key agreement protocol and the key tree structure which are to be used in the consequent part of the protocol. This can be done by simply using explicit message broadcast among members. In these messages, group members can randomly poll which member to generate the key tree structure and agree on the chosen two-party protocol. The poll-chosen member then broadcasts the tree structure to all group members; hence, every group member could determine his own position in the key tree. In order to form the binary key tree structure and facilitate the following group key computing process, some members must perform authentication with up to d partners by applying the chosen two-party protocol. For example, in Figure 3, M_1's partners are M_2, M_3 and M_5.

The binary tree generating process can be as follows: Two members are first randomly chosen to join the key tree and are supposed to authenticate each other and form one interior node. Another two members are then chosen to join the current key tree and are supposed to perform entity authentication with the current two members, respectively, and forms another two interior nodes. This process repeats till the last member is chosen. An example is given in Figure 2. Obviously, the number of partners for any specific group member ranges from 1 to d. Note that the two-party authenticated key agreement protocol executes exactly $n-1$ times.

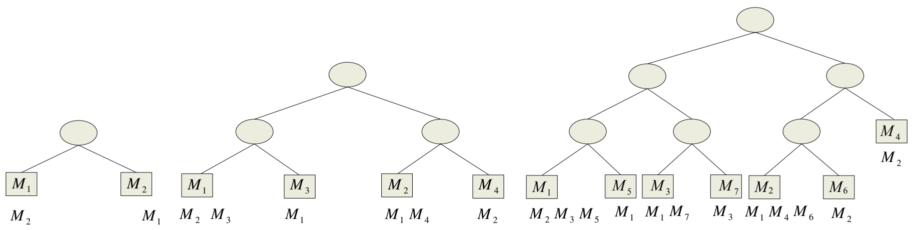


Fig. 2. Key tree structure generating process: an example.

As mentioned above, the entity authentication is achieved by applying the chosen two-party protocol among group members and the number of partners for each member is according to his position in the key tree. On obtaining the tree structure, all group members perform entity authentication with their assigned partners simultaneously. Therefore, the two-party authenticated key agreement protocol is simultaneously executed $n-1$ times. The round number is exactly

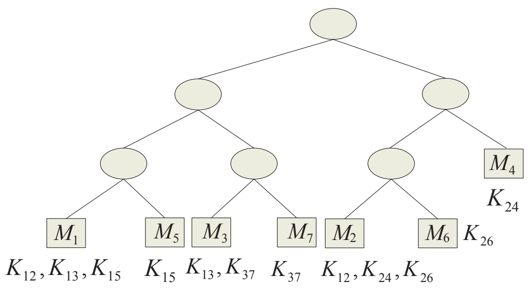


Fig. 3. Results of Phase I.

that of the underlying two-party protocol. A set of peer-to-peer session keys are therefore established as the execution results. Without loss of generality, we choose A-DH as an example and show the precise procedure in Appendix. Note that no peer-to-peer session key confirmation round is executed in Phase I; hence only implicit peer-to-peer session key authentication and entity authentication is provided. For the key structure in Figure 2, the execution results are depicted in Figure 3.

Thus, at the end of Phase I, all group members are implicitly authenticated and a set of peer-to-peer session keys are established among members. The established peer-to-peer session keys not only assure the efficient and secure transmission of keying information in protocol Phase II (*i.e.* act as key encryption key (KEK)), but also (some of them) act as secret key shares of the group members according to the key tree.

Phase II: Group Key Generation. EGAKA-KE Phase II consists of d rounds. Every group member computes one more the secret key along its key-path each round and finally computes the group key after d rounds. All intermediate keying information is encrypted by symmetric cipher using peer-to-peer session keys established in Phase I. Therefore, all the session keys are confirmed and each group member assures its partners' aliveness. (This is important, because no assurance of aliveness can be exploited by many attacks [19].) The protocol operates as follows in Figure 4.

Figure 5 gives an example of Phase II. In round 1, each member first computes the key and blinded key of its key-path node at level 2. Then M_1 sends the keying information $\{B_{15}||M_1\}_{K_{13}}$ to M_3, because B_{15}'s corresponding node belongs to M_3's co-path and M_1 and M_3 share a peer-to-peer session key K_{13}. The same routing is followed by other members. Note that the member ID is included in the message to strength the authentication. Therefore, at the end of round 1, M_1 obtains B_{37}; M_2 obtains B_4; M_3 obtains B_{15}; M_4 obtains B_{26}. In round 2, every member first computes the key and blinded key corresponding to the node in its key-path one-level-up. So M_1 computes K_{1357} and B_{1357}; M_2 computes K_{246} and B_{246}; M_3 computes K_{1357} and B_{1357}; M_4 computes K_{246} and B_{246}. Again each member sends out the keying informa-

Protocol EGAKA-KE Phase II :

Let $\{M_1, \cdots, M_n\}$ be the set of group members. For M_i, let $\{N_{1i}, N_{2i}, ..., N_{li}\}$ and $\{N_{1\hat{i}}, N_{2\hat{i}}, ..., N_{l\hat{i}}\}$ denote the members of KP_i^* and CP_i^* , respectively.

Round 1 :

M_i, $i \in [1,n]$ computes: $K_{N_{(d-1)i}} = K_{i\hat{i}}$, if M_i is at d, $K_{N_{(d-1)i}} = K_i$, if M_i is at $d-1$, and $B_{N_{(d-1)i}} = h(K_{N_{(d-1)i}})$, where K_i is M_i's secret share (a random nonce).

M_i, $i \in [1,n] \longrightarrow \{M_j \mid M_j \in E_i, N_{(d-1)i} \in CP_j^*\}$: $\{B_{N_{(d-1)i}} || M_i\}_{K_{ij}}$.

Round r $(2 \le r < d)$:

M_i, $i \in [1,n]$ decrypts the received message(s) and obtains $B_{N_{l\hat{i}}}$ of $N_{l\hat{i}} \in CP_i^*$. and computes the key pair of $N_{(l-1)i} \in KP_i^*$: $B_{N_{(l-1)i}}$ and $K_{N_{(l-1)i}}$.

M_i, $i \in [1,n] \longrightarrow \{M_j \mid N_{l\hat{i}} \in CP_j^*\}$ (if any): $\{B_{N_{l\hat{i}}} || M_i\}_{K_{N_{li}}}$.

M_i, $i \in [1,n] \longrightarrow \{M_j \mid M_j \in E_i, N_{(l-1)i} \in CP_j^*\}$ (if any):$\{B_{N_{(l-1)i}} || M_i\}_{K_{ij}}$.

Round d :

M_i, $i \in [1,n]$ decrypts the received messages. M_1 and M_2 obtain the blinded key of their co-path nodes at level 1, respectively. Other members obtain the blinded key of their co-path nodes at level 2, respectively.

$M_1 \longrightarrow \{M_j \mid N_{12} \in CP_j^*\}$: $\{B_{N_{12}} || M_1\}_{K_{N_{11}}}$;

$M_2 \longrightarrow \{M_j \mid N_{11} \in CP_j^*\}$: $\{B_{N_{11}} || M_2\}_{K_{N_{12}}}$.

Upon receiving the above message, each group member computes the group key: $K_G = h(B_{N_{11}} || B_{N_{12}})$.

Fig. 4. Protocol EGAKA-KE Phase II.

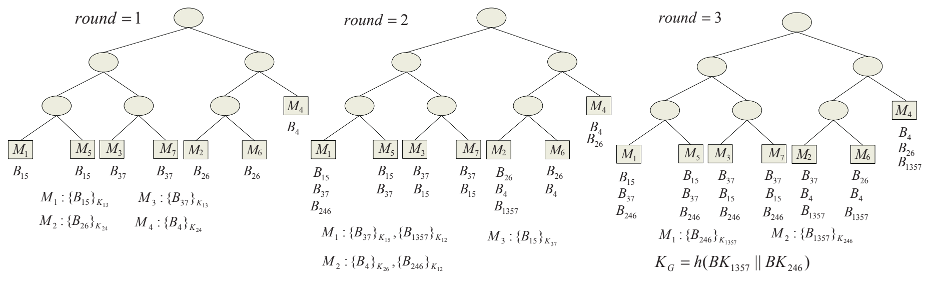


Fig. 5. An example of key establishment process.

tion. Therefore, M_1 multicasts $\{\{B_{37}||M_1\}_{K_{15}}, \{B_{1357}||M_1\}_{K_{12}}\}$; M_2 multicasts $\{\{B_4||M_2\}_{K_{26}}, \{B_{246}||M_2\}_{K_{12}}\}$; M_3 unicasts $\{B_{15}||M_3\}_{K_{37}}$. Thus, at the end of round 2, M_1 gets B_{246}; M_2 gets B_{1357}; M_4 gets B_{26}; M_5 gets B_{37}; M_6 gets B_4; M_7 gets B_{15}. In round 3, M_1 and M_2 multicast the following message: $\{B_{246}||M_1\}_{K_{1357}}$, $\{B_{1357}||M_2\}_{K_{246}}$, respectively. Upon receiving this message, every member now can independently compute the group key as $K_G = h(B_{246}||B_{1357})$.

4.2 EGAKA-KU

In order to accommodate frequently group membership changing, key agreement protocol in dynamic groups should be flexible and fault-tolerant, and provide efficient group re-keying process. To make our protocol concrete, throughout this section we use A-DH [2] as the chosen underlying two-party protocol. In A-DH, the following additional notation is used:

p, q	large prime integers, $q \mid \phi(p)$
G	unique subgroup of Z_p^* of order q
α	exponential base
x_i, α^{x_i}	long-term secret/public key pair of M_i
r_i	M_i's secret nonce
S_{ij}	shared secret by M_i and M_j, *e. g.*, $\alpha^{x_i x_j}$

Member Join Protocol. Again assume there are n members $(M_1, ..., M_n)$ in the current group and a new member M_{n+1} wants to join the group. M_{n+1} first broadcasts a joining request message. The message also includes his available two-party authenticated key agreement protocols in hand. Upon receiving this message, a sponsor M_s at level l is chosen that is responsible for authenticating M_{n+1} and the group key updating. M_s is chosen according to the following rule: Choose an isolated leaf node if any, and the shallowest and leftmost one is the first choice; if no such node, the shallowest and leftmost leaf node is chosen.

Next, M_s creates a new interior node and a new leaf node, and promotes the new interior node to be the parent of both new member node and himself. Then M_s and M_{n+1} execute the chosen two-party authenticated key agreement protocol and establish a fresh peer-to-peer session key $K_{(n+1)s}$. M_s then updated all the key pairs of its key-path. (Of course, if M_s is not an *isolated leaf node*, then M_s must first obtain the updated blinded key from its sibling $M_{\hat{s}}$ before updating all the key pairs.) Then M_s divides the whole group into l subgroups according to its co-path nodes. Members from each subtree rooted in the co-path node of M_s form a subgroup. Clearly, the members of each subgroup only need to update the blinded key corresponding to the sibling node of their subgroup root node. M_s thus encrypts the according keying information for each subgroup and multicasts them together. The joining protocol is depicted in Figure 6.

An example is shown in Figure 7. The new member M_6 wants to join the group, so he broadcasts a join request together with a fresh α^{r_6}. Then M_3 is the sponsor according to the key tree. So M_3 first creates a new interior node and a new leaf node for M_6, and promotes the new interior node to be the parent of both M_6 and himself. After that, M_6 computes: $\alpha^{r_3 S_{s(n+1)}}, K_{36}, B_{36}, B_{1356}$ $(S_{s(n+1)} = \alpha^{x_6 x_3})$ and divides the group into three subgroups. Then M_6 broadcasts: $\{B_{1356}||M_3\}_{K_G}, \{B_{36}||M_3\}_{K_{135}}, \{B_{15}||B_{24}||M_3\}_{K_{36}}, \alpha^{r_3 S_{s(n+1)}}$. On receiving the message, all group members can independently update the new group key. It takes only 2 rounds to finish key updating process.

Member Leave Protocol. Fault-tolerance property is our main focus in the design of member leave protocol. Again we assume there are n $(n > 2)$ members in the current group and M_x is going to leave the group. The sponsor M_s is chosen as before. In order to provide forward secrecy, the leaving member is prohibited to know the new group key afterwards. Thus, current members cease

Member Join Protocol :

Let M_{n+1} be the new member, M_s be the sponsor at level l, and $M_{\hat{s}}$ be the sibling member of M_s if any.

Round 1:

M_{n+1} broadcasts: $\alpha^{r_{n+1}}||Join||Protocol\_choice$.

Round 2: (if any)

$M_{\hat{s}}$ unicasts M_s: $\{B_{s\hat{s}}||M_{\hat{s}}\}_{K_{s\hat{s}}}$.

Round 3 (2):

M_s computes the new key pair :

$$K_{s(n+1)} = \alpha^{r_s r_{n+1}},\ B_{s(n+1)} = h(K_{s(n+1)}).$$

M_s updates all blinded keys of its key-path: $B_{1s}, B_{2s}, ..., B_{(l-1)s}$.

M_s broadcasts: $\{B_{1s}||M_s\}_{K_G}, \{B_{2s}||M_s\}_{K_{1s}}, ..., \{B_{(l-1)s}||M_s\}_{K_{(l-2)s}}$,
$\{B_x, N_x \in CP_s^*\}_{K_{s(n+1)}}, \alpha^{r_s S_{s(n+1)}}$.

M_i updates the new group key K_G', $i \in [1, n+1]$.

Fig. 6. Member join protocol.

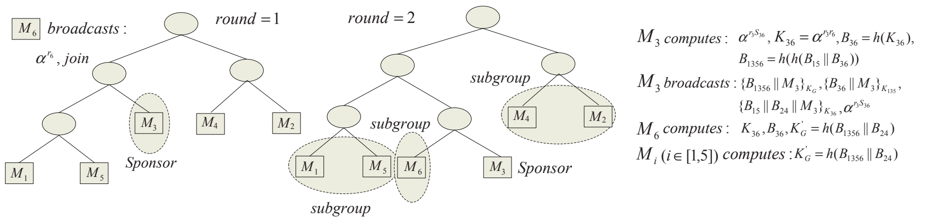

M_6 broadcasts :
α^{r_6}, join
round = 1
M_3
M_4
M_2
M_1
M_5
Sponsor
round = 2
subgroup
M_4
M_2
subgroup
M_1
M_5
M_6
M_3
Sponsor
subgroup
M_3 computes : $\alpha^{r_3 S_{36}}, K_{36} = \alpha^{r_3 r_6}, B_{36} = h(K_{36})$,
$B_{1356} = h(h(B_{15} \| B_{36}))$
M_3 broadcasts : $\{B_{1356} \| M_3\}_{K_G}, \{B_{36} \| M_3\}_{K_{135}}$,
$\{B_{15} \| B_{24} \| M_3\}_{K_{36}}, \alpha^{r_3 S_{36}}$
M_6 computes : $K_{36}, B_{36}, K_G' = h(B_{1356} \| B_{24})$
M_i $(i \in [1,5])$ computes : $K_G' = h(B_{1356} \| B_{24})$

Fig. 7. An example about member join process.

to use any secret key known by M_x right after M_x left the group and delete all the peer-to-peer session keys shared with M_x. So fault-tolerance is most important because the current member cannot use the old group key anymore. Any delay caused by the disability of single current member (*e.g.,* temporary power failure, short-term drop line, out of hop range due to mobility, *etc.*) in update the group key will slow down the group key update process. And this causes group communications in trouble. Our member leave protocol solves this problem by working with any available subgroup member without relying on a particular one.

Suppose M_s is at level l. M_s first updates the key tree structure and its own secret share and all the key pairs of its key-path. (M_s may be required to perform a two-party authenticated key agreement protocol, if he is not an *isolated leaf node.*) Then M_s divides the whole group into $l-1$ subgroups according to its co-path nodes following the same rule as in the member join protocol. At this point, M_s checks whether he could reach all $l-1$ subgroups via the peer-to-peer session keys he has. By reaching a subgroup, we mean the sponsor shares a peer-to-peer session key with at least one subgroup member and thus can transmit the keying information securely using the peer-to-peer session key. If so, M_s just needs to encrypt the according keying information with the peer-to-peer session

Member Leave Protocol :
Let M_x be the leaving member and M_s be the sponsor at level l. Let SE_s^* be the set of members each from different subgroups, which share no session key with M_s. Member nodes of KP_i^* and CP_i^* are denoted as $\{N_{1s}, N_{2s}, ..., N_{(l-1)s}\}$ and $\{N_{1\hat{s}}, N_{2\hat{s}}, ..., N_{(l-1)\hat{s}}\}$, respectively.
Round 1:
M_s multicasts SE_s^*: $\alpha^{r_s}||M_s||Establish$.
Round 2:
Each $M_g \in SE_s^*$ unicasts to M_s: $\alpha^{r_g S_{sg}}||M_g$;
M_s computes: $K_{sg} = \alpha^{r_s r_g}$, $M_g \in SE_s^*$; M_s updates all the key pairs of KP_s^*.
Round 3:
M_s multicasts SE_s^*: $\{\{B_{N_{is}}, N_{is} \in CP_g^*\}_{K_{sg}}, M_g \in SE_s^*\}$
Round 4:
Each $M_g \in SE_s^*$ multicasts own subgroup: $\{B_{N_{is}}, N_{is} \in CP_g^*\}_{K_{N_{i\hat{s}}}}$
All members except for M_x can independently compute the updated group key.

Fig. 8. Member leave protocol.

key for each $l-1$ subgroup member and multicasts them the updated keying information. Upon receiving it, each $l-1$ corresponding subgroup member can obtain the necessary blinded key. Each of them then broadcasts this blinded key to other subgroup members using the secret subgroup key. Otherwise, M_s first needs to establish enough peer-to-peer session key with each of the $l-1$ subgroups before transmitting the keying information. Note that any available member could do this job without relying on a particular one, and therefore, achieves fault-tolerance. M_s is determined by the following principles: Firstly, M_x is not an *isolated leaf node*, then its sibling node must also be a leaf node. In this case, M_x sibling will be chosen as M_s. Secondly, if M_x is an *isolated leaf node*, M_s will be the shallowest leaf node in the subtree rooted in M_x's sibling node. If there is more than one node, then the leftmost isolated leaf node has the priority; otherwise, choose the leftmost leaf node.

The protocol is depicted in Figure 8 using A-DH as the underlying two-party protocol. Totally 4 rounds are needed to update the group key. And the two-party protocol is required to execute 0 time at least and $d-1$ times at the worst case. The upper bound of multicast operations needed in the protocol is $d+1$. It was clear that the computation cost of the member leave process depends on both the position (level) of the leaving member in the key tree and the number of peer-to-peer session keys possessed by the sponsor M_x. Let N_K $(0 \leq N_K \leq d-1)$ be the number of peer-to-peer session keys the sponsor has. Then the two-party authenticated protocol needs to be executed $l - N_K - 1$ times. Obviously, $l - N_k - 1$ varies from 0 to $d-1$.

Two examples of member leave event are shown in Figure 9. In Figure 9(a), member M_5 leaves the group, so M_1 is the sponsor for M_5's leave event. M_1 computes its new secret share $B_1 = h(K_1) = h(\alpha^{r_1})$ with a fresh random nonce r_1 and $B_{136} = h(h(B_1||B_{36}))$. Then, M_1 multicasts $\{\{B_1||M_1\}_{K_{13}}, \{B_{136}||M_1\}_{K_{12}}\}$ to M_3 and M_2. Upon receiving this message, M_2 and M_3 obtain B_{136} and

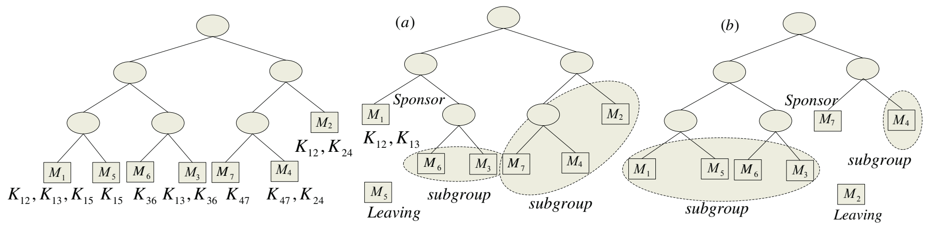


Fig. 9. Two examples of member leave event.

B_1, respectively. In turn, they each multicast it to their subgroup members: $M_2 : \{B_{136}\}_{K_{247}}$; $M_3 : \{B_1\}_{K_{36}}$. So, on decrypting the above message, all group members get the required blinded key and thus can update the group key. In Figure 9(b), M_2 leaves the group, so the sponsor is M_7. In this case, there exist two subgroups: (M_1, M_3, M_5, M_6) and (M_4) from the view of M_7. And M_4 and M_7 share no peer-to-peer session key with these two subgroups. So M_7 first multicasts to M_1 and M_4, two represents from each subgroup: α^{r_4} to establish two peer-to-peer session keys.

5 Complexity Analysis of EGAKA

We analyze the complexity of EGAKA by using A-DH as the underlying two-party protocol in order to provide a clear comparison.

Table 1 compares key establishment protocol of EGAKA with many other well known protocols. It is clearly that EGAKA and the protocol by Yang *et al.* [29] both has the best performance. Only $5n - 4$ exponentiations and $d + 2$ rounds (except for negotiation step) are needed by EGAKA-KE. Note that protocol in [29] requires less exponentiations only because they use an ID-based underlying two party protocol which takes 4 exponentiations per execution. And this protocol is not verifiable contributory as pointed out before. Moreover, this protocol provides no dynamic case and is mainly designed for static groups. On the other hand, protocol by Bresson *et al.* [4] is provably secure against both passive and active attacks, but obviously it is too computational intensive. $(n^2 + 4n)/2 - 1$ exponentiations and n signatures are needed for the key establishment protocol. At the same time, though protocol by Burmester *et al.* takes only two rounds, it is very computational intensive. As pointed out before, both SA-GDH and A-GDH are found to be flawed in [19].

The member join protocol of EGAKA-KU requires exactly two broadcast operations, and the two-party authenticated key agreement protocol executes only once. Communication rounds of member join protocol are usually 2 or 3 at the worst case. What more important and promising is all these operations are independent from the group size. This feature allows EGAKA-KU to provide highly efficient member join service compared with other proposed protocols. Table 2 compares different key update protocols. It is clear that EGAKA provides most efficient member join service. Only fixed 6 exponentiations are needed, which

Table 1. Key establishment protocol comparison.

Group Key Establishment	rounds	total messages	total exponentiations	exponentiations per member	total sigs
EGAKA-KE using A-DH	$d+2$	$2(n-2)$	$5n-4$	$[3, 2d+1]$	-
A.GDH.2 [2]	n	n	$(n^2+4n)/2-1$	$[3, 2n-1]$	-
SA-GDH.2 [2]	n	n	n^2	n	-
Yang *et al.* [29]	$d+1$	$2(n-2)$	$4n-4$	$[2, 2d]$	-
Bresson *et al.* [4]	n	n	$(n^2+4n)/2-1$	$[3, 2n-1]$	n
Burmester *et al.*[7]	2	$2n$	$n(n+1)$	$n+1$	n

are constant to the group size. Comparing this result to that of TGDH using Figure 11 (a) and (b) in [13], we can have a clearer idea about the superiority of EGAKA in member join service. At the same time, the member leave protocol of EGAKA-KU is less efficient, but the up-bound of the computational complexity is still linear to d. Note that the group size is assumed to be less than 200, so d is less than 8. TGDH is relatively efficient in member leave process, but TGDH provides no key establish protocol. The group key establishment is not described in TGDH and thus the security issues of the protocol is not clear. Again, protocol by Bresson *et al.* [4] is too computational intensive.

6 Security Analysis of EGAKA

We perform our security analysis in a computational complexity framework, and full security analysis of EGAKA will be provided separately due to the page limitation. Our attacker model distinguishes between passive and active adversaries. Passive adversaries only eavesdrop on the group communication (in particular they are never group members), whereas active adversaries may be previous group members. We do not consider insider attacks as explained in Section 3.

We assume that a passive attacker could eavesdrop all traffic. Therefore, the attacker does not know any keying information in the key tree, because no keying information is transmitted in the form of plaintext. Clearly, attacks to find the group key can be reduced to the attempt of breaking the underlying symmetric encryption algorithm. This can be viewed as an exhaustive key space searching, provided the symmetric encryption algorithm is secure, which takes $\mathcal{O}(2^n)$ operations, where n is the bit-length of the group key. The passive attacker can't do better by using public peer-to-peer session key establish information, because the underlying two-party protocol is assumed to be secure.

An active attacker's knowledge in our model equals to that of any former group member or their combination. We consider the following question: can an active attacker with such knowledge derive any new group session keys? Clearly, the active attacker can't know the secret key share of at least one current group member. This member is the one who updates its secret share after the latest leaving member. So the active attacker could not know its secret share, under the

Table 2. Key update protocol comparison.

Dynamic Case		rounds	total msgs	total exps	multicast	sigs	vers
EGAKA-KU using A-DH	Join	2, 3	2	6	2	-	-
	Leave	4	$[d, 2d]$	$[0, 5d-4]$	d	-	-
Bresson *et al.* [4]	Join	2	2	$2n$	1	2	$n+1$
	Leave	1	1	$2n$	1	2	$n-2$
TGDH [13]	Join	2	3	$3d/2$	3	2	3
	Leave	1	1	$3d/2$	1	1	1
Burmester *et al.*[7]	Join	2	$2n+2$	3	$2n+2$	-	-
	Leave	2	$2n-2$	3	$2n-2$	-	-

assumption of the security of the underlying symmetric encryption algorithm. So the active attacker cannot know the secret key of this member's key-path under the assumption of the intractability of one way hash function. Another way for an active attacker to compute the new group session keys is to pretend to be a legal party of the current group and trying to establish a peer-to-peer session key with the leave event sponsor and thus get the keying updating information he wants. This is prohibited by the underlying two-party protocol. Therefore, the active attacker cannot compute the group key except for brute force attack, whose complexity is $\mathcal{O}(2^n)$.

EGAKA provides verifiable contributory property. Every member in EGAKA independently computes the group key from its own secret key share and the blinded keys of its co-path nodes obtained from others. So if group members get wrong blinded key from others, then no common group key can be obtained. In other word, if only EGAKA is executed properly, then the resulting group key is verifiable contributory.

7 Conclusion

In this paper, we proposed an efficient group authenticated key agreement protocol (EGAKA), which is designed to be fully distributed, provides efficient dynamic group membership management, mutual authentication among group members and is secure against both passive and active attacks. EGAKA distinguishes itself from other existing protocols as follows: Firstly, EGAKA provides an efficient contributory key agreement framework which accommodates any two party authenticated key exchange protocol. EGAKA can be built on any two-party protocol without relying on a particular one. Therefore, EGAKA achieves scalability and robustness in heterogenous environments by allowing members to use any available two-party protocol in common and deliberately designed fault-tolerant mechanism in dynamic membership management. Secondly, EGAKA is superior to many protocols in the literature in terms of efficiency. In particular, EGAKA provides extremely efficient member join services. Both communication and computation costs are constant to the group size. This property is very useful in the scenarios with frequent member addition.

References

1. N. Asokan and P. Ginzboorg, "Key Agreement in ad-hoc Networks", in Computer Communication Review, 2000.
2. G. Ateniese, M. Steiner and G. Tsudik, "New Multi-party Authentication Services and Key Agreement Protocols", IEEE JSAC on Secure Communication, 2000.
3. C. Boyd and J. M. G. Nieto, "Round-efficient Conference Key Agreement Protocols with Provable Security", vol. 2567 of LNCS, pp. 161-174, Springer-Verlag, 2003.
4. E. Bresson, O. Chevassut and D. Pointcheval, "Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions", vol. 2332 of LNCS, pp. 321-336, 2002.
5. E. Bresson, O. Chevassut and D. Pointcheval, "Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case", In C. Boyd, Ed., Proc. of ASIACRYPT 2001, vol. 2248 of LNCS, pp. 290-309, 2001.
6. E. Bresson, O. Chevassut, D. Pointcheval and J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange", Proc. of the 8th ACM CCS'01, 2001.
7. M. Burmester and Yvo Desmedt, "Towards practical 'proven secure' authenticated key distribution", in 1st ACM CCS'93, Ed., Fairfax, Virginia, ACM Press, 1993.
8. M. Burmester, "On the Risk of Opening Distributed Keys", in Advances in Cryptology – CRYPTO '94, LNCS, pp.308–317, Springer-verlag, 1994.
9. M. Burmester, N. Alexandris, V. Chrissikopoulos and D. Peppes, "Efficient and Provably Secure Key Agreement', IFIP SEC '96, S.K. Katsikas and D. Gritzalis (Eds.), Chapman Hall, pp. 227-236, 1996.
10. M. Hietalahti, "Key Establishment in ad-hoc Networks", Tik-110.501, Seminar on Network Security, HUT TML, 2000.
11. J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange", Proc. of Crypt 2003, vol. 2729 of LNCS, Springer-Verlag, 2003.
12. J. Katz, R. Ostrovsky and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords", In L. Knudsen, Ed., Proc. of EUROCRYPT 2001, vol. 2045 of LNCS, Springer-Verlag, Berlin, 2002.
13. Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups", ACM CCS'2000, 2000.
14. K. Kobara and H. Imai, "Pretty-Simple Password-Authenticated Key-Exchange Under Standard Assumptions", IEICE Trans., vol. E85-A, pp. 2229-2237, 2002.
15. T. Kwon, "Authentication and Key Agreement via Memorable Passwords", In Proc. of NDSS'01, 2001.
16. D. McGrew and A. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", http://www.cs.umbc.edu/ sherman/itse.ps, 1998.
17. S. Lee, Y. Kim, K. Kim and D. Ryu, "An Efficient Tree-based Group Key Agreement using Bilinear map", LNCS Vol. 2846, pp.357-371, Springer-Verlag, 2003.
18. A. Menezes, P. Oorschot, and S. Vanstone, *Handbook of applied cryptography*, CRC Press series on discrete mathematics and its applications, CRC Press, 1997.
19. O. Pereira and J. Quisquater, "A Security Analysis of the Cliques Protocols Suites", 14th IEEE CSFW'01, Cape Breton, Novia Scotia, Canada, 2001.
20. A. Perrig, D. Song, and D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution", Proc. of IEEE Security and Privacy Symposium S&P 2001, 2001.
21. A. Perrig, Y. Kim and G. Tsudik, "Communication-Efficient Group Key Agreement", International Federation for Information Processing IFIP SEC 2001, 2001.
22. K. C. Reddy and Divya Nalla, "Identity Based Authenticated Group Key Agreement Protocol", Progress in Cryptology - INDOCRYPT 2002, India, 2002.

23. J. Smith and F. Weingarten, Eds., *Research Challenges for the Next Generation Internet.*, Workshop on Research Directions for the Next Generation Internet, 1997.
24. D. Steer L. Strawczynski, W. Diffie and M. Wiener, "A Secure Audio Teleconference System", In Proc. of CRYPTO'88, 1990.
25. M. Steiner, G. Tsudik and M. Waidner, "Key Agreement in Dynamic Peer Groups", IEEE Transactions on Parallel and Distributed Systems, 2000.
26. W. Tzeng and Z. Tzeng, "Round-efficient Conference Key Agreement Protocols with Provable Security", ASIACRYPT 2000, vol. 1976 of LNCS, pp. 614-627, 2000.
27. D. Wallner, E. Harder, and R. Agee, "key management for multicast: Issues and architecture", Internet Draft, draft-wallner-key-arch-00.txt, Jun. 1997.
28. C. Wong, M. Gouda, and S. Lam, Secure group communications using key graphs", IEEE/ACM Trans. on Networking, 8(1):16-30, 2000.
29. W. Yang, and S. Shieh, "Secure Key Agreement for Group Communications", ACM/PH International Journal of Network Management, Vol 11, No. 6, 2001.

Appendix: Protocol of EGAKA-KE Phase I Using A-DH

By using A-DH as the underlying two-party protocol, we depict EGAKA-KE Phase I as below.

Protocol EGAKA-KE Phase I :
Let $\{M_1, \cdots, M_n\}$ be a set of members wishing to establish a group key K_G. Let E_i be the set of group members that are the *partner*s of M_i.
Round 1
M_i, $i \in [1, n] \longrightarrow \{M_j \mid M_j \in E_i, j > i\}$: α^{r_i}.
Round 2
M_i, $i \in [1, n] \longrightarrow \{M_j \mid M_j \in E_i, j < i\}$: $\alpha^{r_i S_{ij}}$.
The resulting peer-to-peer session key is
$K_{ij} = \alpha^{r_i r_j}$.

Fig. 10. Protocol EGAKA-KE Phase I using A-DH.

A Ring Signature Scheme Using Bilinear Pairings

Jing Xu[1,2], Zhenfeng Zhang[1,3], and Dengguo Feng[1,3]

[1] State Key Laboratory of Information Security, P.R. China
[2] Graduate School of Chinese Academy of Sciences, Beijing 100039, P.R. China
[3] Institute of Software, Chinese Academy of Sciences, Beijing 100080, P.R.China
{xujing,zfzhang,feng}@is.iscas.ac.cn

Abstract. The bilinear pairings such as Weil pairing or Tate pairing over elliptic curves and hyperelliptic curves have been found various applications in cryptography very recently. Ring signature is a very useful tool to provide the user's anonymity and the signer's privacy. In this paper, we propose a ring signature scheme based on the bilinear pairings, which is secure against chosen message attacks without random oracles. Moreover, we use this ring signature scheme to construct a concurrent signature scheme for fair exchange of signatures.

1 Introduction

Group-oriented cryptography deals with those situations in which a secret task (signing or decrypting) is performed by a group of entities or on behalf of such a group. Threshold cryptography is an approach to this situation. In a threshold scheme, some participants have shares of the unique secret key of the group. Participation of some determined subset of players is required to perform the corresponding secret task in a correct way.

Two related but different approaches are ring signatures and group signatures. In a ring signature scheme, an entity signs a message on behalf of a set (or ring) of members that includes himself. The verifier of the signature is convinced that it was produced by some member of the ring, but he does not obtain any information about which member of the ring actually signed. The real signer includes in the signature the identities of the members of the ring that he chooses, depending on his purposes, and probably without their consent.

The idea behind group signature schemes is very similar to that of ring signatures, but with some variations. First of all, there exists a group manager in charge of the join and revocation of the members in the group. Therefore, a user cannot modify the composition of the group. And second, some mechanisms are added in order to allow (only) the group manager to recover the real identity of the signer of a message, for example in the case of a legal dispute.

Ring signatures are a useful tool to provide anonymity in some scenarios. For example, if a member of a group wants to leak to the media a secret information about the group, he can sign this information using a ring scheme and convince everyone that the information indeed comes from the group itself, without being accused of leaking the secret; or if a union could set up such a scheme for all workers of some company which then can leak information that provably come from this company (and thus know the working conditions), without revealing who complained.

A different application is the following: By using ring signature scheme, we can turn standard signature schemes into designated verifier signature schemes. If the signer A of a message wants that the authorship of the signature could be entirely verified only by some specific user B, he can sign the message with respect of the ring $\{A, B\}$. The rest of users could not know who between A and B is the actual author of the signature, but B will be convinced that the author is A.

In [1], Rivest, Shamir and Tauman formalize the concept of ring signature schemes, and propose a ring signature scheme which is proved to be existentially unforgeable under adaptive chosen-message attacks, in the ideal cipher model, assuming the hardness of the RSA problem. This scheme also uses a symmetric encryption scheme and the notion of combining function.

Bresson, Stern and Szydlo show in [2] that the scheme of [1] can be modified in such a way that the new scheme is proved to achieve the same level of security, but under the strictly weaker assumption of the random oracle model.

In [3], Abe, Ohkubo and Suzuki give general constructions of ring signature schemes for a variety of scenarios, including those where signature schemes are based on one-way functions, and those where signature schemes are of the three-move-type (for example, Schonrr's signature scheme).

Some security results for generic ring signature schemes, as well as a new specific scheme based on Schnorr's signature scheme, are given by Herranz and Saez in [4].

Recently, Dodis, Kiayias, Nicolosi and Shoup propose constant-size ring signatures using the Fiat-Shamir transform in [5]. This is the first such constant-size scheme, as all the previous proposals had signature size proportional to the size of the ring.

The bilinear pairings, namely the Weil-pairing and the Tate-pairing of algebraic curves, are important tools for research on algebraic geometry. They have been found various applications in cryptography recently [6],[7],[8],[9],[10].

In [10], Boneh and Boyen propose a short signature scheme which is existentially unforgeable under chosen message attacks without the random oracles. Based on this short signature scheme, we present a new ring signature scheme using the bilinear pairings, which is the first ring signature scheme secure without random oracle.

The problem of fair exchange of signatures is a fundamental and well-studied problem in cryptography, with potential application in a wide range of scenarios in which the parties involved are mutually distrustful. Ideally, we would like the exchange of signatures to be done in a fair way, so that by engaging in a protocol, either each party obtains the other's signature, or neither party does. It should not be possible for one party to terminate the protocol at some stage leaving the other party committed when they themselves are not.

For fair exchange of signatures, Chen etc. [11] introduce a new concept of concurrent signatures. These allow two entities to produce two signatures in such a way that, from the point of view of any third party, both signatures are ambiguous with respect to the identity of the signing party until an extra piece of information (the keystone) is released by one of the parties. Upon release of the keystone, both signatures become binding to their true signers concurrently. Using our ring signature scheme, we construct a concurrent signature scheme based on the bilinear maps.

The rest of the paper is organized as follows. In Section 2 we formally state our definition of security as well as basic tools used in our scheme. In Sections 3 and 4, we present a ring signature scheme and analyze its security and performance, respectively. In Section 5, a concurrent signature scheme based on our ring signature is constructed. And we end with concluding remarks in Section 6.

2 Definitions

2.1 The Bilinear Pairing

Let G be a cyclic additive group generated by P, whose order is a prime q, and V be a cyclic multiplicative group of the same order. Let $e : G \times G \rightarrow V$ be a pairing which satisfies the following conditions:

1. Bilinearity: For any $P, Q, R \in G$, we have $e(P+Q, R) = e(P, R)e(Q, R)$ and $e(P, Q + R) = e(P, Q)e(P, R)$. In particular, for any $a, b \in \mathbf{Z}_q$,

$$e(aP, bP) = e(P, P)^{ab} = e(P, abP) = e(abP, P).$$

2. Non-degeneracy: There exists $P, Q \in G$, such that $e(P, Q) \neq 1$.

3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G$.

The typical way of obtaining such pairings is by deriving them from the Weil-pairing or the Tate-pairing on an elliptic curve over a finite field. The interested reader is referred to [12] for a complete bibliography of cryptographic works based on pairings.

2.2 Ring Signature

The idea of a ring signature is the following: a user wants to compute a signature on a message, on behalf of a set (or ring) of users which includes himself. He wants the verifier of the signature to be convinced that the signer of the message is in effect some of the members of this ring. But he wants to remain completely anonymous. That is, nobody will know which member of the ring is the actual author of the signature.

A regular operation of a ring signature scheme consists of the execution of the two following algorithms:

Ring-Sign: If a user $A_t (1 \leq t \leq n)$ wants to compute a ring signature on behalf of a ring $A_1, A_2, \cdots, A_n$, he executes this probabilistic algorithm with input a message m, the public keys $pk_1, pk_2, \cdots, pk_n$ of the ring and his secret key sk_t. The output of this algorithm is a ring signature σ for the message m.

Ring-Verify: This is a deterministic algorithm that takes as input a message m and a ring signature σ, that includes the public keys of all the members of the corresponding ring, and outputs "*accept*" if the ring signature is valid, or "*reject*" otherwise.

The resulting ring signature scheme must satisfy the following properties:

Correctness: A ring signature generated in a correct way must be accepted by any verifier with overwhelming probability.

Anonymity: Any verifier should not have probability greater than $1/n$ to guess the identity of the real signer who has computed a ring signature on behalf of a ring of n members. If the verifier is a member of the ring distinct from the actual signer, then his probability to guess the identity of the real signer should not be greater than $1/(n-1)$.

Unforgeability: Among all the proposed definitions of unforgeability (see [13]), we consider the strongest one: any attacker must not have non-negligible probability of success in forging a valid ring signature for some message m on behalf of a ring that does not contain himself, even if he knows valid ring signatures for messages, different from m, that he can adaptively choose.

2.3 Concurrent Signature

We will refer to any two-party signature with the property that it could have been produced by either of the two parties as an ambiguous signature. We briefly explain how a concurrent signature protocol can be built using ambiguous signatures.

Since either of two parties could have produced such an ambiguous signature, both parties can deny having produced it. However, we note that if A creates an ambiguous signature which only either A or B could have created, and sends this to B, then B is convinced of the authorship of the signature (since he knows that he did not create it himself). However B cannot prove this to a third party. The same situation applies when the roles of A and B are reversed.

Suppose now that the ambiguous signature scheme has the property that, when A computes an ambiguous signature, she must choose some random bits h_B to combine with B's public key, but that the signing process is otherwise deterministic. Likewise, suppose the same is true for B with random bits h_A (when the roles of A and B are interchanged). Suppose A creates an ambiguous signature σ_A on M_A using bits h_B that are derived by applying a hash function to a string k that is secret to A; h_B is then a commitment to k. B can verify that A created the signature σ_A but not demonstrate this to a third party. Now B can create an ambiguous signature σ_B on M_B using as its input h_A the same h_B that A used. Again, A can verify that B is the signer. As long as k remains secret, neither party can demonstrate authorship to a third party.

But now if A publishes the keystone k, then any third party can be convinced of the authorship of both signatures. The reason for this is that the only way that B could produce σ_B is by following his signing algorithm, choosing randomness h_A and deterministically producing σ_B. The existence of a pre-image k of B's randomness h_A determines B as being the only party who could have conducted the signature generation process to produce σ_B. The same holds true for A and σ_A. Thus the pairs $< k, \sigma_A >$ and $< k, \sigma_B >$ amount to a simultaneously binding pair of signatures on A and B's messages. These pairs are called concurrent signatures [11].

The resulting concurrent signature scheme must satisfy the following security properties:

Correctness: A concurrent signature generated in a correct way must be accepted by any verifier with overwhelming probability.

Unforgeability: For concurrent signatures, the definition of existential unforgeability against chosen message attacks of [13] should be extended to the multi-party setting. The extension is similar to that of [14] and is strong enough to capture an adversary who can simulate and observe concurrent signature protocol runs between any pair of participants.

Ambiguity: Either of two parties could have produced such an signature, both parties can deny having produced it.

Fairness: From the point of view of any third party, both signatures are ambiguous with respect to the identity of the signing party until the keystone is released by one of the parties. Upon release of the keystone, both signature become binding to their true signers concurrently.

Formal definition of concurrent signature and concurrent signature protocol is referred to [11].

3 Our Ring Signature Scheme

The ring signature scheme comprises three algorithms: **Key Generation, Ring Signing**, and **Ring Verification**. Recall that G is a cyclic additive group generated by P, whose order is a prime p, and $e : G \times G \to V$ is a bilinear map.

Key Generation. For a particular user, pick $x_s, y_s \in \mathbf{Z}_p^*$ at random, and compute $u_s = x_s P$, $v_s = y_s P$. The user's public key is (u_s, v_s). The corresponding secret key is (x_s, y_s).

Ring Signing. Given public keys (u_1, v_1), (u_2, v_2), $\cdots$, (u_n, v_n), a message $m \in \mathbf{Z}_p^*$, and a private key (x_s, y_s) corresponding to one of the public keys (u_s, v_s) for some s, choose $r \in \mathbf{Z}_p^*$ and $a_i \in \mathbf{Z}_p^*$ at random for all $i \neq s$. Compute

$$\sigma_s = \frac{1}{m + x_s + y_s r}\left(P - \sum_{i \neq s} a_i(mP + u_i + rv_i)\right).$$

In the unlikely event that $m + x_s + y_s r = 0$, we try again with a different random number $r \in \mathbf{Z}_p^*$. For all $i \neq s$, let $\sigma_i = a_i P$. Output the ring signature

$$\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n, r).$$

Ring Verification. Given public keys (u_1, v_1), (u_2, v_2), $\cdots$, (u_n, v_n), a message $m \in \mathbf{Z}_p^*$, and a ring signature $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n, r)$, verify that

$$\prod_{i=1}^{n} e(mP + u_i + rv_i, \sigma_i) = e(P, P). \tag{1}$$

4 Analysis of Our Scheme

There are three properties for the security analysis of a ring signature one must consider: correctness, anonymity and unforgeability.

Correctness. The signature scheme is correct because of the following.

$$\begin{aligned}
&\prod_{i=1}^{n} e(mP + u_i + rv_i, \sigma_i) \\
&= \prod_{i \neq s} e(mP + u_i + rv_i, \sigma_i) e(mP + u_s + rv_s, \sigma_s) \\
&= \prod_{i \neq s} e(mP + u_i + rv_i, \sigma_i) \\
&\quad \times e\left(mP + u_s + rv_s, \frac{1}{m + x_s + y_s r}\left(P - \sum_{i \neq s} a_i(mP + u_i + rv_i)\right)\right) \\
&= e(P, P).
\end{aligned}$$

Anonymity. We show by the following theorem that the identity of the signer is unconditionally protected.

Theorem 4.1. *For any algorithm $\mathcal{A}$, any set of users U, and a random $u \in U$, the probability $Pr[\mathcal{A}(\sigma) = u]$ is at most $\frac{1}{|U|}$, where σ is a ring signature on U generated with private key sk_u.*

Proof. Assume that $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n, r)$ is a ring signature on the set of users U generated with private key sk_u. All σ_i except σ_u are taken randomly from G due to $\sigma_i = a_i P$ and $a_i \in_{\mathcal{R}} \mathbf{Z}_p^*$, while σ_u is computed by these a_i, m and sk_u. Therefore, for any fixed U and m, $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ has $|U|^{n-1}$ possible values, all of which can be chosen by the signature generation procedure with equal probability and regardless of the signer. At the same time, the distribution $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ is identical to the distribution

$$\left\{(a_1 P, a_2 P, \cdots, a_n P) : \sum_{i=1}^{n} a_i P = C\right\},$$

here C is an element of G depend on U and m. So the conclusion is correct. □

Unforgeability. It can be easily seen that, for $n = 1$, our ring signature is actually the short signature scheme proposed by D.Boneh [10], which is existentially unforgeable under the chosen message attacks without random oracles. The unforgeability of our ring signature can be shown by the similar method as [10].

Our ring signature scheme can be performed with supersingular elliptic curves or hyperelliptic curves. The essential operation in our signature schemes is to compute bilinear pairings, which can be efficiently computed due to [15] and [16].

There are $n + 1$ pairing operations in the verification algorithm. However, from (1) we see that n of these pairing operations can be executed in parallel, which is quit different from the sequential operation in Rivest's scheme [1]. Note that the $e(P, P)$ operation only need to be computed at initialization and its value can be cached. Hence the verification operation can be implemented in parallel and the running time is comparable to one pairing operation.

5 Concurrent Signature Scheme

We present a concrete concurrent signature scheme in this section, which is directly derived from our ring signature scheme.

5.1 Concurrent Signature Algorithm

The four algorithms (SETUP, ASIGN, AVERIFY, VERIFY) of our concurrent signature scheme are as follows:

SETUP. Let G be a cyclic additive group generated by P, whose order is a prime p, and $e : G \times G \to V$ be a bilinear map. The keystone space $\mathcal{K} = \{0, 1\}^*$, the keystone fix space $\mathcal{F} = \mathbf{Z}_p^*$. Two cryptographic hash functions $H_1, H_2 : \{0, 1\}^* \to \mathbf{Z}_p^*$ are selected. Define KGEN: $\mathcal{K} \to \mathcal{F}$ to be H_1. For all $1 \le i \le n$, private keys x_i, y_i are chosen uniformly at random from $\mathbf{Z}_p^*$. The corresponding public keys are computed as $U_i = x_iP, V_i = y_iP$ and are made public.

ASIGN. The algorithm ASIGN takes as input $\langle U_i, V_i, U_j, V_j, x_i, y_i, h_2, m\rangle$, where $i \neq j$, and (U_i, V_i), (U_j, V_j) are public keys, (x_i, y_i) is the private key corresponding to (U_i, V_i), $h_2 \in \mathcal{F}$ and m is a message. The algorithm picks $a \in \mathbf{Z}_p^*$, $r \in \mathbf{Z}_p^*$ at random and then computes:

$$\begin{aligned} h_1 &= H_2(h_2 \| U_j \| V_j \| m) \\ s_1 &= \frac{1}{m + x_i + y_i r}\left(h_1^{-1}P - a\,(mP + U_j + rV_j)\right) \\ s_2 &= a h_2^{-1} h_1 P \end{aligned}$$

Here "$\|$" denotes concatenation. The algorithm outputs an ambiguous signature

$$\sigma = \langle s_1, s_2, r, h_1, h_2\rangle.$$

AVERIFY. The algorithm AVERIFY takes as input $\langle \sigma, U_i, V_i, U_j, V_j, m \rangle$, where $\sigma = \langle s_1, s_2, r, h_1, h_2 \rangle$ is an ambiguous signature with $h_1, h_2 \in \mathcal{F}$, and (U_i, V_i) and (U_j, V_j) are public keys, m is a message. The algorithm checks that the following equation:

$$\prod_{i=1}^{2} e\Big(h_i\left(mP + U_i + rV_i\right), s_i\Big) = e(P, P)$$

and outputs *accept* if it holds. Otherwise, it outputs *reject*. The correctness of AVERIFY can be easily seen from (1).

VERIFY. This is an algorithm which takes as input $\langle k, S \rangle$, where $k \in \mathcal{K}$ is a keystone and S is of the form $S = \langle \sigma, U_i, V_i, U_j, V_j, m \rangle$, where $\sigma = \langle s_1, s_2, r, h_1, h_2 \rangle$ is an ambiguous signature with $h_1, h_2 \in \mathcal{F}$, and (U_i, V_i) and (U_j, V_j) are public keys, m is a message. The algorithm checks if KGEN$(k) = h_2$. If not, it terminates with output *reject*. Otherwise it runs AVERIFY(S).

5.2 Concurrent Signature Protocol

We will describe a concurrent signature protocol between two parties A and B (or Alice and Bob). Since one party needs to create the keystone and send the first ambiguous signature, we call this party the initial signer. A party who responds to this initial signature by creating another ambiguous signature with the same keystone fix is called a matching signer. Without loss of generality, we assume A is the initial signer, and B is the matching signer. The signature protocol works as follows:

A and B run SETUP to determine the public parameters of the scheme. We assume that A's public keys and private keys are (U_A, V_A) and (x_A, y_A) respectively, and B's public keys and private keys are (U_B, V_B) and (x_B, y_B).

1. A picks a random keystone $k \in \mathcal{K}$, and computes f =KGEN(k). A takes her own public key (U_A, V_A) and B's public key (U_B, V_B) and picks a message m_A to sign. A then computes her ambiguous signature to be $\sigma_A =$ ASIGN $(U_A, V_A, U_B, V_B, x_A, y_A, f, m_A)$, and sends it to B.

2. Upon receiving A's ambiguous signature σ_A, B verifies the signature by checking that AVERIFY$(\sigma_A, U_A, V_A, U_B, V_B, m_A) =$ *accept*. If not B aborts, otherwise B picks a message m_B to sign and computes his ambiguous signature $\sigma_B =$ ASIGN$(U_A, V_A, U_B, V_B, x_B, y_B, f, m_B)$ and sends this back to A. Note that B uses the same value f in his signature as A did to produce σ_A.

3. Upon receiving B's signature σ_B, A verifies its correctness by checking that AVERIFY$(\sigma_B, U_A, V_A, U_B, V_B, m_B)$=*accept*, where f is the same keystone as A used in Step 1. If not, A aborts, otherwise A sends keystone k to B.

Note that inputs $\langle k, S_A \rangle$ and $\langle k, S_B \rangle$ will now both be accepted by VERIFY, where $S_A = \langle \sigma_A, U_A, V_A, U_B, V_B, m_A \rangle$ and $S_B = \langle \sigma_B, U_A, V_A, U_B, V_B, m_B \rangle$.

5.3 Security Analysis

Now we analyze the security of the concurrent signature scheme.

Unforgeability: Since the concurrent signature scheme is based on the signature scheme proposed by D.Boneh [10] which is existentially unforgeable under a chosen message attacks, it is also unforgeable.

Ambiguity: The concurrent signature scheme is a direct modification of our ring signature scheme, hence it is ambiguous in the random oracle model.

Fairness: The concurrent signature scheme is fair in the random oracle.

Proof: Since H_1 is a random oracle, the adversary $\mathcal{A}$'s probability of producing k such that $f = H_1(k)$ is negligible. Furthermore, suppose S_A is accepted by AVERIFY and $\langle k, S_A \rangle$ is accepted by VERIFY, we must have KGEN$(k) = f$. Since S_A and S_B share the value f, we must also have that $\langle k, S_B \rangle$ is accepted by VERIFY. The detailed proof is similar to [11]. $\square$

So our concurrent signature scheme is secure in the random oracle.

6 Conclusion

In this paper we presented a new ring signature scheme using the bilinear pairings which is the first ring signature scheme secure against chosen message attacks without random oracle. Furthermore, we proposed a concurrent signature scheme based on this ring signature, which can be used for fair exchange of signatures.

Acknowledgements

This work is supported by the National Grand Fundamental Research Program of China under Grant No. G1999035802, the National Natural Science Foundation of China under Grant No. 60373039, and the Youth Foundation of the National Natural Science of China under Grant No. 60025205. The authors would like to thanks the anonymous referees for their helpful comments.

References

1. R. Rivest, A. Shamir and Y. Tauman. How to leak a secret. Advances in Cryptology-Asiacrypt 2001, LNCS 2248, 552-565 , Springer-Verlag, 2001.
2. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. CRYPTO 2002, LNCS 2442, 465-480. Springer-Verlag, 2002.
3. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. ASIACRYPT 2002, LNCS 2501, 415-432. Springer-Verlag, 2002.
4. Javier Herranz, German Saez. Forking Lemmas for Ring Signature Schemes. INDOCRYPT 2003, LNCS 2904, 266-279. Berlin: Springer- Verlag, 2003.
5. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous Identification in Ad Hoc Groups, Eurocrypt 2004,LNCS 3027, 609-626, Springer-Verlag, 2004.
6. D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing, Advances in Cryptology-Crypto 2001, LNCS 2139, 213-229, Springer-Verlag, 2001.
7. D. Boneh, B. Lynn, and H. Shacham, Short signatures from the Weil pairing, Advances in Cryptology-Asiacrypt 2001, LNCS 2248, 514-532, Springer-Verlag, 2001.

8. A. Joux, The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems, ANTS 2002, LNCS 2369, 20-32, Springer-Verlag, 2002.
9. M.S. Kim and K. Kim, A new identification scheme based on the bilinear Diffie-Hellman problem, Proc. of ACISP(The 7th Australasian Conference on Information Security and Privacy) 2002, LNCS 2384, 464-481, Springer-Verlag, 2002.
10. D.Boneh and X.Boyen, Short Signatures Without Random Oracles. Eurocrypt 2004, LNCS 3027, 56-73, Springer-Verlag, 2004.
11. L.-Q. Chen, C. Kudla, and K.G. Paterson, Concurrent Signatures, Eurocrypt 2004, LNCS 3027, 287-305, Springer-Verlag, 2004.
12. The pairing-Based Crypto Lounge. Web page maintained by Paulo Barreto: http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html
13. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptative chosen-message attacks. SIAM Journal of Computing, 17 (2), 281-308, 1988.
14. X. Boyen, Multipurpose identity-based signcryption. A Swiss army knife for identity-based cryptography. Advances in Cryptology - CRYPTO 2003, LNCS 2729, 383-399, Springer-Verlag, 2003.
15. P.S.L.M. Barreto, H.Y. Kim, B.Lynn, and M.Scott, Efficient algorithms for pairing-based cryptosystems, Advances in Cryptology-Crypto 2002, LNCS 2442, 354-368, Springer-Verlag, 2002.
16. S. D. Galbraith, K. Harrison, and D. Soldera, Implementing the Tate pairing, ANTS 2002, LNCS 2369, 324-337, Springer-Verlag, 2002.

Verifiable Pairing and Its Applications

Sherman S.M. Chow

Department of Computer Science
The University of Hong Kong
Hong Kong
smchow@cs.hku.hk

Abstract. Pairing-based cryptography is rapidly emerging in recent years. Many cryptographic protocols, such as signcryption, threshold decryption and undeniable signature enabled by pairing require sending the result of the pairing function with private key as one of the input. Since private key is only known to its owner, the correctness of the result may not be easily verifiable (which requires solving the decisional bilinear Diffie-Hellman problem). In this paper, we introduce the notion of *Verifiable Pairing*, together with a concrete construction, to ensure the robustness of these protocols. Verifiable pairing is a useful primitive in many information security applications. As examples, we show how verifiable pairing can be applied in signcryption, threshold decryption and how it can help in fixing insecure protocol. In adding verifiability to threshold decryption, our solution is more efficient than the previous proposal in [21]. As a bonus result, we find that our scheme for verifiable pairing gives rise to a new identity-based signature that is provably secure in the random oracle model without using the forking lemma, assuming the hardness of the computational bilinear Diffie-Hellman problem.

Keywords: Bilinear pairing, verifiable pairing, ID-based cryptography, ID-based signature, exact security, cryptographic primitives, cryptographic protocols, signcryption, threshold decryption, undeniable signature

1 Introduction

Bilinear pairing (see [4] for implementation details) is a cryptographic primitive that is recently applied extensively in cryptography. It gives rise to many cryptographic schemes that are yet to be (efficiently) constructed using other cryptographic primitives, e.g. aggregate signature [5], short signature [6] and short group signature [3]. One of the most distinguishing cryptographic schemes enabled by bilinear pairing is identity based (ID-based) encryption [4], which solves the open problem proposed by [27] in 1984. After this seminal work, a lot of other ID-based cryptographic schemes are proposed, including provably secure signature [7, 24], hierarchical encryption and signature [11, 17], blind signature [12, 28], signcryption [13, 22], threshold decryption and signature [10, 21], undeniable signature [18, 23], etc.

Many cryptographic protocols constructed from pairing, such as signcryption, threshold decryption and undeniable signature, require sending the result of the pairing function with private key as one of the input. Since private key is only known to its owner, it may not be possible that the correctness of the result can be verified (which requires solving the decisional bilinear Diffie-Hellman problem). In this paper, we introduce the notion of *Verifiable Pairing* with a concrete construction to ensure the robustness of these protocols. We also show the completeness and soundness of our scheme. As examples of application, we propose how verifiable pairing helps in ensuring the non-repudiation property of signcryption, adding robustness to threshold decryption and fixing an insecure undeniable signature. Besides, our solution is more efficient than the previous mechanism that ensures robustness in threshold decryption [21].

Interestingly, we find that the verifiability of our proposed solution can help in making a new ID-based signature scheme from bilinear pairings that is provable secure in the random oracle model [2] without using the forking lemma [26]. This open question has remained unsolved for a few years until the recent work in [24]. As far as we know, our scheme is the second such scheme.

The rest of the paper is organized as follows. In Section 2, we review the properties of bilinear pairings and the related complexity assumptions. We introduce the notion of verifiable pairing in Section 3, together with a concrete construction and its security analysis. Section 4 shows how verifiable pairing adds robustness to threshold decryption schemes and how our solution outperforms the existing proposal in [21]. Section 5 presents a new provably secure ID-based signature scheme without using the forking lemma. In Section 6, we show that verifiable pairing is indeed essential in ensuring the non-repudiation property of the ID-based signcryption proposal by [22]. We then show how verifiable pairing plays its role in fixing insecure protocol like [18] in Section 7. Finally we conclude our paper in Section 8.

2 Preliminaries

2.1 Bilinear Pairings and Related Complexity Assumptions

Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ be two cyclic groups of prime order q. The bilinear pairing is given as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which satisfies the following properties:

1. *Bilinearity*: $\forall\ P, Q, R \in \mathbb{G}_1$, $\hat{e}(P + Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$, and $\hat{e}(P, Q + R) = \hat{e}(P, Q)\hat{e}(P, R)$.
2. *Non-degeneracy*: There exists $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)\ \forall P, Q \in \mathbb{G}_1$.

Definition 1. *Given a generator P of a group $\mathbb{G}$ and a 3-tuple (aP, bP, cP), the Decisional Diffie-Hellman problem (DDH problem) is to decide whether $c = ab$.*

Definition 2. *Given a generator P of a group $\mathbb{G}$ and a 2-tuple (aP, bP), the Computational Diffie-Hellman problem (CDH problem) is to compute abP.*

Definition 3. *If $\mathbb{G}$ is a group such that DDH problem can be solved in polynomial time, but no probabilistic algorithm can solve the CDH problem with non-negligible advantage within polynomial time, then we call $\mathbb{G}$ a Gap Diffie-Hellman (GDH) group.*

Definition 4. *Given a generator P of a group $\mathbb{G}$, and a 3-tuple $(aP, bP, cP) \in \mathbb{G}_1^3$, the Computational Bilinear Diffie-Hellman problem (CBDH problem) is to compute $\tau = \hat{e}(P, P)^{abc}$.*

Weil pairing [4] and Tate pairing [16] are two admissible bilinear pairings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ that one can use them to solve the DDH problem in polynomial time but the CDH problem remains difficult (i.e. $\mathbb{G}_1$ is a GDH group).

3 Verifiable Pairing

3.1 Overview

Verifiable Pairing aims at solving the following problem: Given a generator P of a group $\mathbb{G}$ and a 4-tuple $(aP, bP, abP, cP) \in \mathbb{G}_1^4$, one can enable another party, who only knows the value of aP, bP and cP, to compute $\hat{e}(P, P)^{abc}$ but not $\hat{e}(abP, R)$ for any arbitrary $R \in \mathbb{G}_1$, i.e. helping other to solve one instance of the CBDH problem without leaking abP. Notice that a, b and c are unknown to both parties.

The significance of the verifiable pairing comes from the fact that there are many ID-based schemes (e.g. [7, 17, 22–24, 28]) sharing the same private key extraction algorithm as that of [4]:

`Extract:`
Let $H_1(\cdot)$ be a cryptographic hash function where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$. The user with identity $ID \in \{0,1\}^*$ submits ID to the private key generator (PKG). the PKG sets the user's public key Q_{ID} to be $H_1(ID) \in \mathbb{G}_1$, computes the user's private key S_{ID} by $S_{ID} = sQ_{ID}$, where $s \in \mathbb{F}_q^*$ is the master secret key of the PKG. Then PKG sends the private key to the user in a secure channel.

Treating a as the master secret key s, bP as the public key Q_{ID} of the user ID, and cP as a certain part of the messages in an ID-based scheme (e.g. signature, encrypted message), verifiable pairing helps the holder of the private key S_{ID} to convince other the correctness of $\hat{e}(S_{ID}, cP)$ without leaking the private key.

3.2 Proposed Construction

Define $\mathbb{G}_1$, $\mathbb{G}_2$ and $\hat{e}(\cdot,\cdot)$ as in previous section. Suppose Alice holds the value of abP, and she wants to let Bob to compute $\hat{e}(abP, cP)$ without leaking abP. She proceeds as follows.

1. Randomly choose z from $\mathbb{F}_q^*$.
2. Compute z^{-1} (Both z and z^{-1} are kept in secret).
3. Compute $T = z^{-1}(cP)$.
4. Compute $U = z(bP)$.
5. Compute $V = z(abP)$.
6. Send (T, U, V) to Bob (It is assumed Bob knows the value of P, aP, bP and cP already.)

After receiving (T, U, V) from Alice, Bob performs the following to compute the value of $\hat{e}(P, P)^{abc}$.

1. Check whether $\hat{e}(V, P) = \hat{e}(U, aP)$.
2. Check whether $\hat{e}(U, T) = \hat{e}(bP, cP)$.
3. If both of the equalities hold, Bob can compute the value of $\hat{e}(P, P)^{abc}$ by $\hat{e}(V, T)$; otherwise, Bob knows that Alice is intended to cheat him.

3.3 Completeness and Soundness

It is easy to see the completeness of the scheme. For soundness, $\hat{e}(V, P) = \hat{e}(U, aP)$ implies $\hat{e}(V, P) = \hat{e}(aU, P)$. From the non-degeneracy of bilinear pairing, we know that $V = aU$. Assume $T = d(cP)$, where $d \in \mathbb{F}_q^*$, from $\hat{e}(U, T) = \hat{e}(bP, cP)$, we know that $\hat{e}(dU, cP) = \hat{e}(bP, cP)$, which implies $U = d^{-1}(bP)$. Finally, $\hat{e}(V, T) = \hat{e}(aU, d(cP)) = \hat{e}(d^{-1}(abP), d(cP)) = \hat{e}(P, P)^{abc}$. Moreover, obtaining abP from (T, U, V) is difficult as it involves solving the discrete logarithm problem. We defer our efficiency analysis to Section 4 and we show a more formal analysis in Section 5.

3.4 A More Efficient Construction

From the bilinearity of bilinear pairing, it seems possible to save another pairing computation by aggregating the step 1 and step 2 of the verification process, for example, by checking whether $\hat{e}(U, aP + T) = \hat{e}(bP, cP)\hat{e}(V, P)$. However, this "batch verification" makes the scheme insecure. After the above change, it is possible for the adversary to construct special values of T', U' and V' in a way different from our proposed construction, yet (T', U', V') can still pass the verification process.

The above class of attack is possible only when the adversary can manipulate the values of T and U and V according to the some pre-defined equation like $\hat{e}(U, aP + T) = \hat{e}(bP, cP)\hat{e}(V, P)$. To remove this vulnerability, we can borrow the idea of Fiat-Shamir heuristic [15]. The new verification process is to check whether $\hat{e}(U, aP + xT) = \hat{e}(bP, x(cP))\hat{e}(V, P)$, where $x \in \mathbb{F}_q^*$. Since the adversary does not know the value of x before the tuple (T, U, V) is sent, he/she has no idea in how to manipulate their values to satisfy the unknown verification equation. The probability of cheating is greatly reduced by this technique[1].

4 ID-Based Threshold Decryption

4.1 Overview

One of the motivations of threshold decryption schemes is to decentralize the decryption power. Threshold decryption schemes also address the problem of unavailability, in which any t out of n $(t < n)$ entities can together decrypt a given ciphertext. ID-based threshold decryption is more useful than traditional threshold decryption since the identity can be the name of the group sharing a decryption key. Other information security applications of ID-based threshold decryption include mediated ID-based encryption scheme [21]. We first review Libert and Quisquater's threshold decryption scheme [21], and their scheme's robustness feature, then we compare their construction with ours.

4.2 Libert and Quisquater's Scheme

Let H_1 and H_2 be two cryptographic hash functions where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$. The following shows the details of the scheme.

[1] An anonymous reviewer suggested this approach.

- `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$. The PKG chooses $s \in \mathbb{F}_q^*$ randomly and sets $P_{pub} = sP$. The master secret key is s, which is kept secret and known by the PKG only. Then it picks $a_1, \cdots, a_{t-1} \in_R \mathbb{F}_q^*$ and constructs a polynomial $f(x) = s + a_1 x + \cdots + a_{t-1}x^{t-1}$ of degree $t-1$. For $i = 1, \cdots, n$, it computes $P_{pub}^{(i)} = f(i)P$. The system parameters are

$$\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}^{(1)}, \cdots, P_{pub}^{(n)}, P_{pub}, \hat{e}(\cdot,\cdot), H_1(\cdot), H_2(\cdot)\}.$$

- `Extract`: It is the threshold extension of the private key extraction algorithm defined in Section 3.2. Given an identity $ID \in \{0,1\}^*$, the PKG sets the public key Q_{ID} to be $H_1(ID) \in \mathbb{G}_1$. For $i = 1, \cdots, n$, it computes the user's private key $S_{ID}^{(i)}$ by $S_{ID}^{(i)} = f(i)Q_{ID}$. Then the PKG sends one share of the private key $S_{ID}^{(i)}$ to the user i in a secure channel.
- `Encrypt`: To encrypt a plaintext m to user ID,
  1. Compute $Q_{ID} = H_1(ID)$,
  2. Pick a random $r \in \mathbb{F}_q^*$, (the random short term private key)
  3. Compute $g = \hat{e}(P_{pub}, Q_{ID})$,
  4. Set the ciphertext to $C =< rP, m \oplus H_2(g^r) >$. ($g^r$ can be viewed as the session key)
- `Decrypt`: Given a ciphertext $< U, V >$ and a private key share $S_{ID}^{(i)}$, player i computes $g^{(i)} = \hat{e}(U, S_{ID}^{(i)})$.
- `Recombine`: After the recombiner receives t valid shares $g^{(i)} = \hat{e}(U, S_{ID}^{(i)})$ from a certain subset of size t of the user group $\{1, \cdots, n\}$, the recombination goes as follows.
  1. Compute $g = \prod_{i \in S} {g^{(i)}}^{L_i}$, where L_i denotes the i-th Lagrange coefficient.
  2. Recover $m = V \oplus H_2(g)$,

4.3 Robustness Feature Provided by Libert and Quisquater's Scheme

Even if the recombiner gets only one corrupted or inconsistent share, the decryption fails since $t-1$ entities cannot recover the encrypted text in a threshold decryption scheme. To add robustness feature to threshold decryption, we need a mechanism to check the validity of a share. Below shows the procedure proposed in [21], with the help of an extra hash function $H_3 : \mathbb{G}_2^4 \to \mathbb{F}_q^*$.

Each participating player i, after having computed $g^{(i)} = \hat{e}(U, S_{ID}^{(i)})$, does the following.

1. Pick $R \in_R \mathbb{G}_1$.
2. Compute $w_1 = \hat{e}(P, R)$.
3. Compute $w_2 = \hat{e}(U, R)$.
4. Compute $h = H_3(g^{(i)}, \hat{e}(P_{pub}, Q_{ID}), w_1, w_2)$.
5. Compute $V = R + hS_{ID}^{(i)}$.
6. Send (w_1, w_2, V) to the recombiner.

Then the recombiner verifies the shares by following the below procedure.

1. Compute $h = H_3(g^{(i)}, \hat{e}(P_{pub}, Q_{ID}), w_1, w_2)$.
2. Check whether $\hat{e}(P, V) = w_1 \hat{e}(P_{pub}^{(i)}, Q_{ID})^h$.
3. Check whether $\hat{e}(U, V) = w_2 {g^{(i)}}^h$.

4.4 Comparison with Our Verifiable Pairing

To compare these two pieces of work, we consider operations including point addition in $\mathbb{G}_1$ ($\mathbb{G}_1$ Add), point scalar multiplication in $\mathbb{G}_1$ ($\mathbb{G}_1$ Mul), exponentiation in $\mathbb{F}_q$ ($\mathbb{F}_q$ Exp), division in $\mathbb{F}_q$ ($\mathbb{F}_q$ Div), hashing (Hash) and pairing operation (Pairing).

By using the above procedures proposed by [21], to convince another party, one needs to perform 3 pairing operations, 1 point addition and 1 point scalar multiplication, while the verifier needs to compute 4 pairing operations and 2 exponentiations in $\mathbb{G}_2$. Compared with our verifiable pairing, our solution is more efficient in the number of pairing operations needed (see Table 1, we call the party who needs to convince another the *initiator*).

Although some researches have been done in analyzing the complexity and speeding up the pairing computation (for example, [1, 16, 19, 9]), pairing operations are still rather expensive. Moreover, the inefficiency of [21] magnifies in the situation of threshold decryption since there are t parties to convince the recombiner. It gives the motivation for us to remove *all* the pairing computations of the initiator side. For the verifier side, our basic construction only requires one more pairing computation and the variant of our basic construction is as efficient as the construction in [21].

5 A New ID-Based Signature with Exact Security

5.1 Overview

The use of the forking lemma [26] to prove the unforgeability of signature schemes is very popular in recent years. However, application of the forking lemma does not

Table 1. Efficiency of our proposed constructions.

Entities	Efficiency					
	G_1 Add	G_1 Mul	F_q Exp	F_q Div	Hash	Pairing
Previous Construction in [21]						
$Initiator$	1	1	0	0	1	3
$Verifier$	0	0	2	0	1	4
Verifiable Pairing (Basic Construction)						
$Initiator$	0	3	0	1	0	0
$Verifier$	0	0	0	0	0	5
Verifiable Pairing (More Efficient Version)						
$Initiator$	0	3	0	1	0	0
$Verifier$	0	2	0	0	0	4

yield tight security reductions. To the best of our knowledge, there is only one ID-based signature (IBS) [24]that is provably secure without using the forking lemma (and hence has a tight security reduction). In this section, we show how to spawn another ID-based signature scheme with a good exact security from our verifiable pairing. Our construction requires a source of randomness but we do not require the oracle replay method and the forking lemma in the security proof.

5.2 Security Notion of ID-Based Signature

We consider the notion of *existential forgery for adaptive chosen-message-and-identity attack* (EF-CMIA) [7]. Its formal definition is based on the following EF-CMIA game.

Definition 5. *The existential forgery for adaptive chosen-message-and-identity attack game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is defined as follows.*

Setup:
The challenger $\mathcal{C}$ takes a security parameter k and runs `Setup` to generate the common public parameters $param$ and the master secret key s. $\mathcal{C}$ sends $param$ to $\mathcal{A}$. The master secret key s will be kept by $\mathcal{C}$.

Attack:
The adversary $\mathcal{A}$ can perform a polynomially bounded number of queries in an adaptive manner (that is, each query may depend on the responses to the previous queries). The types of queries allowed are described below.

- Any hashing query necessary for normal execution of the scheme.
- `Extract`: $\mathcal{A}$ chooses an identity ID. $\mathcal{C}$ computes $\texttt{Extract}(ID) = S_{ID}$ and sends the result to $\mathcal{A}$.
- `Sign`: $\mathcal{A}$ chooses an identity ID, and a plaintext m. $\mathcal{C}$ signs the plaintext by computing $\sigma = \texttt{Sign}(m, S_{ID})$ and sends σ to $\mathcal{A}$.

Forgery:
After the Attack phase, the adversary $\mathcal{A}$ outputs (σ, m^*, ID^*) where ID^* does not appear in any `Extract` query and the `Sign` query on (m^*, ID^*) does not appear in the Attack phase too.

The adversary $\mathcal{A}$ wins the game if the response of the `Verify` on (σ, m, ID^*) is not equal to $Invalid$. The advantage of $\mathcal{A}(\texttt{AdvSig}_{\mathcal{A}})$ is defined as the probability that it wins.

Definition 6. *A forger $\mathcal{A}$ $(t, q_S, q_H, q_E, \epsilon)$-breaks a signature scheme if $\mathcal{A}$ runs in time at most t, and makes at most q_S signature queries, q_H hash queries and q_E key extraction queries, while $\texttt{AdvSig}_{\mathcal{A}}$ is at least ϵ. A signature scheme is $(t, q_S, q_H, q_E, \epsilon)$-existentially unforgeable under an adaptive chosen message attack if there exists no forger that can $(t, q_S, q_H, q_E, \epsilon)$-break it.*

5.3 Proposed Construction

Let $H_1(\cdot)$ and $H_2(\cdot)$ be two cryptographic hash functions where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$.

– `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$. The PKG chooses $s \in \mathbb{F}_q^*$ randomly and sets $P_{pub} = sP$. The master secret key is s, which is kept secret and known by the PKG only. The system parameters are

$$\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}, \hat{e}(\cdot,\cdot), H_1(\cdot), H_2(\cdot)\}.$$

– `Extract`: Same as the private key extraction algorithm defined in Section 3.2.

– `Sign`: For the user ID with secret key S_{ID} to sign on a message m, he/she follows the steps below.
  1. Randomly choose z from $\mathbb{F}_q^*$.
  2. Compute z^{-1} (Both z and z^{-1} are kept in secret).
  3. Compute $T = z^{-1}H_2(m)$.
  4. Compute $U = zP_{pub}$.
  5. Compute $V = zS_{ID}$.
  6. The final signature is (T, U, V).

– `Verify`: After receiving (T, U, V) from the signer, any verifier performs the following to verify the signature.
  1. Check whether $\hat{e}(V, P) = \hat{e}(U, Q_{ID})$.
  2. Check whether $\hat{e}(U, T) = \hat{e}(P_{pub}, H_2(m))$.
  3. If both of the equalities hold, the signature is considered to be valid.

5.4 Security Analysis

Theorem 1 *In the random oracle model, if a PPT forger $\mathcal{A}$ can $(t, q_S, q_H, q_E, \epsilon)$ breaks our scheme, then the CBDH problem can be solved with an advantage ϵ' within a time $t' < t + o(t)$.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ that has advantage ϵ in attacking our IBS. We can show that an algorithm $\mathcal{C}$ can be constructed to solve the CBDHP in $\mathbb{G}_1$. That is, given (P, aP, bP, cP), algorithm $\mathcal{C}$ is able to compute (or output) $\hat{e}(P,P)^{abc}$, with the assumption that $\mathcal{A}$ will ask for $H_1(ID)$ before ID is used in any other query. During the game, $\mathcal{A}$ will consult $\mathcal{C}$ for answers to the random oracles H_1 and H_2, $\mathcal{C}$ will keep lists L_1 and L_2 to store the answers used respectively. Let $P_{pub} = bP$. Algorithm $\mathcal{C}$ interacts with $\mathcal{A}$ in the EF-CMIA game as follows:

Queries on oracle H_1 for identity:
We embed part of the challenge aP in the answer of many H_1 queries [14]. When $\mathcal{A}$ asks queries on the hash value of identity ID, $\mathcal{C}$ picks $y_1 \in_R \mathbb{F}_q^*$ and repeats the process until y_1 is not in the list L_1. $\mathcal{C}$ then flips a coin $W_1 \in \{0,1\}$ that yields 0 with probability ζ_1 and 1 with probability $1 - \zeta_1$. (ζ_1 will be determined later.) If $W_1 = 0$ then the hash value $H_1(ID)$ is defined as y_1P; else if $W_1 = 1$ then returns $H_1(ID) = y_1(aP)$. In either case, $\mathcal{C}$ stores (ID, y_1, W_1) in the list L_1.

Queries on oracle H_2 for message:
This time we embed the remaining part of the challenge cP in the answer of many H_2 queries. When $\mathcal{A}$ asks queries on the hash value of message m, $\mathcal{C}$ picks $y_2 \in_R \mathbb{F}_q^*$ and repeats the process until y_2 is not in the list L_2. $\mathcal{C}$ then flips a coin $W_2 \in \{0,1\}$ that yields 0 with probability $1 - \zeta_2$ and 1 with probability ζ_2. (ζ_2 will be determined later.) If $W_2 = 0$ then the hash value $H_2(m)$ is defined as $y_2(cP)$; else if $W_2 = 1$ then returns $H_2(m) = y_2P$. In either case, $\mathcal{C}$ stores (m, y_2, W_2) in the list L_2.

Private key extraction queries:
When $\mathcal{A}$ asks for the private key of user ID, $\mathcal{C}$ looks for (ID, y_1, W_1) entry in L_1. If $W_1 = 1$ then $\mathcal{C}$ outputs "failure" and halts since $\mathcal{C}$ does not know how to compute y_1abP. Otherwise a valid private key $y_1(bP)$ is returned.

Sign queries:
When $\mathcal{A}$ asks for a signature of user ID on message m, $\mathcal{C}$ looks for (ID, y_1, W_1) entry in L_1. If $W_1 = 0$, no matter what the value $H_1(m)$ is, $\mathcal{C}$ output the signature using the `Sign` algorithm since $\mathcal{C}$ knows the corresponding private key. If $W_1 = 1$, $\mathcal{C}$ checks for the list L_2 again. If $(m, y_2, 0)$ is found in the list, $\mathcal{C}$ fails and halts; otherwise $\mathcal{C}$ will pick $x \in_R \mathbb{F}_q^*$. The signature to be output is ($T = x^{-1}y_2(bP)$, $U = xP$, $V = xy_1aP$). It is easy to see that the above signature is valid.

Forgery:
Algorithm $\mathcal{A}$ returns a forgery (ID, m, T, U, V) such that (T, U, V) is a valid signature on m by user ID. We neglect the rare case (the probability is at most $1/2^k$, where k is the security parameter) that $\mathcal{A}$ makes a valid forgery without asking for the value of $H_2(m)$ from $\mathcal{C}$.

Probability of success:
$\mathcal{C}$ can solve the CBDHP if $\mathcal{A}$ have made a forged signature on message m for user ID where $(ID, y_1, 1)$ is in the list L_1 and $(m, y_2, 0)$ is in the list L_2, otherwise $\mathcal{C}$ cannot compute $\hat{e}(P, P)^{abc}$ from the forgery made by $\mathcal{A}$.

The probability that $\mathcal{C}$ answers to all private key extraction queries is $\zeta_1^{q_E}$, and the probability that $\mathcal{A}$ makes a forged signature for user ID where $(ID, y_1, 1)$ is in the list L_1 is $1 - \zeta_1$. Similarly, the probability that $\mathcal{C}$ answers to all sign queries is at most $\zeta_2^{q_S}$, and the probability that $\mathcal{A}$ makes a forged signature on message m where $(m, y_2, 0)$ is in the list L_2 is $1 - \zeta_2$. Hence the probability for $\mathcal{C}$ to solve CBDHP successfully is $f_{q_E}(\zeta_1)f_{q_S}(\zeta_2)$ where $f_x(\zeta) = \zeta^x(1 - \zeta)$.

Simple differentiation shows that $f_x(\zeta)$ is maximized when $\zeta = 1 - (x+1)^{-1}$, and the corresponding probability is $\frac{1}{x}(1 - \frac{1}{x+1})^{x+1}$. So the maximum probability for $\mathcal{C}$ to solve CBDHP successfully is

$$\frac{1}{q_S q_E}(1 - \frac{1}{q_E + 1})^{q_E+1}(1 - \frac{1}{q_S + 1})^{q_S+1}$$

For large q_S and q_E, this probability is equal to $1/e^2 q_S q_E$.

Solving CBDHP:
If $\mathcal{A}$ makes a forged signature on message m for user ID where $(ID, y_1, 1)$ is in the list L_1 and $(m, y_2, 0)$ is in the list L_2, T is in the form of $z^{-1}y_2(cP)$ and V is in the form of $zy_1(abP)$, hence $\mathcal{C}$ can solve CBDHP by computing $\hat{e}(V, T)^{y_1^{-1}y_2^{-1}} = \hat{e}(zabP, z^{-1}cP) = \hat{e}(P, P)^{abc}$ □

5.5 Relationship with Verifiable Pairing

The above formal security analysis does not only show the unforgeability of our IBS. Notice that our IBS is indeed the same as our verifiable pairing, with the role of cP being replaced by $H_2(m)$. In the proof of Theorem 1 we proved that if the invocation of verifiable pairing algorithm enables one to compute $\hat{e}(abP, R)$ for any arbitrary $R \in \mathbb{G}_1$, then our IBS is forgeable, which is contradictory to our proof. One may argue that it is generally insufficient to prove that an adversary cannot compute $\hat{e}(abP, R)$ for a randomly-chosen R since partial information about abP might be leaked. However, our proof on the existential unforgery of our IBS has shown that the it is not the case, otherwise our IBS is universally forgeable since the knowledge of private key is leaked.

From the security analysis we also know that without knowing the private key, one cannot execute the algorithm of verifiable pairing to enable another party to calculate the value of $\hat{e}(abP, cP)$, which is also an essential property of verifiable pairing.

6 ID-Based Signcryption

6.1 Overview

Signcryption scheme (e.g. [25, 30]) is a cryptographic primitive that combines encryption and signing in one step at a lower computational cost.

An ID-based signcryption scheme consists of five algorithms: `Setup`, `Extract`, `Signcrypt`, `Unsigncrypt` and `TP_Verify` (if public verifiability is satisfied). In essence, `Setup` generates the common public parameters and the master secret key depending on the security level parameter; `Extract` generates the private key for each user according to the user's public identity; `Signcrypt` produces the ciphertext from a sender to a designated recipient; `Unsigncrypt` recovers the original message after checking its integrity and origin; `TP_Verify` enables any third party to verify the integrity and the origin of the message.

An ID-based signcryption scheme is *publicly verifiable* if given a message m, a signcrypted message σ, and possibly some *additional information* τ provided by the recipient, a third party can verify that σ is a valid signature of the sender for m, without knowing the recipient's private key[2]. This property makes the party who makes the signcryption not able to repudiate to anybody afterwards. There are not many ID-based signcryption schemes having this feature, one of which is [22].

Obviously, we need some mechanisms to check for the correctness of τ since it may be the point of attack for a dishonest recipient to forge a valid-looking signcrypted message. This is where our verifiable pairing comes to play.

6.2 Libert and Quisquater's Scheme

We first review Libert and Quisquater's signcryption scheme [22]. Let H_1, H_2 and H_3 be three cryptographic hash functions where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ and $H_3 : \{0,1\}^n \times \mathbb{G}_2 \rightarrow \mathbb{F}_q^*$. Let E, D be the encryption and decryption algorithms of

[2] Note that we adopt a different definition from that in [20].

a secure symmetric cipher which takes a plaintext/ciphertext of length n respectively, and also a key of length n. (For example, a one-time pad cipher.) The following shows the details of the scheme.

- `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$. The PKG chooses $s \in \mathbb{F}_q^*$ randomly and sets $P_{pub} = sP$. The master secret key is s, which is kept secret and known by the PKG only. The system parameters are

$$\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}, \hat{e}(\cdot,\cdot), H_1(\cdot), H_2(\cdot), H_3(\cdot,\cdot), E_{(\cdot)}(\cdot), D_{(\cdot)}(\cdot)\}.$$

- `Extract`: Same as the private key extraction algorithm defined in Section 3.2.
- `Signcrypt`: To send a message $m \in \{0,1\}^n$ to B, A follows the steps below.
  1. Choose x from $\mathbb{F}_q^*$ randomly.
  2. Compute $k_1 = \hat{e}(P, P_{pub})^x$ and $k_2 = H_2[\hat{e}(P_{pub}, Q_{ID_B})^x]$.
  3. Compute $c = E_{k_2}(m)$.
  4. Compute $r = H_3(c, k_1)$.
  5. Compute $S = xP_{pub} - rS_{ID_A}$.
  6. The ciphertext is $\sigma = (c, r, S)$.
- `Unsigncrypt`: To unsigncrypt a signcrypted message (c, r, S) from A, B follows the steps below.
  1. Compute $r'_A = rQ_{ID_A}$.
  2. Compute $k'_1 = \hat{e}(P, S)\hat{e}(P_{pub}, r'_A)$.
  3. Compute $k'_2 = H_2[\hat{e}(S, Q_{ID_B})\hat{e}(r'_A, S_{ID_B})]$.
  4. Recover $m = D_{k'_2}(c)$.
  5. Compute $r' = H_3(c, k'_1)$.
  6. Accept the message if and only if $r' = r$, return $Invalid$ otherwise.
  7. Give (k'_2, m, σ) to the third party.
- `TP_Verify`:
  1. Compute $k'_1 = \hat{e}(P, S)\hat{e}(P_{pub}, Q_{ID_A})^r$.
  2. Compute $r' = H_3(c, k'_1)$.
  3. Accept the origin of ciphertext if and only if $r = r'$.
  4. Moreover, accept the message authenticity if and only if $m = D_{k'_2}(c)$.
  5. Return $Valid$ if all tests are passed, $Invalid$ otherwise.

6.3 Against Existential Forgery of Dishonest Recipient by Verifiable Pairing

Since the third party has no way to ensure the correctness of session key k'_2 obtained from the recipient, dishonest recipient can randomly choose k'_2 such that the signcrypted message (c, r, S) decrypts to a plaintext m' which is not equal to $D_{H_2[\hat{e}(S,Q_{ID_B})\hat{e}(r'_A,S_{ID_B})]}(c)$. This issue was not addressed in [22]. A simple fix to this attack is to disclose the recipient's decryption key to the third party, but this makes the scheme rather inflexible and unrealistic.

Now we present modifications to Libert and Quisquater's scheme which make their scheme secure against this attack. In the modifications, recipient executes our verifiable pairing protocol with the third party, i.e. apart from the signcrypted message (c, r, S), recipient also sends $T = z^{-1}S$, $U = zP_{pub}$ and zS_{ID_B} to the third party. This does not compromise the recipient's decryption key and only enables the third party to decrypt signcrypted messages in the form of $(c^{\#}, r^{\#}, S^{\#})$ where $S^{\#} = S$, which is a rare case as S can be considered as randomly generated by the sender. The third party can compute a correct k_2' by itself from the formula $k_2' = H_2[\hat{e}(T, V)\hat{e}(r_A', Q_{ID_B})]$.

After our modifications, the third party must get a session key and hence a correct decryption of the signcrypted message, which ensures the non-repudiation property of the scheme.

7 ID-Based Undeniable Signature

7.1 Overview

Digital signature is an important cryptographic primitive. A digital signature binds a signer to an e-document. The validity of the digital signature can be verified by any person who receives it without any help from the signer. This feature is undesirable in some applications. For example, Bob can show a signed love letter from Alice to a third party without the consent of Alice and Alice cannot deny to be the author of the letter. To overcome this deficiency of digital signature, Chaum *et al.* introduced the *undeniable signature* [8]. To verify an undeniable signature, the verifier must go through an interactive protocol with the signer.

At the fourth ACM conference on electronic commerce (EC'03), Han *et al.* proposed the first ID-based undeniable signature scheme[3] [18], but their scheme is not secure against the denial attack and the forge attack [29]. In this section, we add verifiable pairing to Han *et al.*'s scheme and propose an enhanced scheme that is secure against Zhang *et al.*'s attacks.

7.2 Han *et al.*'s Scheme

We first review Han *et al.*'s ID-based undeniable scheme using their notations [18]. Then, we show that the changes they proposed to the scheme to prevent the signer from denying a valid signature do not work.

Their Scheme: Let H_1, H_2 be two cryptographic hash functions where $H_1 : \{0,1\}^* \rightarrow \mathbb{F}_q$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$. Let A be a larger number about 10^{20} and $[A] = \{1, 2, \cdots, A\}$.

– `Setup`: Let P be the generator of $\mathbb{G}_1$. The PKG chooses $s \in \mathbb{F}_q^*$ randomly. It sets $P_{pub} = sP$. The master secret key is s, which is kept secret and known by itself only. The system parameters are

$$\{\mathbb{G}_1, \mathbb{G}_2, q, P, P_{pub}, \hat{e}(\cdot,\cdot), H_1(\cdot), H_2(\cdot), A\}.$$

– `Extract`: Similar to the private key extraction algorithm defined in Section 3.2, but there is one more component in the private key. Signer with identity $ID \in \{0,1\}^*$ submits ID to the PKG. The PKG sets the signer's public key Q_{ID} to be $H_2(ID) \in \mathbb{G}_1$, computes the signer's private key (D_{ID}, L_{ID}) by $D_{ID} = sQ_{ID}$ and $L_{ID} = s^{-1}Q_{ID}$. Then the PKG sends the private key to the signer.

[3] The authors claimed that the scheme is a confirmer signature scheme, but it is actually an undeniable signature scheme (as this has also been pointed out by Zhang *et al.* in [29]).

- `Signing`: To sign a message $m \in \{0,1\}^*$, signer chooses a random $k \in \mathbb{F}_q^*$, and sets the alleged signature to be $\{r, S\} = \{kP, k^{-1}D_{ID} + H_1(m)L_{ID}\}$.
- `Confirmation`: To confirm an alleged signature $\{r, S\}$ for a message m,
  1. Verifier randomly chooses $x \in [A]$, $y \in \mathbb{F}_q^*$, and sends $C_1 = xyr$, $C_2 = xyP$ to signer.
  2. Signer computes $X = \hat{e}(r + P_{pub}, P - L_{ID})$ and $R = \hat{e}(C_1, L_{ID})$, then sends them to verifier.
  3. Verifier checks whether
  $$\hat{e}(r, S)^x = \hat{e}(P_{pub}, Q_{ID})^x R^{H_1(m)y^{-1}},$$
  $$R^{y^{-1}} X^x \hat{e}(P, Q_{ID})^x = \hat{e}(r + P_{pub}, P)^x.$$
  If all of the equalities hold, then verifier accepts the signature as valid. Otherwise, the validity of the signature remains undetermined.
- `Denial`:
  1. Verifier randomly chooses $x \in [A]$, $y \in \mathbb{F}_q^*$, and sends $C_1 = xyr$, $C_2 = xyP$ to signer.
  2. Signer computes $B = \frac{\hat{e}(C_1, S)}{\hat{e}(C_2, D_{ID})\hat{e}(C_1, L_{ID})^{H_1(m)}}$ and sends it to verifier.
  3. Verifier calculates the inverse of y and sends $C = B^{y^{-1}}$ to signer.
  4. Signer computes x' from C by computing $\frac{\hat{e}(r,S)}{\hat{e}(P_{pub}, Q_{ID})\hat{e}(r, L_{ID})^{H_1(m)}}$ and sends x' to verifier.
  5. Verifier checks whether $x' = x$. If the equality holds, verifier accepts the signature as invalid. Otherwise, the invalidity of the signature remains undetermined.

A Minor Flaw: The scheme described in the previously cannot prevent a signer from denying a valid signature. The reason behind is that verifier has no way to verify the values of D_{ID} and L_{ID} used in B are valid based only on the value of B. So, the authors proposed some changes to the denial protocol to handle this problem. We find that the proposed changes will make the scheme fail in handling an invalid signature.

Their Proposed Changes: In Step 2 of the denial protocol, in addition to B, the signer also sends $G = \hat{e}(C_2, D_{ID})$ and $R = \hat{e}(C_1, L_{ID})$ to the verifier. The verifier can check whether $\hat{e}(r, S)^x = G^{y^{-1}} R^{H_1(m)y^{-1}}$. If this equality does not hold, the verifier aborts the protocol and concludes that the signer is lying.

The Flaw of Proposed Changes: If the signature $\{r, S\}$ is valid, the changes work. However, if $\{r, S\}$ is an invalid signature, the equality does not hold even if both G and R are valid. In other words, the signer cannot deny an invalid signature based on the changes proposed even if the signer complies legally with the protocol.

7.3 Zhang *et al.*'s Attacks

In this section, we review the two attacks given by Zhang *et al.* [29], which are based on the Han *et al.*'s scheme without the changes described in previous subsection.

The Denial Attack: Denial attack is an attack launched by the signer to deny a valid signature. The proposed attack is given as follows. After the verifier sends C_1 and C_2

to the signer, the signer picks $\alpha \in \mathbb{F}_q^*$, computes $B = \hat{e}(C_2, \alpha P)$ and sends it to the verifier.

Then the verifier sends $C = B^{y^{-1}}$ to the signer according to the denial protocol. As $C = \hat{e}(P, \alpha P)^x$, the signer can find x' from $[A]$ such that $C = \hat{e}(P, \alpha P)^{x'}$, the verifier is convinced that the alleged signature $\{r, S\}$ is not created by the signer.

The Forge Attack: Forge attack is an attack launched by an entity to forge a signature with an arbitrary identity ID on any message m. The steps are given as follows. To forge a signature for message $m \in \{0,1\}^*$, the attacker $\mathcal{A}$ picks $\beta \in \mathbb{F}_q^*$ randomly in addition to the k as in the original signing step. Without the knowledge of D_{ID} and L_{ID}, $\mathcal{A}$ forms the alleged signature $\{r, S\}$ by computing $r = kP_{pub}$ and $S = k^{-1}(Q_{ID} + \beta H_1(m)P)$.

In the confirmation protocol, after the verifier sends C_1 and C_2 to $\mathcal{A}$, $\mathcal{A}$ computes X and R by the equations $X = \hat{e}(r + P_{pub}, P)\hat{e}(P, Q_{ID})^{-1}\hat{e}(P_{pub}, \beta P)^{-1}$ and $R = \hat{e}(\beta P_{pub}, C_2)$, then $\mathcal{A}$ sends them to the verifier. It can be shown that both equalities $\hat{e}(r, S)^x = \hat{e}(P_{pub}, Q_{ID})^x R^{H_1(m)y^{-1}}$ and $R^{y^{-1}} X^x \hat{e}(P, Q_{ID})^x = \hat{e}(r + P_{pub}, P)^x$ hold. In other words, the verifier will be convinced that the signature $\{r, S\}$ for a message m is a valid signature of the signer with identity ID.

7.4 Our Enhanced Scheme

– `Setup`: Same as Han *et al.*'s scheme. In addition, the PKG sets $P_{inv} = s^{-1}P$ and publishes it. i.e. The system parameters are

$$\{\mathbb{G}_1, \mathbb{G}_2, q, P, P_{pub}, P_{inv}, \hat{e}(\cdot,\cdot), H_1(\cdot), H_2(\cdot), A\}.$$

– `Extract` and `Signing`: Same as Han *et al.*'s scheme.

– `Confirmation` and `Denial`: To confirm or deny a signature $\{r, S\}$ for a message m,

1. Verifier chooses $x \in [A], y \in \mathbb{F}_q^*$ uniformly and randomly, and sends $C_1 = xyr = xykP$, $C_2 = xyP$ to signer.
2. Signer chooses $z \in \mathbb{F}_q^*$ uniformly and randomly, sets $X = \hat{e}(r + P_{pub}, P - L_{ID})$, $T = z^{-1}C_1$, $U = zP_{inv}$, $V = zL_{ID}$ and sends them to verifier.
3. Verifier checks the validity of U and V by checking whether $\hat{e}(V, P) = \hat{e}(U, Q_{ID})$.
4. Verifier checks the validity of U and T by checking whether $\hat{e}(U, T) = \hat{e}(P_{inv}, C_1)$.
5. If not all of the equalities hold, then verifier will consider signer lying and the verification procedures will be aborted; otherwise, verifier computes $R = \hat{e}(V, T) = \hat{e}(C_1, L_{ID})$.
6. To confirm a valid signature, the signer computes $X = \hat{e}(r + P_{pub}, P - L_{ID})$, verifier checks the validity of signature by checking whether

$$\hat{e}(r, S)^x = \hat{e}(P_{pub}, Q_{ID})^x R^{H_1(m)y^{-1}},$$
$$R^{y^{-1}} X^x \hat{e}(P, Q_{ID})^x = \hat{e}(r + P_{pub}, P)^x.$$

If all of the equalities hold, then verifier accepts the signature as valid. Otherwise, the validity of the signature is undetermined.

7. To deny an invalid signature,
   (a) Verifier computes $B = \frac{\hat{e}(C_1,S)}{\hat{e}(xyP_{pub},Q_{ID})R^{H_1(m)}}$ and sends $C = B^{y^{-1}}$ to signer.
   (b) Signer computes x' from C by computing $\frac{\hat{e}(r,S)}{\hat{e}(P_{pub},Q_{ID})\hat{e}(r,L_{ID})^{H_1(m)}}$ and sends x' to verifier.
   (c) If $x' = x$, verifier accepts the signature as invalid. Otherwise, the invalidity is undetermined.

7.5 Security Analysis

The denial attack given in [29] is prevented since R is calculated by verifier instead of by signer. Although R is calculated based on the information (T, U and V) provided by signer, signer cannot cheat by providing "invalid" T, U and V that can pass the validity check of verifier. Similarly, the forge attack given in [29] is prevented since the attack is made possible by setting $R = \hat{e}(\beta P_{pub}, C_2)$ where β is chosen by the attacker. Moreover, the private key of signer will not be compromised since verifier does not know z.

7.6 Convertible Undeniable Signature

We say that an undeniable signature scheme is *convertible* if the alleged signature can be converted into an universally verifiable signature by the signer.

As a bonus result, we show how to make our scheme convertible. The signer just chooses k in a way that is recoverable by the signer only for each message to be signed instead of a random one (e.g. setting $k = H_1(m||D_{ID}||L_{ID})$). Releasing k turns the alleged signatures into ordinary digital signatures since verifier can prove the validity of the alleged signature by showing that $\hat{e}(r, S)=\hat{e}(P_{pub},Q_{ID})\hat{e}(kP_{inv},H_1(m)Q_{ID})$.

8 Conclusion

We introduce the notion of *Verifiable Pairing*, together with a concrete construction, to ensure the security or the robustness of many cryptographic protocols, including (but not necessarily limited to) threshold decryption, signcryption and undeniable signature. As a bonus result, our scheme for verifiable pairing gives rise to an identity-based signature that is provably secure in the random oracle model without using the forking lemma, assuming the hardness of the computational bilinear Diffie-Hellman problem.

Acknowledgement

The author is grateful to Dr. S.M. Yiu for giving helpful suggestions, Dr. Lucas C. K. Hui and Dr. K.P. Chow for their support, Ms. H.W. Go for her comments on the camera-ready version of the paper, and anonymous reviewers for their improvement on the scheme.

References

1. Paulo S.L.M. Barreto, Hae Y. Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In Moti Yung, editor, *Advances in Cryptology: Proceedings of CRYPTO 2002 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18-22, 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag Heidelberg, 2002.
2. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2004. Available at http://www.cs.stanford.edu/ xb/crypto04a.
4. Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag Heidelberg, 2001.
5. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
6. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.
7. Jae Choon Cha and Jung Hee Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups . In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2002.
8. David Chaum and Hans Van Antwerpen. Undeniable Signatures. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer, 1990.
9. YoungJu Choie and Eunjeong Lee. Implementation of Tate Pairing of Hyperelliptic Curves of Genus 2. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003, 6th International Conference Seoul, Korea, November 27-28, 2003, Revised Papers*, volume 2971 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2003.
10. Sherman S.M. Chow, Lucas C.K. Hui, and S.M. Yiu. Identity Based Threshold Ring Signature. Cryptology ePrint Archive, Report 2004/179, July 2004. Available at http://eprint.iacr.org.
11. Sherman S.M. Chow, Lucas C.K. Hui, S.M. Yiu, and K.P. Chow. Secure Hierarchical Identity Based Signature and its Application. In *6th International Conference on Information and Communications Security (ICICS04)*, Lecture Notes in Computer Science, Malaga, Spain, October 2004. Springer-Verlag. To Appear.

12. Sherman S.M. Chow, Lucas C.K. Hui, S.M. Yiu, and K.P. Chow. Two Improved Partially Blind Signature Schemes from Bilinear Pairings. Cryptology ePrint Archive, Report 2004/108, April 2004. Available at http://eprint.iacr.org.
13. Sherman S.M. Chow, S.M. Yiu, Lucas C.K. Hui, and K.P. Chow. Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003, 6th International Conference Seoul, Korea, November 27-28, 2003, Revised Papers*, volume 2971 of *Lecture Notes in Computer Science*, pages 352–369. Springer, 2003.
14. Jean-Sébastien Coron. On the Exact Security of Full Domain Hash. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.
15. Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.
16. Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the Tate Pairing. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.
17. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002. Available at http://eprint.iacr.org.
18. Song Han, Winson K.Y. Yeung, and Jie Wang. Identity-based Confirmer Signatures from Pairings over Elliptic Curves. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 262–263. ACM Press, 2003.
19. Tetsuya Izu and Tsuyoshi Takagi. Efficient Computations of the Tate Pairing for the Large MOV Degrees. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 283–297. Springer-Verlag Heidelberg, 2003.
20. Ik Rae Jeong, Hee Yun Jeong, Hyun Sook Rhee, Dong Hoon Lee, and Jong In Lim. Provably Secure Encrypt-then-Sign Composition in Hybrid Signcryption. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 16–34. Springer-Verlag Heidelberg, 2003.
21. Benoît Libert and Jean-Jacques Quisquater. Efficient Revocation and Threshold Pairing Based Cryptosystems. In *Proceedings of 22nd Annual Symposium on Principles of Distributed Computing*, pages 163–171. ACM Press, 2003.
22. Benoît Libert and Jean-Jacques Quisquater. New Identity Based Signcryption Schemes from Pairings. In *IEEE Information Theory Workshop*, pages 155–158, 2003. Full Version Available at http://eprint.iacr.org.
23. Benoît Libert and Jean-Jacques Quisquater. Identity Based Undeniable Signatures. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 112–125. Springer, 2004.

24. Benoît Libert and Jean-Jacques Quisquater. The Exact Security of an Identity Based Signature and its Applications. Cryptology ePrint Archive, Report 2004/102, 2004. Available at http://eprint.iacr.org.
25. H. Petersen and M. Michels. Cryptanalysis and Improvement of Signcryption Schemes. *IEE Proceedings - Computers and Digital Techniques*, 145(2):149–151, 1998.
26. David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 13(3):361–396, 2000.
27. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO 1984, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 19–22 August 1985.
28. Fangguo Zhang and Kwangjo Kim. Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, volume 2727 of *Lecture Notes in Computer Science*, pages 312–323. Springer, 2003.
29. Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. Attack on Han *et al.*'s ID-based Confirmer (Undeniable) Signature at ACM-EC'03. Cryptology ePrint Archive, Report 2003/129, 2003. Available at http://eprint.iacr.org.
30. Yuliang Zheng. Digital Signcryption or How to Achieve Cost (Signature & Encryption) $<<$ Cost(Signature) + Cost(Encryption). In Burton S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO 1997 5th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.

Improving the Performance of Signature-Based Network Intrusion Detection Sensors by Multi-threading[*]

Bart Haagdorens[1,**], Tim Vermeiren[2], and Marnix Goossens[1]

[1] TELE Group, ETRO Department,
Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
{bart.haagdorens,marnix.goossens}@vub.ac.be
[2] R&I, Alcatel Bell, Francis Wellesplein 1, B-2018 Antwerp, Belgium
tim.vermeiren@alcatel.be

Abstract. Signature-based Network Intrusion Detection System (NIDS) sensors match network packets against a pre-configured set of intrusion signatures. Current implementations of NIDS sensors employ only a single thread of execution and as a consequence benefit very little from multi-processor hardware platforms. A multi-threaded sensor would allow more efficient and scalable exploitation of these multi-processor machines. We present in detail a number of novel designs for a multi-threaded NIDS sensor and provide performance evaluation figures for a number of multi-threaded implementations of the popular open-source Snort system.

1 Introduction

A signature-based NIDS sensor analyzes packets from a computer network and matches them against a set of signatures that trigger on known intruder techniques. In a typical setup, a NIDS *sensor* analyzes all packets flowing through some point in the network. The sensor detects whether the packets match any of its signatures and delivers matching packets to on analysis backend.

The performance of the detection process is an important metric in the evaluation of a NIDS. An overloaded system will eventually drop incoming packets and fail to scan all traffic.

Intrusion detection systems are often implemented in software on general-purpose Central Processing Units (CPUs), often on off-the-shelf server hardware. These systems offer high performance at moderate cost, are flexible and allow easy maintenance. However, current NIDS sensors are implemented as a single thread of execution and this limits their performance to the performance of a single CPU, even if this CPU is part of a multi-processor platform.

[*] The authors would like to thank IWT - Vlaanderen for its support in the PANEL framework.

[**] Bart Haagdorens is an *aspirant* at FWO – Vlaanderen.

In this paper, we present a new approach to improve NIDS sensor performance beyond the single CPU performance limit. Small-scale multi-processor computing platforms have been around for a while now and operating system support for them has matured. In our opinion, these platforms are excellent candidates to run a compute-intensive job as a NIDS sensor. To exploit the capabilities of multiple processors, we develop a multi-threaded NIDS sensor. This allows to spread the signature matching workload over multiple CPUs of the same machine and performance to scale beyond single-CPU machines.

Paper Organization

In section 2 we discuss some related efforts to increase NIDS sensor performance. In section 3, we show a general model for a single-threaded signature-based NIDS sensor. Section 4 discusses new elements needed to evolve towards a multi-threaded implementation and presents five multi-threaded designs for a NIDS sensor. Section 5 presents the algorithm we implemented to guarantee the Order-of-Seniority Processing (OoSP) of packets when multiple threads are processing packets in parallel. In section 6, we evaluate our multi-threaded implementations on different hardware configurations. We show that our approach can give a 16% performance increase on a dual CPU platform, and provide some results that indicate further scalability to platforms with more than two CPUs. We conclude the article with a discussion of the results and suggestions for future work.

2 Related Work

The performance of signature-based NIDS sensors has certainly received attention from the research community over the last couple of years. Many research papers use the popular open-source Snort implementation [1, 2] for analysis and evaluation of their ideas, although the results are probably more generally applicable. We will also use modified Snort implementations to evaluate our ideas in section 6.

2.1 Multi-processor and Multi-threading

Analysis of the performance of software NIDS sensors on server hardware has shown that it is very dependent on the details of the hardware (processor architecture, clock speed, cache size, ...) [3]. It also showed that existing single-threaded sensor implementations only improve slightly when executed on multi-processor platforms. The small improvement is achieved because interrupts related to packet reception can be handled by the Operating System (OS) kernel on one processor, while the NIDS application software runs on an other.

The approach of a multi-threaded NIDS sensor has been explored only partially up till now: some work has been done to separate signature matching on the one hand and output or storage of detected intrusions on the other into different threads or processes running on the same machine [4, 5]. By isolating the output in a separate thread of execution, the signature matching thread is no longer hindered by the I/O latency caused by storing the detected intrusions on disk or in a (remote) database . However, this technique employs multi-threading to isolate latencies, not to exploit the computing power of multiple CPUs in a single machine. The workload-intensive signature matching is still in a single thread and can not be spread over multiple CPUs.

2.2 Traffic Splitting

As another approach to scale beyond the performance of a single CPU, *traffic splitters* or *IDS load balancers* have been proposed. In these setups, incoming traffic is split and sent to a number of NIDS sensors running on separate machines. These sensors work mostly independently of each other, they exchange only a limited amount of data or no data at all. Though some research [6, 7] and commercial [8] splitter implementations exist, the approach introduces some issues:

– all packets belonging to the same intrusion scenario, are to be sent to the same sensor. This is not a trivial issue, as it is very dependent on the type of intrusion: for example, how can packets potentially belonging to a distributed attack be identified?
– balancing the workload between sensors is not trivial either. Reference [9] studied the workload of Snort and reported the workload per packet to be spread by a factor four and the workload per byte by a factor eight, depending on the traffic characteristics. Hence, neither the bit rate nor the packet rate handled by a sensor are adequate measures to balance the workload.
– the splitting process itself is not distributed. To prevent it from becoming a bottleneck itself, the splitting decision has to be taken in a limited time.
– system setup and maintenance is more complex compared to a single sensor as the system now consists of both a splitting point and multiple separate sensors.

These issues are not entirely solved in existing splitter solutions. However, their performance seems to scale up quite well because hardware or Network Processors (NPs) can be used for the splitters.

Nevertheless we believe that a single machine with multiple CPU's under control of a single OS suffers less from each of the issues above, mainly because the CPU's have access to a shared memory where all state information can be centralized. Therefore, exchange of information between threads is easier and faster.

3 Model of a Signature-Based NIDS Sensor

Figure 1 shows a generalized model of a signature-based NIDS sensor. The solid black line represents normal sequential packet processing by the single thread

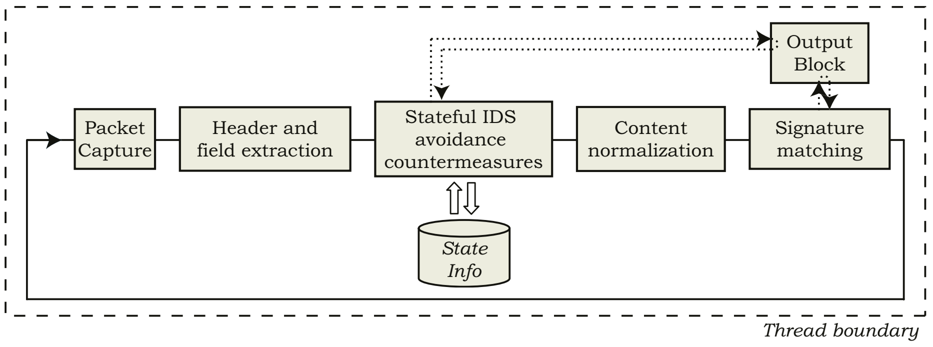


Fig. 1. Model of a single-threaded signature-based NIDS. The solid line represents normal sequential processing of packets, the dotted lines represent a call to and return from the output block in case of a detected intrusion.

of execution. An incoming packet is taken through the functional blocks consecutively. The dotted line shows the call to the output block when an intrusion is detected. It is this call that can cause high-latency I/O to become a packet processing bottleneck [4, 5].

When a packet has been completely processed (with or without intrusions being detected), a new packet is acquired for processing.

We describe the functional elements in the following sections.

3.1 Packet Capture

This block makes input network traffic available for processing. Its goal is to get a 'raw' network packet (for example a complete Ethernet packet) into a memory location that can be readily accessed by the other blocks of the NIDS sensor.

In many implementations, packet capture is handled through the cross-platform *libpcap* [10] API and library.

3.2 Protocol Header and Field Extraction

A network packet consists of multiple encapsulated protocols. Signatures will often need to check various fields of various protocols. Therefore, a NIDS identifies protocol headers and field boundaries early on and uses them throughout the matching process. This extraction step also allows multiple link-layer protocols and complex encapsulation (caused by tunneling for example) to be made transparent for later steps in the process.

3.3 Stateful IDS Avoidance Countermeasures

A NIDS that considers incoming packets independently of each other, and hence maintains no state information across packets, can easily be evaded: by spreading elements of the signature over multiple packets (for example, through IP fragmentation or TCP segmentation), an intruder can avoid detection by a stateless

NIDS. Even a stateful NIDS can sometimes be evaded by exploiting subtle differences in the way the packet is handled by the NIDS on one hand and the targeted system at the other. Reference [11] gives a good overview of different techniques to avoid detection by both stateful and stateless NIDS.

Current NIDS systems implement a number of countermeasures to be less vulnerable to evasion. These countermeasures require *state information* to be kept across different packets of the same connection. These techniques include:

- Reassembling IP fragments into complete datagrams for signature matching.
- Reassembling TCP segments into a byte stream. The byte stream is then used for signature matching.
- Checking for and alerting on overlapping IP fragments or TCP segments with different data.
- Checking for and alerting on violations of protocol rules (for example, TCP segments that are sent outside of the TCP window).

This list is not exhaustive and we will not go into the details of each technique here. What they all have in common however, is that the integrity of their state information will need to be protected when it is accessed from multiple threads in a multi-threaded NIDS (see also section 4).

3.4 Content Normalization

Some application level protocols allow multiple equivalent representations of the same content. For example, Uniform Resource Names (URNs) as used in HTTP allow an alphanumeric character to be represented both as a single character ('A') and as the escaped hex-code of that character ('%41').

This complicates signature-matching on such content, as it is generally not feasible to have all possible representations of the same signature in the signature database. This would cause an explosion in the signature database and a prohibitive increase in the (already large) signature matching costs.

Therefore, a more interesting approach is *content normalization*: the content in an incoming packet is converted to a standard representation and all signatures are also in that same standard representation.

Content normalization is a per-packet stateless operation.

3.5 Signature Matching

In most NIDS systems, a signature is constructed as a logical conjunction of predicates. When all predicates are satisfied, the signature matches.

Predicates can be distinguished into two categories: those that are concerned with protocol headers and fields (generally up to and including the TCP/UDP layer) and those that are concerned with higher-layer protocols and the 'content' of the packets. These content predicates usually involve string search and this string matching has been identified as the most time-consuming algorithm in a signature-based NIDS: reference [12] found 31% of the Snort execution time to be caused by string matching. The complete signature matching block, including both header and content predicates, accounts for more than 50% of the execution time.

3.6 Output Block

In a typical system, the output block will generate information about the detected intrusions: a timestamp, the class and priority of the intrusion, details about the addresses involved in the intrusion, a complete packet trace of the intrusion scenario, etc. This information is either stored locally or sent to a central NIDS analysis backend.

4 Designs for a Multi-threaded NIDS

In this section, we present a number of multi-threaded designs based on the general NIDS model laid out in 3. The different designs have some elements in common, so we will discuss these elements before introducing the designs themselves.

4.1 Common Elements for the Designs

Pools. In the multi-threaded designs, multiple packets will be processed and multiple events can be pending simultaneously. To implement the (de-)allocation of packet and event memory in a flexible and efficient way, a number of buffers is allocated during the system's initialization. The buffers are initially stored in pools, so a thread can readily take buffers from a pool when it needs to and return them to the pool when they are no longer used.

Queues. A second structure that will show up in the following designs is the queue between threads. Such a FIFO queue buffers objects created by producing threads until they are taken off by a consumer thread. Multiple threads can produce and consume objects for and from the same queue. The queues are implemented as doubly-linked circular lists.

A queue facilitates asynchronously passing an object between two threads. Care needs to be taken that the queue doesn't get corrupted due to simultaneous access, but the critical section that needs protection is small.

Overhead. Multi-threaded designs will introduce some overhead as compared to the original single-threaded model. This overhead is caused by three factors. First of all, variables shared between threads (such as the queue and pools) need to be protected by synchronization primitives, for example mutexes or semaphores, that need to be locked and unlocked. Second, multiple threads are active at the same time and need to be managed by the OS. This overhead is dependent on the frequency and cost of thread switching and is mainly dependent on OS and hardware architecture. Third, multiple events and packets can be pending simultaneously which requires a more conservative design: global variables can never be used for packet- or event-dependant data.

It is important for the multi-threaded designs to limit the overhead introduced and simultaneously maximize the opportunities for concurrent execution on several CPUs.

Configurability. The multi-threaded designs introduce new parameters to configure the designs: the size of queues and pools and the number of threads working in parallel. These parameters need to be chosen with care. For example, in general it is not a good idea to have much more running threads than the system can concurrently execute on its CPUs, as this will only increase the thread switching overhead.

Whenever the design figures below show more than one thread in parallel, the designs are actually configurable for any number of parallel threads.

4.2 Design 1: Separate Output Block

Figure 2 shows design 1 where the output block is isolated into a separate thread. A FIFO queue buffers *events* (e.g. alerts about intrusion signatures) produced by the detection thread for consumption by the output thread. This design isolates the output latencies into a separate thread and is comparable to the designs in [4, 5].

4.3 Design 2: Parallel Signature Matching

Design 2 breaks the single packet processing loop and runs parallel signature matching threads. After the content normalization block, packets enter the *Matching Queue* and are taken out by a signature matching thread.

This design allows the most time-consuming block to be executed in parallel threads potentially running on different CPUs. It does not introduce a lot of

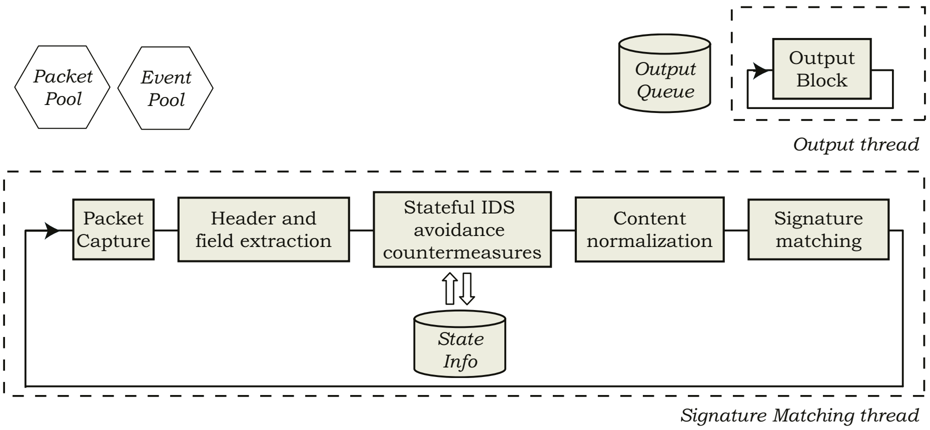


Fig. 2. Design 1.

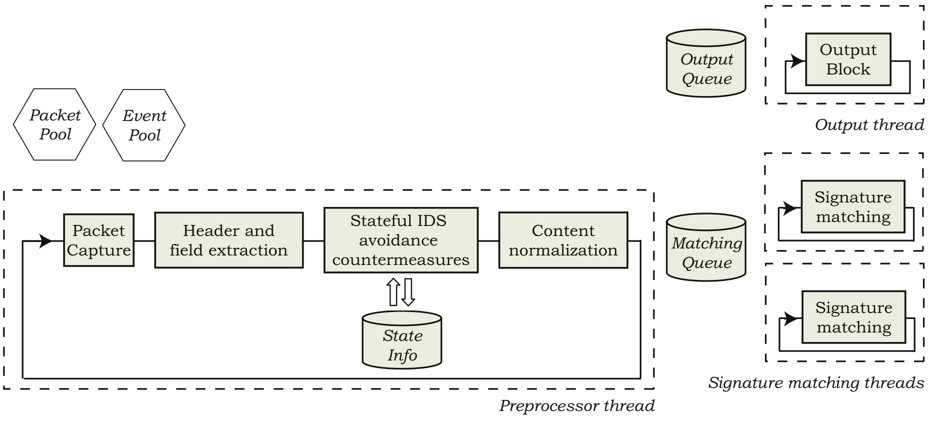


Fig. 3. Design 2.

extra overhead compared to design 1, since the signature matching block does not depend on state information.

4.4 Design 3: Parallel Content Normalization and Signature Matching

Design 3 is a slight variation on design 2. The content normalization blocks are also moved into the signature matching threads, to allow a little extra code to be executed in parallel threads. This change introduces no extra overhead.

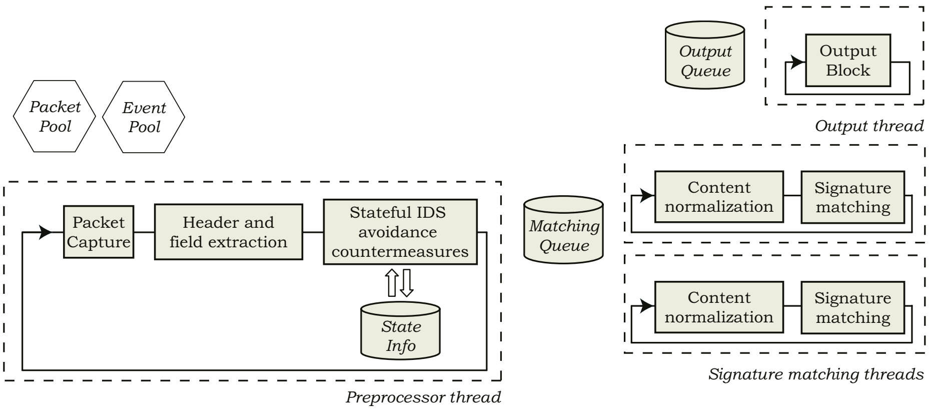


Fig. 4. Design 3.

4.5 Design 4: Parallel Stateful Countermeasures, Content Normalization and Signature Matching

Design 4 moves the stateful NIDS avoidance countermeasures into the signature matching loops. This introduces two additional issues.

First, the same state information is now potentially accessed from multiple threads. Therefore, synchronization constructs such as mutexes need to be added to the code to prevent the shared state information from being corrupted. The granularity of the synchronization (the amount of data protected by a single mutex, which influences how often and how long that mutex is locked) is a difficult issue when implementing this protection. Fine-grained protection typically causes more overhead from frequently locking and unlocking mutexes, but requires threads to wait less often and less long to enter a critical section. Coarse-grained protection causes less overhead and is easier to implement, but typically causes threads to wait more often and longer. This granularity issue is very dependent on the structure of the state information and the algorithms used to access it. We explain the granularity used in the evaluation implementations in section 6.1.

Second, because multiple threads execute in parallel it is now possible that the stateful algorithms do not process all packets in the order they originally arrived in. To guarantee the Order-of-Seniority Processing (OoSP) in design 4, we added an OoSP conflict resolution block. The OoSP issue and the algorithm used will be discussed separately in section 5. To avoid confusion in design 5, the queue has been renamed to *Capture Queue*.

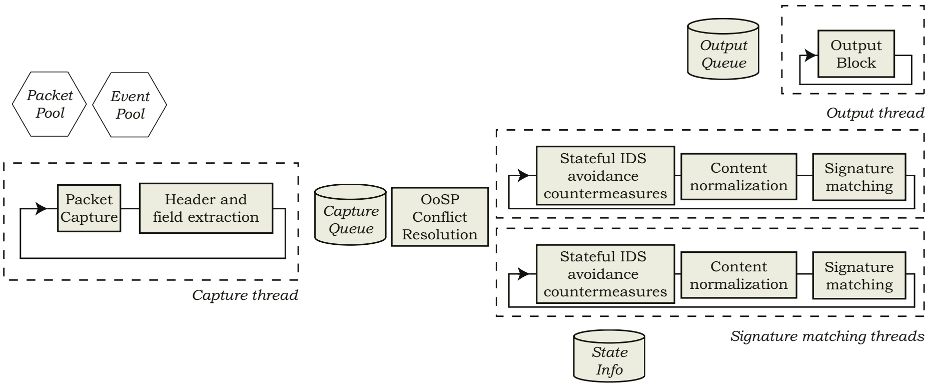


Fig. 5. Design 4.

4.6 Design 5: Parallel Stateful Countermeasures with Separate Parallel Content Normalization and Signature Matching

Design 5 restores the matching queue between the stateful IDS avoidance countermeasures and the content normalization blocks. This design limits the impact of the OoSP requirement imposed by the countermeasures block: after the packets pass through the countermeasures block, they are stored in the matching queue and can be further processed without enforcing the OoSP requirement.

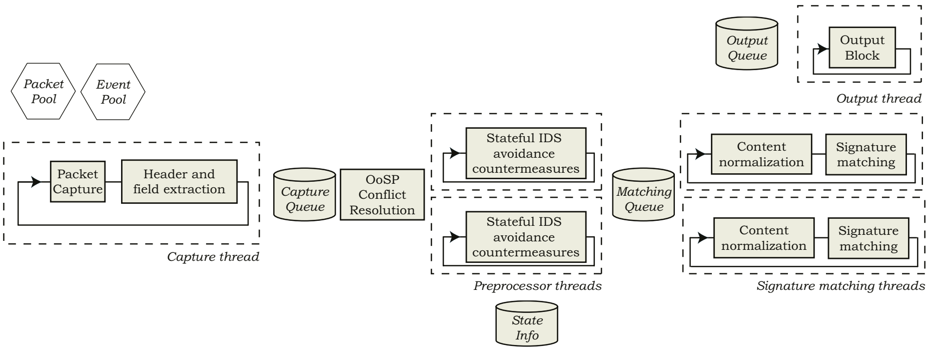


Fig. 6. Design 5.

5 Order-of-Seniority Processing

5.1 Problem Statement

Order-of-seniority-processing is the requirement that packets are processed in the order they originally arrived in. In the NIDS sensors, only the NIDS avoidance countermeasures impose this requirement because of their stateful nature. A stateful algorithm such as TCP anomaly detection for example can behave erratically when the segments of the TCP three-way handshake are not processed in their original order. A NIDS sensor located between two communicating hosts will capture the packets in their original order, but if it does not process these packets in-order, this would be interpreted as a violation of the TCP protocol rules and could trigger a false positive intrusion alert.

For most algorithms, the OoSP requirement is not imposed across *all* packets (this would make parallel processing very difficult), but only for packets within the same flow. In general, all packets that read or update the same elements in the state information of the algorithm form a flow.

5.2 An OoSP Algorithm for Designs 4 and 5

We developed a strategy to guarantee the OoSP requirement for the stateful IDS avoidance countermeasures block in designs 4 and 5. It is simple and flexible and

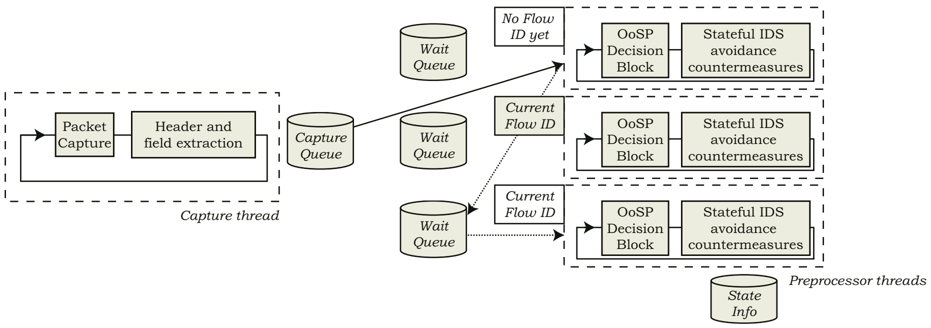


Fig. 7. The conflict resolution algorithm used in designs 4 and 5. The first preprocessor thread is consuming a packet from the capture queue. The dotted line indicates what happens in case of a conflict.

it provides fast resolution when an OoSP conflict is found. The strategy consists of three elements: each packet is assigned a flow identifier (flow ID), the flow ID is used to detect an OoSP conflict and when a conflict is found, it is resolved by storing the packet into a specific queue.

Flow Identifier. Because *packet capture* and *header and field extraction* are carried out in a single thread, packets enter the Capture Queue in order-of-arrival. Each packet is associated with a flow ID, an identifier that is identical for all packets belonging to that same flow. Because the flow ID is dependant on header fields, it is assigned in the *header and field extraction* block in in designs 4 and 5.

To reduce the likelihood of an OoSP conflict and to allow a maximal workload distribution between threads, flow identification should result in sufficient different concurrent flow IDs.

OoSP Conflict Detection. Each preprocessor thread is related to a flow ID field. A thread that is processing a packet, announces its current flow ID in this field. A preprocessor thread that is not processing a packet, will consume a packet from the Capture Queue and compare the flow ID of the new packet to the flow IDs of the *other* preprocessor threads. If a match is found, an *OoSP conflict* is detected and needs to be resolved. When no match is found, the thread announces the flow ID its own and starts processing. Because it is read and written by multiple threads, the list of active flow IDs is protected by a mutex.

OoSP Conflict Resolution. When an OoSP conflict is detected by a thread that consumed a packet from the capture queue, the new packet can not be further processed by that thread. To resolve this situation, every preprocessing thread is equipped with a FIFO queue. The thread that detected the conflict will store the packet into the wait queue of the thread that is processing packets with the same flow ID. As a consequence, all packets in a wait queue belong to the flow ID that is being processed by its corresponding thread.

A preprocessor thread will process all packets from its wait queue before consuming new packets from the capture queue.

This strategy assures that packets with the same flow ID are never simultaneously processed in different threads and that they are always processed in-order-of-arrival.

5.3 Discussion

The strategy laid out in this section is very well suited for designs 4 and 5. First, an OoSP conflict is immediately resolved by placing the conflicting packet in the appropriate wait queue and caused no blocking. Second, there is no fixed relation between flow IDs and preprocessing threads: except when they belong to a flow that is already being handled, packets are handled by the first preprocessor thread that becomes available. Finally, the impact of the strategy is limited: when no conflict is found, the overhead introduced is minimal.

6 Evaluation

6.1 Test Setup

We implemented the five multi-threaded designs through modifications of the Snort 2.0.0 release. The multi-threading primitives were implemented with the Posix threads (pthreads) API.

The granularity of protection of state-information in designs 4 and 5 is rather coarse: the stateful IDS avoidance countermeasures in Snort are implemented as modular *preprocessors*, where each preprocessor performs a specific function (IP defragmentation, TCP reassembly, ...). In our implementation, each preprocessor is associated with a single mutex that protects all its state information.

The flow ID used in the OoSP Conflict Resolution block is composed of source and destination IP address ordered in order of magnitude, thus mapping all packets between a pair of hosts onto the same flow. This flow definition suits both the IP defragmentation and TCP reassembly blocks.

The implementations were evaluated on a dual Intel Xeon machine running at 2.8GHz with a 400MHz bus, 512kB L2 cache and 1GB of RAM memory. The Xeon processors support Hyper-Threading [13], a technology where *two* threads can be simultaneously executed on *one* processor. However, these two threads share some resources in the processor's pipeline and their performance is lower compared to two threads running on separate processors. A dual Xeon machine with Hyper-Threading enabled hence supports *four* concurrent threads. Hyper-Threading can be disabled in the system's BIOS.

The system runs RedHat Linux 9 with a RedHat 2.4.20 kernel and glibc 2.3.2. This combination provides the Native Posix Threading Library (NPTL) [14], a recent and improved implementation of the pthreads API.

All experiments were carried out with the 1458 default rules of Snort's 2.0.5 release. The alerts were directed to the standard output (stdout). Traffic is taken from the Lincoln Labs DARPA Intrusion Detection Evaluation Data Set [15, 16], specifically the outside tcpdump data of weeks 1 and 4. Care was taken that the dumpfile was completely cached in RAM memory at the start of each experiment. Each experiment was repeated 20 times and the run times were averaged.

The number of parallel threads and the queue and pool sizes were adapted to the number of threads the system could simultaneously execute.

6.2 Experimental Results

Table 1 shows the sum of the averaged run times for the 10 different dumpfiles. The table also shows the run time percentage, relative to Snort 2.0.0 on a single CPU as a reference.

Table 1. Comparison of the cumulated run time and run time percentage on the 10 Lincoln Labs dumpfiles (Snort 2.0.0 on single CPU = 100%).

Run time (s)	single Xeon, Hyper-Threading disabled	dual Xeon, Hyper-Threading disabled	dual Xeon, Hyper-Threading enabled
Snort 2.0.0	132.1 (100%)	131.8 (99.8%)	131.6 (99.6%)
Design 1	163.6 (123.8%)	162.7 (123.2%)	162.4 (122.9%)
Design 2	189.8 (143.7%)	124.3 (94.1%)	113.7 (86.1%)
Design 3	180.2 (136.4%)	117.6 (89.0%)	111.0 (84.0%)
Design 4	222.4 (168.4%)	206.4 (156.2%)	199.9 (151.3%)
Design 5	223.9 (169.5%)	173.6 (131.4%)	159.0 (120.4%)

Looking at the performance of Snort 2.0.0 on the three configurations, these results confirm the findings of [3] that current NIDS sensor implementation benefit very little from multi-CPU machines because of the reasons we mentioned in section 2.

Looking at the performance of different designs on a single-CPU-machine, the original Snort 2.0.0 design is clearly the most efficient. The more complex multi-threaded designs introduce the extra overhead expected in section 4.1. This overhead can not be compensated for on a single-CPU configuration. There is also no performance gain from the isolation of the output block (design 1), as the output to stdout causes no considerable latencies.

The most interesting results are achieved by designs 2 and 3. They carry a considerable overhead when executed on a single-CPU machine (36-44%), but are able to spread their workload on the multi-CPU machines, were they actually outperform the single-threaded implementation by 16%. Individual dumpfiles show an improvement from 7% up to 20%. Design 3 is slightly faster than design 2 because the the former also executes its content normalization block in parallel threads.

The dual Xeon machine with Hyper-Threading enabled supports 4 concurrent threads, two threads on each Xeon CPU. The results indicate a considerable performance increase because of these extra threads for designs 2 and 3. We expect even more performance increase from a machine with 4 separate CPUs or from one with 4 Hyper-Threading-enabled CPUs, but unfortunately we were unable to evaluate the designs on such a platform.

Designs 4 and 5 clearly carry a large overhead because of the more complex designs, more threads and more shared data between threads. On a single CPU-machine, they add almost 70% overhead. Part of this overhead can be compensated for on the dual-CPU machines, but then the designs are still unable to outperform the original Snort 2.0.0 implementation.

We identify two reasons for this disappointing performance of designs 4 and 5. First, the protection of the state information is implemented with rather coarse granularity (see section 6.1). This limits the potential for parallel execution in the stateful avoidance countermeasures block. Second, the maximum number of concurrent threads in our dual CPU test-setups is too small to compensate the additional complexity introduced by these designs.

Design 5 outperforms design 4, because the impact of the OoSP requirement is smaller in the former: once the packets enter the matching queue, in-order processing no longer needs to be enforced.

Figure 8 shows the run times for the individual dumpfiles on the dual Xeon with Hyper-Threading enabled.

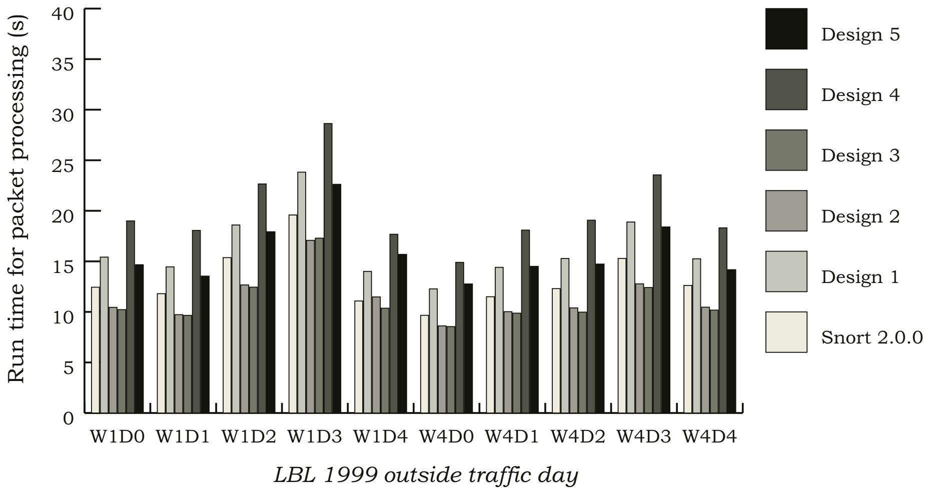


Fig. 8. Results on a Dual Xeon machine, Hyper-Threading enabled.

7 Conclusion and Future Work

In this article, we showed that a multi-threaded implementation of a signature-based NIDS sensor is able to outperform existing single-threaded implementations on small-scale multiprocessor machines. This result provides a new approach to increase the performance of individual sensors.

However, the design choices for a multi-threaded implementation have a considerable impact on the performance increase. Of the 5 designs presented here, 3 didn't show an improvement on a multi-CPU machine. The stateful NIDS avoidance countermeasures are best kept in a single thread to avoid the large overhead introduced by protecting the state data and by enforcing the OoSP requirement. Instead, multi-threading should first focus on the workload-intensive blocks that keep no state information, such as signature matching or content normalization

We were only able to evaluate our designs on a dual-processor platform, supporting at most 4 concurrent threads. It would be very interesting to evaluate multi-threaded NIDS sensors on small-scale multiprocessor machines with four or more processors. On these machines, we expect designs 2 and 3 to scale their performance by adding signature matching threads until the stateful NIDS avoidance countermeasures become the bottleneck. In this situation, designs 4 and 5 might allow performance to increase further by spreading the stateful NIDS avoidance countermeasures workload.

For future NIDS implementations, we believe that a multi-threaded approach should be seriously considered. We were able to achieve a performance increase through modification of existing single-threaded code, but we think that a completely new implementation that takes multi-threading into account from the start would suffer less overhead and perform even better.

References

1. Roesch, M., Caswell, B.: Snort, the open source network intrusion detection system. `http://www.snort.org/` (2004)
2. Roesch, M.: Snort – lightweight intrusion detection for networks. In: Proceedings of LISA '99: 13th Systems Administration Conference, Seattle, WA (1999)
3. Schaelicke, L., Slabach, T., Moore, B., Freeland, C.: Characterizing the performance of network intrusion detection sensors. In: Recent Advances in Intrusion Detection, Proceedings. Number 2820 in Lecture Notes in Computer Science, Springer-Verlag (2003) 155–172
4. Geschke, D.: Fast logging project for snort. `http://www.geschke-online.de/FLoP/` (2004)
5. Abbas, S.Y.: Introducing multi threaded solution to enhance the efficiency of snort. Master's thesis, Florida State University (2002)
6. Charitakis, I., Anagnostakis, K., Markatos, E.: An active traffic splitter architecture for intrusion detection. In: Proceedings of the 11th International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE/ACM (2003)
7. Kruegel, C., Valeur, F., Vigna, G., Kemmerer, R.: Stateful intrusion detection for high-speed networks. In: Proceedings of the IEEE Symposium on Research on Security and Privacy, IEEE (2002)
8. Top Layer Networks: Top Layer IDS load balancing system. `http://www.toplayer.com/` (2004)
9. Antonatos, S., Anagnostakis, K.G., Markatos, E.P., Polychronakis, M.: Performance analysis of content matching intrusion detection systems. In: Proceedings of the International Symposium on Applications and the Internet. (2004)

10. Jacobson, V., Leres, C., McCanne, S.: Libpcap. `http://www.tcpdump.org/` (2)
11. Ptacek, T.H., Newsham, T.N.: Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc. (1998)
12. Fisk, M., Varghese, G.: Fast content-based packet handling for intrusion detection. Technical Report CS2001-0670, UCSD (2001)
13. Intel Corporation: Hyper-threading technology. `http://www.intel.com/technology/hyperthread/` (2004)
14. Drepper, U., Molnar, I.: The native Posix thread library for Linux. Technical report, Redhat, Inc. (2003)
15. Zissman, M.: 1999 DARPA intrusion detection evaluation data set. `http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html` (2004)
16. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks: The International Journal of Computer and Telecommunications Networking **34** (2000) 579–595

An Effective Placement of Detection Systems for Distributed Attack Detection in Large Scale Networks*

Seok Bong Jeong, Young Woo Choi, and Sehun Kim

Department of Industrial Engineering, Korean Advanced Institute of Science and Technology,
373-1 Guseong-Dong, Yuseong-Gu, Daejeon, 305-701, Korea
Phone: +82 42 869 2954, Fax: +82 42 869 3110
{sbjung,ywchoi,shkim}@tmlab.kaist.ac.kr

Abstract. Recently, there has been strong interest in distributed schemes for intrusion detection in order to defend against distributed attacks such as DDoS attacks. In this paper, we focus on a placement problem of a detection system across large-scale networks for distributed intrusion detection approaches. We formulate the placement problem as a set packing problem that is NP-hard. We then present an efficient algorithm for minimizing the number of detection systems and finding the optimal placement while limiting the impact of distributed attacks.

1 Introduction

With the Internet's rapidly growing connectivity, networked computer systems are playing increasingly vital roles in modern society. While the Internet has brought great benefits to society, it has also made critical systems vulnerable to malicious attacks [1]. Distributed and coordinated attacks (e.g. the Mitnik and most distributed denial of service (DDoS) attacks [2-4]) are becoming increasingly popular among hackers; such attacks are difficult to detect and to defend against.

In recent years, there has been strong interest in distributed schemes for intrusion detection in order to defend against distributed attacks. These approaches, such as EMERALD [5] and GrIDS [6], have focused on intrusion detection across a computer network. They collect information from a variety of vantage points within computer systems and networks, and analyze this information for symptoms of security problems. The efforts include research of techniques on how to coordinate data generated by intrusion detection system (IDS) placed across a network.

In order to design an effective and deployable distributed architecture for intrusion detection, there are many challenging tasks involving a variety of algorithmic and engineering design issues [8]. For instance, what is the minimum number of detection systems (DSs) required in the given infrastructure? Generally, a larger set of DSs translates into higher cost and a longer delay in response to intrusion. Additionally, optimal placement of the DSs, which is closely related to the size of the DS set, is also an important problem; the objective here is to ensure that the set of DSs can observe most of the attack scenarios.

* This work was sponsored in part by the Korean Ministry of Information and Communication in the context of University IT Research Center Project.

In this paper, we focus on the placement problem of DSs across large scale networks for distributed intrusion detection approaches. Our objective is to place DSs across networks so as to minimize the overall number of DSs while limiting possible nodes that can participate in an attack. We formulate this problem as a set packing problem and present an efficient heuristic algorithm.

In the next section, we present the issues of DS placement and we formulate the DS placement problem and propose a heuristic algorithm. In the following section, we compare the performance between our proposed scheme and other schemes. Finally, we provide conclusions in the last section.

2 Placement of Distributed Detection System

2.1 Objectives for DDS Placement

In order to design an effective and deployable distributed architecture for intrusion detection, the objective of DSs placement is to ensure that the set of DSs can observe most attack scenarios.

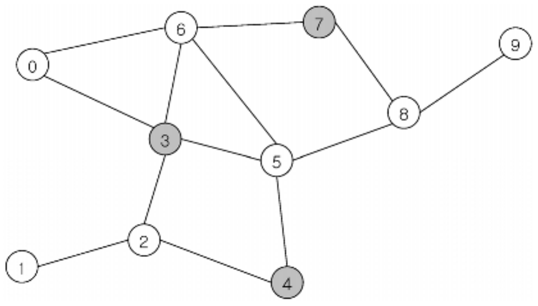


Fig. 1. Illustration of DS executed at nodes 3, 4, and 7

Consider the undirected graph shown in Fig. 1. Each node can be interpreted as a router or an autonomous system. In this paper, we assume that all attack traffic passing through sensor nodes that perform DS are detected and routing is performed by the shortest path between two nodes. In this figure, we assume that DSs are placed in nodes 3, 4, and 7. Then, all attack traffic destined to node 1 is detected except traffic from node 2. On the other hand, attack traffic from nodes 0, 6, 8, and 9 can reach node 5 because there is no DS along the path. Thus, node 1 is more robust than node 5 against distributed attacks. If additional DS is placed in node 5, no node in this network can be attacked from any other nodes that are separated by more than 2-hops. This example serves to illustrate the potential opportunities available by placing DSs in networks to detect attacks. On the practical side, it is impossible to implement DSs in all nodes in a network because this requires much greater system resources and cost. Additionally, most distributed attacks (e.g. DDoS attack) become critical threats when a great number of nodes (e.g. servers or hosts) participate in an attack. Thus, if we place DSs across the network in a well distributed manner, the impact of attacks can be sufficiently localized and minimized and can thus be ignored.

The key objectives of placing DSs can be summarized as follows: (i) minimize the total number of the DSs; (ii) minimize the number of nodes that could send the attack packets to any other nodes that are separated by more than the given number of hops without passing through sensors; (iii) find the optimal placement of the DSs that satisfy (i) and (ii).

In the following section we formulate the DS placement problem (DSPP) and drive algorithm to solve this problem more efficiently.

2.2 DS Placement Problem

Let G = (*V, E*) be an undirected graph representing Internet topology. Each node in *V* can be interpreted as a router or an autonomous system. We use T to denote a subset $T \subseteq V$ of nodes where intrusion detection is performed. We call $\tau = |T| / |V|$ the coverage ratio. Let r be the localization factor, which means the tolerance level against attacks. That is, each node can be permitted to be attacked by any other nodes that are separated by less than r hops. This is because nodes that can be candidates for attack sources are localized by r and the attack impact of these nodes can be small enough to be ignored when r is small.

To quantify the detecting effect, we define the risk level of node i against attacks, $c_i(r)$, as the number of nodes that are more than r hops apart from node i and can send attack packets to node i without passing through DSs. If $c_i(r) = 0$, then every attack can be localized to within a small set of candidate nodes with a distance of less than r hops from node i. Additionally, if $c_i(1) = 0$, then all attack packets destined to node i are detected because all traffic destined to node i must pass through at least one DS. However, a smaller value of r in V means that T has to be larger. On other hand, a larger value of r means that a node is at greater risk, because the node can receive more attack traffic without being detected by DSs executed in T. Generally, DS should be located at strategic locations in the Internet because implementing DS in a specific node requires much more system resources and cost. On other hand, the number of DSs has to be sufficient and DSs have to be well distributed among nodes in order to detect a distributed attack effectively. Therefore, our objective is to minimize the total number of DSs while satisfying $c_i(r)$ =0 for a given *r*.

Mathematically, we can formulate the DSPP as follows:

(DSPP)

$$\begin{aligned} &\min \ |T| \\ &\text{s.t.} \ \ c_i(r) = 0, \quad \text{for } \forall i \in V \end{aligned}$$

Let x_i be the decision variable, which is 1 if node i performs DS and 0 otherwise. Let T_e be the subset of E, which is composed of the edges that connect the nodes that perform DS. We define the distance between node i and j in G, $d(i,j)$ as the minimum number of hops between the nodes. For a given T, consider $G'(V',E')$ where $V' = V \setminus T$, $E' = E \setminus T_e$. Let c_{ij} be 1 if the distance between node i and j is more than r in G', and 0 elsewhere. Then, the above formulation is re-presented as follows:

(DSPP2)

$$\begin{aligned} \min \ & \sum_i x_i \\ \text{s.t.} \ & \sum_j c_{ij} = 0, \quad \text{for } \forall i \in V' \\ & x_i \in \{0,1\}, \quad \text{for } \forall i \in V \end{aligned}$$

However, in this problem the evaluation of c_{ij} will be an exhaustive work, because the graph should be modified according to the value of x_i. Moreover, the modification becomes an obstacle to grasp an intuitively optimal solution. Thus, we transform the above formulation to an equivalent formulation.

Let $V = \{1,\ldots,m\}$ be a finite node set and let $\{V_j\}$ for $j \in N = \{1,\cdots,n\}$ be a given collection of subsets of V. Let's introduce a set packing problem. We say that $F \subseteq N$ is a packing with respect to V if $V_j \cap V_k = \varnothing$ for all $j, k \in F, j \neq k$. Each packing is composed of nodes that are not DS nodes. Maximizing the number of these nodes is equivalent to minimizing the number of DS nodes for a given graph. The maximum value of $d(i,j)$ for all nodes $i, j \in V_k$ in a packing should be less than r. The nodes that are not included in a packing are the DS nodes. Let x_j be the decision variable, which is 1 if the index j of V_j is included in the set packing *F*, and 0 otherwise. Let a_{ij} be the coefficient, which is 1 if the node i is included in V_j, and 0 otherwise. Then, the previous DSPP2 is equivalent to the following combinatorial optimization problem:

(DSPP3)

$$\begin{aligned} \max \ & \sum_{i \in V} \sum_{j \in N} a_{ij} x_j \\ \text{s.t.} \ & \sum_{j \in N} a_{ij} x_j \leq 1 \quad \text{for } \forall i \in V && (1) \\ & d(i,k) x_j \leq r - 1 \quad \text{for } \forall i,\ k \in V_j,\ i \neq k,\ \forall j \in N && (2) \\ & 2 x_j x_l \leq d(i,k) \quad \text{for } \forall i \in V_j,\ k \in V_l,\ j \neq l,\ \forall j,l \in N && (3) \\ & x \in B^n && (4) \end{aligned}$$

Constraint (1) is the general set packing constraint. Constraint (2) means that the maximum value of $d(i,k)$ for all nodes i, k in a feasible packing should be less than r. Constraint (3) means that the distance between a node of V_j and a node of V_l should always be more than 2-hops. Since we pack the nodes that are not DS nodes, DS nodes should be placed around a packing. That is, if V_j and V_l are feasible packs, then there should be DS nodes between a node of V_j and a node of V_l.

This problem is basically a set packing problem, which is NP-hard [7]; hence, we have to rely on heuristic solution methods. However, in this DSPP3 formulation, intuitively, the set of DS nodes becomes the cut across the packs. We propose a heuristic algorithm based on this intuition. Our heuristic algorithm packs some connected nodes while the maximum value of distances in the pack is less than r. Then, all nodes connected directly to the pack become DS nodes. This process repeats until all nodes are packing or become DS nodes.

```
<Heuristic Algorithm>
Step 1: Set $i=1$ and $G^1\left(V^1,E^1\right)=G(V,E)$. $DS=\varnothing$.
Step 2: Search a node $j^*$ that has the minimum number of edges in $G^i\left(V^i,E^i\right)$.
    $V_i=\left\{j^*\right\}$. $E_i=\varnothing$. $DS_i=\varnothing$. $d_i=0$.
    Step 2.1: Search a node $j^*$ that has the minimum number of edges in the nodes
        such that $j^*\in V^i$, $j^*\notin V_i$, $\exists k\in V_i \; s.t. \; e\left(j^*,k\right)\in E$.
        Step 2.1.1: If there is not such a node then proceed to 3.
    Step 2.2: $E_i=E_i\cup\left\{e\left(j^*,k\right)\right\} \; \forall k \; s.t. \; e\left(j^*,k\right)\in E, k\in V_i$.
    Step 2.3: $V_i=V_i\cup\left\{j^*\right\}$.
    Step 2.4: Calculate the maximum value of distances $d_i$ in $G\left(V_i,E_i\right)$.
    Step 2.5: if $d_i<r-1$ then proceed to 2.1, else proceed to 3
Step 3: $DS_i=DS_i\cup\{k\} \; \forall k\in V^i, k\notin V_i, e(k,l)\in E^i s.t. \; l\in V_i$.
    Step 3.1: $E_i'=E_i\cup\{e(j,k)\} \; \forall j,k \; s.t. j\in DS_i, k\in V_i, e(j,k)\in E^i$
    Step 3.2: $V_i'=V_i\cup DS_i$.
    Step 3.3: $E_i'=E_i'\cup\{e(j,k)\} \; \forall j,k \; s.t. j\notin V_i', j\in V^i, k\in DS_i, e(j,k)\in E^i$
    Step 3.4: $V^{i+1}=V^i\setminus V_i'$, $E^{i+1}=E^i\setminus E_i'$.
    Step 3.5: If $V^{i+1}$ is empty then proceed to 4, else $i=i+1$ and proceed to 2.
Step 4: $DS=\bigcup DS_i$. Terminate.
```

3 Numerical Results

The performance of the proposed scheme is evaluated by simulations over a 12-node ring network, a 14-node NSFNET network, and 48-node CTNET network topology (Fig. 2(a), (b), and (c), respectively). For the purpose of simulation, we have also considered a connection-based placement scheme (CPS) that samples nodes as DSs from V based on the number of connections of each node. The CPS chooses the node that has the most connections as a DS repeatedly until the target r is reached. In the case of a tie, it chooses a node randomly.

In our simulations, we consider routing policies that allow multi-path routing. When routing is performed between two nodes, we select all possible loop-free shortest paths between them.

Fig. 3 shows the average coverage ratio for each network that satisfies $c_i(r)=0$ for all nodes as r increases from 1 to 8. From these figures, we observe that the average performance of CPS is very limited. This reinforces that DS placement is an important issue. As shown in Fig. 3, the proposed scheme always has a lower coverage ratio. This means that our scheme requires fewer DSs than CPS for satisfying the same r. As a result, the proposed scheme performs better than CPS.

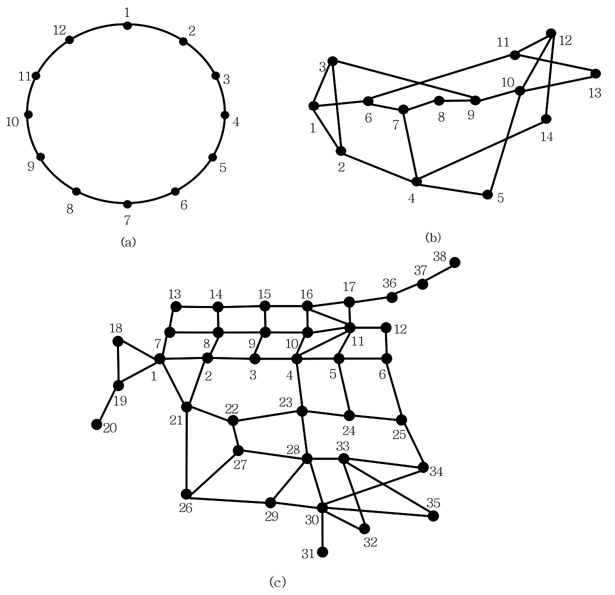


Fig. 2. (a) 12-node ring network. (b) 14-node NSFNET network. (c) 38-node CTNET network

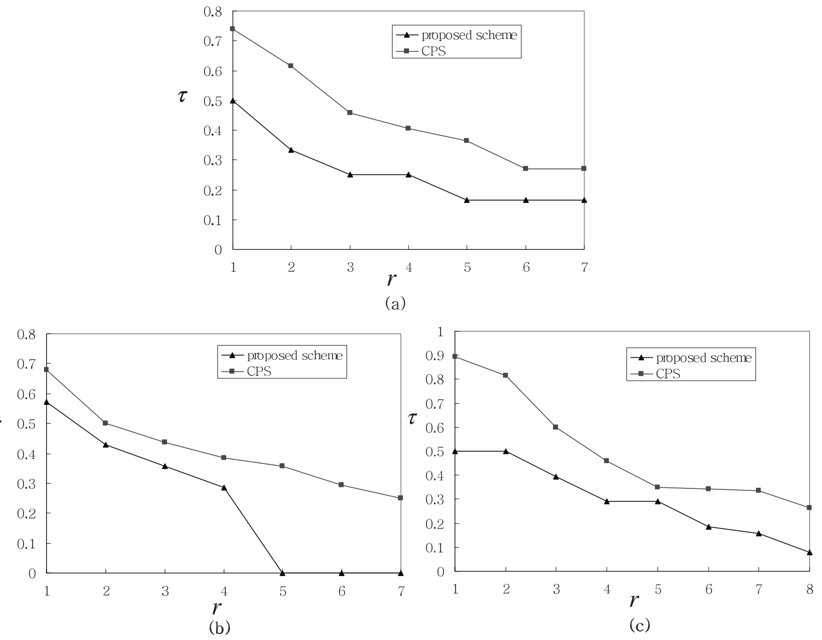


Fig. 3. Coverage ratio of (a) 12-node ring network, (b) 14-node NSFNET network, and (c) 38-node CTNET network

We can observe that with an increase of r, the average coverage ratio decreases. However, a larger value of r means that the network is more vulnerable against distributed attacks. On the other hand, when $r = 1$, the coverage ratios of CPS and the proposed scheme are above 65% and 50%, respectively. These values are so high as to be negligible due to the high cost of larger coverage. Therefore, perfect detection of attacks (in the case of $r = 1$) is intrinsically difficult to attain, and should not be construed as a viable performance goal. Thus, we have to select r carefully based on the current system resources and the impact of attacks.

4 Conclusions

We have presented a DSs placement approach in order to detect distributed attacks. We have shown that perfect detection is difficult to achieve in the Internet environment while maintaining sparse coverage. However, this is mitigated by the fact that attack traffic that can escape the DS can be localized within r hops.

The performance results show that our scheme reduces the total number of DSs while localizing attack candidate sources. Our scheme can be carried over to AS or router topologies within an AS.

References

1. Ghosh, A.K., Wanken, J., Charron, F.: Detecting anomalous and unknown intrusions against programs, Proceeding of the 14th Annual computer security applications conference, December (1998) 259-267
2. Northcutt, S.: Network Intrusion Detection: An Analyst's Handbook, New Rider (1999)
3. Lau, F., Rubin, S.H., Smith, M.H., Trajovic, L.: Distributed denial of service attacks, In IEEE International conference on systems, Man, and Cybernetics, Nashville, TN, USA, October (2000) 2275-2280
4. Mirkovic, J., Martin, J., Reiher, P.: A taxonomy of DDoS attacks and DDoS defense mechanisms, Technical report, Computer Science Department, University of California, Los Angeles (2002)
5. Porras, P.A., Neumann, P.G.: EMERALD: event monitoring enabling response to anomalous live disturbances, Proceedings of the 20th National Information Systems Security Conference, National Institute of Standards and Technology (1997)
6. Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Rowe, J., Staniford-Chen, S., Yip, R., Zerkle, D.: The design of grids: A graph-based intrusion detection system. Technical report CSE-99-2, U.C.Davis Computer Science Department, January (1999)
7. Garey, M.R., Johson, D.S.: Computers and Interactability: a Guide to the Theory of NP-completeness, W.H. Freeman and Company, San Francisco (1979)
8. Wan, K.K., Roky, K.C. Chang: Engineering of a global defense infrastructure for DDOS attacks, Proceedings of IEEE International Conference on Networks, August (2002)

Application of Content Computing in Honeyfarm

Yi-Yuan Huang[1], Kwok-Yan Lam[1], Siu-Leung Chung[2], Chi-Hung Chi[3], and Jia-Guang Sun[1]

[1] School of Software, Tsinghua University, Beijing, PR China
{huangyy02,lamky,sunjg}@tsinghua.edu.cn
[2] School of Business Administration, The Open University of Hong Kong
slchung@ouhk.edu.hk
[3] School of Computing, National University of Singapore
chich@comp.nus.edu.sg

Abstract. Currently there are a wide range of techniques dedicated to network security among which honeyfarm is one of the newest development. Honeyfarm provides replication of web sites aiming to redirect hackers from the corresponding production sites. To achieve the goal, transparency and quick access are the basic requirements for honeyfarm. Content delivery network (CDN) is one method for providing replicated web sites and the technique is able to address the requirement for rapid content delivery by redirecting clients to the replicas. However, there is no transparency in the redirection. Furthermore, CDN also redirect clients in a centralized way which will induce latency in content delivery. This paper describes the ways to improve CDN to make it suitable to use as a routing mechanism for honeyfarm.

Keywords: Network security; distributed computing; content delivery network; honeyfarm.

1 Introduction

CDN is the acronym of "content delivery network". It is a dedicated network of servers, deployed throughout the Internet, which Web publishers can use to distribute their content on a subscription basis. In order to enhance the response of network, some mirror web sites will be constructed to provide the same contents and responses as the origin servers. When a client types the domain name of his target or the IP address in the browser, the traffic will be transmitted to the local DNS. Then the local DNS server sends this request to the RRI (request-routing infrastructure). After receiving the request from the local CDN server, the RRI chooses the most appropriate CDN node and sends the IP address of the chosen CDN node to the users. This way, all the users' requests will be redirected to this IP address and the CDN node will return responses accordingly. CDN is a well-developed technique for users to enhance the speed of reaching their target web sites or getting their needful contents. There are four main components in CDN namely surrogate servers, routers, redirection server and accounting logs [1]. The functions of these four components will be outlined in section 2.

Honeypot is a fast developing security technology nowadays. However, there is a serious problem with honeypot in that it is difficult to create good honeypot to lure hackers, especially sophisticated hackers whose possess techniques to penetrate into systems with highly confidential contents. In this respect, honeyfarm, a type of new and advanced honeypot, is focused on. Honeyfarm is made up of a large number of honeypots, which provides services as hackers need. Traditional honeypots provide nothing or only some fake information so that they cannot attract hackers to show their behaviors efficiently. On the other hand, honeyfarm will simulate everything as the hackers want to attract them to stay on that fake environment. The closer to the real environment the honeyfarm is able to imitate, the more possible the honeyfarm can lure the hackers to stay on with them, and as a result, the more information about hackers' behaviors researcher are able to obtain. Since people who have joined in the CDN services would be redirected to the topologically nearest web site in terms of the topological proximity, the routing mechanism of CDN can also be used for the honeyfarm to redirect hackers. Similarly in a honeyfarm, when a hacker is detected, some lookup requests will be sent to the honeyfarm to select the most appropriate honeypot to deliver the emulated responses to the hacker. If the honeypot can provide what the hacker is interested in, then the hacker will be redirected to the selected honeypot. Otherwise, the hacker will be given some limited privilege such as limited bandwidth to access to their real target. In the meantime, honeyfarm will download all the content of the target (i.e. the whole web site) to its fake environment much faster than hackers can obtain so that it can simulate the hackers' target before the hacker reaches his goals. After the honeyfarm completes its simulation, the hacker may just get a part of the contents that he is interested in. Then the honeyfarm will redirect the hacker to what the honeyfarm has imitated. Afterward, the hacker will slump into the fake service and will take actions without knowing that his behaviour is being observed and recorded. However, there are transparency and centralized problems in CDN redirection mechanism, CDN should be improved if it is to be used as a redirect mechanism for honeyfarm. Moreover, because CDN can be utilized to reduce the latency [2] when redirecting hackers to the appropriate honeypot, the honeyfarm can answer their requests more successfully and quickly than the traditional honeyfarm redirection mechanism. The rest of the paper introduces the ways to modify CDN and use it to redirect hackers to the honeyfarm. The paper is organized as follow. Section 2 focuses on the concept of CDN and its redirection mechanisms. Section 3 introduces the concepts of honeypot and honeyfarm. Section 4 outlines the use of CDN to build the honeyfarm and the realization of the redirection of the honeyfarm by adding CDN theory into the conventional honeyfarm redirection mechanism. The paper is concluded in Section 5.

2 An Overview of Content Delivery Networks

Content Delivery Network is a kind of dedicated network optimized to deliver specific contents such as static web pages, transaction-based web sites, streaming media, or even real-time video or audio [3]. Generally speaking, CDN mainly consists of four basic components [1]:

– Surrogate servers, which is used to store the contents of the original servers like mirror web sites. Clients can get the contents from the topologically nearest surrogate servers according to the allocation policies of CDN.
– Routers, which act as the redirectors to allocate clients to the topologically nearest surrogate servers according to some predefined policies.
– Request-routing infrastructure (RRI), which are diversity of components responsible for distributing the clients' requests from the original servers to the surrogate servers.
– Accounting logs, which have the functions of delivering system logs and accounting information to the original servers.

The CDN surrogate servers and the RRI are deployed throughout the Internet which Web publishers can make use of to distribute their contents. CDN first emerged in 1998 whence the web servers cannot meet the requirements of transmissions over long distances. Before CDN came forth, ISPs used another kind of technology, the proxy, to cache the contents so that they can serve their clients more efficiently. However, there are two drawbacks in proxy technology. First, if the proxy is not appropriately and immediately updated, the clients will receive just outdated data. Second, bottleneck becomes another serious problem due to the exploding growth of the number of Internet clients. CDN is able to solve these two problems by distributing clients' requests to an appropriate CDN node in terms of the topological proximity between the client and each CDN node that can serve clients as their origin destination can serve. Moreover, CDN essentially focuses on more advanced issues than proxies. While ISPs use proxies to store the most frequently used and most recently requested contents, CDN not only caches these contents, but can also store secure contents, stream contents and dynamic contents. CDN makes use of two primary technologies, which are intelligent wide area traffic management and cache. The function of intelligent wide area traffic management is to direct clients' requests to the optimal site according to the topological proximity. Two types of technologies, DNS (domain name service) redirection and URL rewriting, are currently used to direct clients' requests to the appropriate server. In DNS redirection, the DNS server performs the mapping between a surrogate server's symbolic name and its numerical IP address. The following steps describe the ways to locate an appropriate surrogate server for the clients:

– The user sends a DNS query to the local DNS server, then the local DNS server routes the query to the CDN's request-routing infrastructure, which is called RRI.
– The RRI asks each surrogate server to examine and measure its particular route to the local DNS server.
– Each surrogate server sends measurement results to both the local DNS server and the RRI. Additionally, it also sends other criteria to the RRI, which lets the infrastructure compare each server's topological proximity

with the local DNS server. Server measurements are based on network metrics such as latency, packet loss, and router hops from surrogate servers to requesting entities and feedback from a pool of surrogate servers, from which the proximity-load-threshold algorithm selects the least-loaded surrogate server [1].
- The RRI compares all the previous measurements and chooses the most appropriate surrogate server to deliver the requested content. In the mean time, RRI sends a DNS response to the user's local DNS server.
- The user's local DNS server sends that response to the user.

Generally the DNS-based CDN service can be categorized into two types, which are full site CDN and partial site CDN. For full-site content delivery, the RRI redirects all the requests from the clients through the DNS to a CDN server. On the contrary, partial-site CDN delivers only embedded objects such as web page images from the corresponding CDN. With partial-site content CDNs, the surrogate server fetches only non-cacheable objects from the origin web server, which can reduce the load on the site's content generation infrastructure and the data that the surrogate server should retrieve from the origin server. Because of the frequently changing embedded objects, partial-site deployment typically achieves better performance [1]. While most CDNs use DNS-based mechanism to redirect their clients to the appropriate server, some also used 'rewriting URL' to fulfill the redirection. To achieve this goal, the origin server will rewrite the dynamically generated pages' URL links. For instance, with a web page containing an HTML file and embedded objects, the web server would amend references to embedded objects so that the client could get them from the most appropriate surrogate server by calculating and comparing the topological proximity between each CDN node and the clients. CDNs offer special scripts that transparently parse Web page content and cover for embedded URLs. Additionally, URL rewriting is also called content modification because CDNs use it in just one of the following two ways [1]:

- Statically. The Web server modifies content and rewrites embedded URLs before the content is stored on the origin server and made available to clients.
- Dynamically, the content is amended according to clients' demand in real time. For dynamic Web pages, the basic drawback of URL rewriting is that it imposes an important performance overhead as scripts are ought to be continually executed.

Besides intelligent wide area traffic management, cache is another pivotal issue in CDN. Cache is a generic way of providing web server function without clients' having to know application-specific details. There are millions of cache nodes deployed in the entire Internet by CDN, all of which save the useful contents that can be provided to the local clients to decrease the burden of the origin servers. Two simple and popular policies will be utilized by cache to determine which content need to be stored and which need to be expired. The first one is the least frequently used standard. In this standard, objects or contents, which are least frequently used, should be eliminated in order to provide space for new objects. The second one is the least recently used standard which evicts objects that have not been accessed for the longest time. When a cache receives a request for a web object, it first checks whether the copy of this web object is in the cache. If so, the cache sends a header request to the web server to see if the content has changed. If the object is fresh, the cache fulfills the request by sending the client a copy of the object. If not, the cache makes a request to the origin server to fetch the content that its clients need, then it fulfills the client's request and returns the appropriate response.

3 Overview of Honeypot and Honeyfarm

In this section honeypot will be first introduced followed by the concept of honeyfarm. Honeypot is a security resource whose value lies in being probed, attacked or compromised [4]. The goal of honeypot is to research and explore hackers' behaviors such as hackers' actions, motives, tools, targets and tactics. To achieve this goal, honeypot will be deployed around the Internet, beside the organization or in the intranet. For instance, when a hacker is lured into a honeypot, it will provide hackers the needed information as much as it can so that hackers would show their behaviors on the honeypots. But it has to ensure that the hackers don't know they are being inspected by researchers. After the hackers have done something on the honeypots without knowing the detection, researchers can observe and record the activities and behaviors of the hackers and through the analysis of these information, new prevention tools such as new rules in the rule-based IDS can be devised. Based on the level of interaction that honeypot can provide to the hackers, honeypots can be categorized into three different types, whose functions are increased according to the level of interaction between the honeypots and the hackers:

- Low-interaction honeypot. As the name implies, low-interaction honeypot is easy to install and simply simulates a few services. The attacker is limited to interact with these predefined services (such as fake ftp or http service). The primary value of low-interaction honeypot is the detection, especially detection of those unauthorized connection attempts and unauthorized scans. The disadvantage of low-interaction honeypots is that they cannot provide detailed information about hackers in which researchers may have interests. But its simplicity is also its advantage. Since low-interaction honeypots are simple, they have the lowest level of risk. For there is little function provided in the low-interaction honeypots, there is less to go wrong. Additionally, because low-interaction honeypots do not offer real operating system for the hackers to interact with, this kind of honeypot cannot be used to harm, attack or monitor other computers and systems in the Internet.
- High-interaction honeypot. On the contrary, high-interaction honeypots provide real operating system and emulate the service as real as hackers need so that they can lure unauthorized people to connect to them, especially sophisticated hackers. High-interaction honeypots are actual systems with full-blown operating systems and applications. They are primarily used for research purposes. Since a real operating system is provided, the level of risk

is much higher than low-interaction honeypot. But with high-interaction honeypots, researchers can discover new tools, identify new vulnerabilities in operating systems or applications, and learn how hackers communicate among one another.
– Medium-interaction honeypot. Medium-interaction honeypots are easier to deploy and install than high-interaction honeypots but more complex than low-interaction honeypots. They mainly provide the following four functions to clients [4]:
  1. Actually capturing worm payloads or attacker activity.
  2. To learn what happen after attackers gain access to a system.
  3. To capture their toolkits.

Although medium-interaction is able to get a greater amount of information than low-interaction, it inevitably brings a higher level of risk and more complexity to its clients.

Honeyfarm is one type of high-interaction honeypot. It consists of a cluster of honeypots which are deployed in the same location or all over the Internet and used to simulate what hackers want as real as possible so that they can attract hackers, especially sophisticated hackers, to intrude into them and show their behaviors without knowledge about the inspection. For example, when an unauthorized person is detected to connect to some resource, the resource will send the request to the honeyfarm to ensure whether those needed contents have been simulated in the honeyfarm. If so, the person will be redirected to the honeyfarm. If not, the honeyfarm will first emulate what the person is interested in and then redirect the person to it. Currently there are two types of honeyfarms:

– All the honeypots in the honeyfarm are deployed all over the Internet, beside those organizations that have joined the honeyfarm mechanism just like CDN nodes and permit researchers to use their resource to lure hackers.
– All the honeypots in the honeyfarm are deployed in one or some locations so that administrators can spare more resources to manage them. In this type of honeyfarm, organizations won't deploy honeypots on their networks. Instead, they will simply deploy a hardware device that monitors unused IP addresses, similar to some kind of high-interaction honeypot tool called honeyd [4], and redirects all attacker traffics to a single cluster of honeypots by some routing mechanism.

Traditionally, both types of honeyfarm use layer 2 VPN to redirect the hackers. In this paper, to utilize the advantages of CDN, the honeyfarm is constructed in the second way outlined above, deploying all the honeypots all over the Internet in the way in which normal CDN nodes are constructed. In the honeyfarm mechanism, one of the most important technologies is redirection since all the unauthorized traffic ought to be redirected to the honeyfarm without hackers' detection. To achieve this goal, researchers have find a way to distinguish authorized and unauthorized access. After identifying the hackers, they can use the appropriate redirection mechanism to route them to the honeyfarm and record their behaviors. There are three primary requirements in the redirection of honeyfarm:

- Transparency. Transparency is the most important requirements for redirection since if hackers discover that they have been inspected, they will abort their connection to the honeypots. Then researchers are not able to record the hackers' behaviors and the honeyfarm will lose its value as a security resource. Furthermore, because the clients' system can communicate with the honeyfarm without any restriction, if hackers detect the inspection, it will change its policies and utilize the honeyfarm to harm the clients' systems.
- Upgrade. Besides transparency, upgrade is also an important requirement for the honeyfarm because the clients who have joined in the honeyfarm services always update their contents in their web sites. To emulate the honeyfarm as truly as possible, researchers should also follow the clients who have join in the honeyfarm service and update the contents in the honeyfarm in time.
- Latency. Speed of connection to the honeyfarm is important. With the fast growth in the number of hackers, the conventional honeyfarms impose large latency between the honeyfarm and the hackers, which may not only make hackers lose the patience to wait for the responses and quit.

With all these requirements, it can been seen that CDN is suitable to be used as the redirecting mechanism for honeyfarm.

4 A New Redirection Mechanism in Honeyfarm

To ensure the success of honeyfarm, the most important aspect is to redirect hackers to the honeyfarm transparently. Currently there are three kinds of redirection mechanisms, namely DNS-based redirection, TCP handoff and HTTP-based redirection. The DNS-based redirection and the HTTP-based redirection cannot route the suspected traffic to the honeyfarm transparently, and the TCP handoff cannot meet the upgrade requirement of the honeyfarm. So in traditional honeyfarm, layer 2 VPN, is used to implement the transparent redirection. However, since layer 2 VPN is one type of access-centralized redirection mechanism. When creating or removing a VPN, the client needs to request a VPN manager to permit this kind of request [5]. So it will create latency with the growth of the number of hackers and their traffic, which is called centralized problem. Ideally a client's request should not be directed to a centralized manager to get the response. In this paper, the redirection mechanism of CDN (as described in section 2) is modified and combined with DNS-based redirection mechanism to route the unauthorized traffic transparently and efficiently. It is well known that the primary goal of CDN is to redirect clients to the particular servers to reduce the Internet traffic. But the routing mechanism of CDN may not satisfy the transparency requirement of the redirection in honeyfarm. Furthermore, when redirecting suspected traffic, RRI needs to compare all the topological proximity, which still give rise to the centralized problem. In this paper, in order to satisfy the transparency requirement, we propose to integrate RRI, local DNS server and the proxy cache into one simple component, which is called a redirection server. To eliminate the centralized problem, all the honeypots are organized using the CDN architecture and the redirection servers are organized in a tree structure, in which each redirection server is distributed to compare the topological proximity between each honeypot in the honeyfarm and the redirection server. This way, the information about honeypots can be easily updated. Honeypots can send their own information about the load condition and the content they possess to the leaf redirection server that monitors the domain in which they are located.

In the following, we will outline the way to identify potential hackers the way to use the routing mechanism in CDN to meet the requirements of the redirection in honeyfarm. To provide this kind of routing mechanism, the structure of the collection of redirection servers is described briefly, then the ways the redirection server work and the process of redirecting hackers in the honeyfarm will be provided. In order to facilitate the operations of the honeyfarm more smoothly and efficiently, researchers should identify unauthorized access as soon as possible. In the honeyfarm mechanism, three methods are combined to identify the suspects:

- Monitor those unused IP address in the intranet. Because these IP addresses are not used, anyone who tries to access to them or scan them are labelled as suspects.
- After the suspects are being identified, rule-based IDS is used to identify hackers. Since rule-based IDS technology is well developed, researchers can easily find suspects and redirect them to the honeyfarm.
- Lastly, firewall is used to find suspected connection.

All these systems are called mid-system because they are not the hackers' final targets. Instead, the hackers just utilize them to harm their ultimate targets and a huge number of sophisticated hackers are using this kind of method to access their target. In this paper, only this kind of hackers is considered. After identifying the hackers, the CDN should be constructed in that a collection of redirection servers are constructed as a tree instead of RRI and local DNS servers in CDN and deployed all over the Internet. This kind of deployment is considered as a CDN network and the CDN redirection mechanism is modified to route unauthorized traffic. The whole redirection service in the honeyfarm relies on two main components, a hierarchical collection of redirection servers and the mechanism of the redirection server itself.

Before we introduce how the redirection mechanism of CDN is modified to implement the redirection in the honeyfarm, the hierarchical collection of redirection servers should be introduced. In order to handle the routing request in a distributive manner, the collection of redirection servers can be organized as follow. The whole Internet is partitioned into several domains, each of which is monitored by one and only one redirection server. The distributed redirection servers are organized in a tree structure. The node at the lowest level of tree is the leaf node. Each leaf node is a redirection server, which monitors all the honeypots in one domain and stores the information about the honeypots in this domain. According to the way in which the Internet is divided, each leaf redirection server in the same domains is aggregated into one server, which is called the father node. Recursively all the leaf nodes will be aggregated into one single server, which monitors the entire Internet. This server is called the root node for the tree [2]. Each node in this tree is a redirection server that helps people redirect risky traffic to the honeyfarm. For example, the whole Internet is divided into six parts corresponding to the six continents, which are Asia, Europe, North America, South America, Oceania and Africa. Each part is further subdivided into smaller parts. The more parts the world each part is divided into, the more resources have to be consumed to construct the collection of redirection servers, but the less comparing work that each redirection server have to do and the more efficient the redirection. Figure 1 shows the design of the collection of the redirection servers. The root server monitors the entire Internet, the six servers in the second layer monitor the six parts. Additionally, in Figure 1 not only is the world divided into six parts, but also are these six parts divided into some more detailed parts, each of which is also monitored by one leaf redirection server to distribute the handling of the requests, either from the mid-systems or from other redirection servers.

To simplify the architecture of the collection of the redirection servers, five functions should be integrated into the leaf redirection server.

1. Answer incoming requests and sending requests to other servers in the tree.
2. Manage the local information for the honeypots in its own monitoring domain.
3. Execute the functions of the local DNS server for the web sites in its monitoring domain.
4. Compare the topological proximity between each honeypot in their inspecting domains and the hacker.
5. Fetch the contents that the clients need and act as if they are from the primary destination.

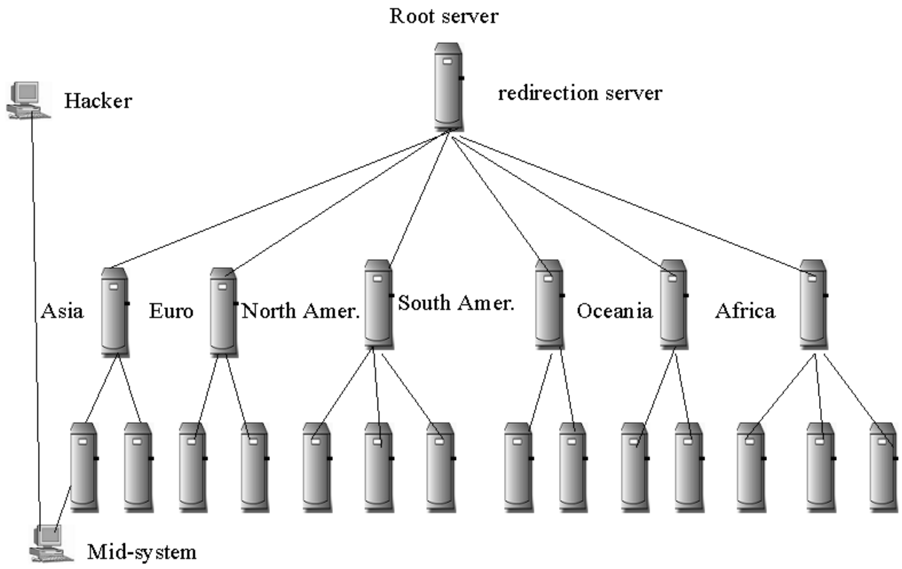


Fig. 1. The design of a collection of redirection servers.

While the leaf redirection server should perform the five functions, other redirection servers just need to perform function 1 and function 4. When a client sends a request to the local redirection server to get contents, the following three steps will be processed to select the optimal honeypot:

– The local redirection server computes and compares the topological proximity from each honeypot, which it monitored, to itself, chooses the optimal honeypots and sends its IP address to its father node.
– In the meantime, the local redirection server sends a request through its father node to the whole tree of collection of the redirection servers, asking all the leaf nodes to select the optimal honeypots, relative to the client, in their domain, and return the results to their father node.
– The father nodes compares all the results, chooses the optimal honeypot's IP address and sends them to their father nodes. Recursively, the root redirection server will find out the most appropriate honeypot for the local redirection server and send the measurement result to it.

For the design of the redirection servers, the DNS-based redirection can be used to route the suspects to the honeyfarm transparently. It is the duty of the client-side redirection server to maintain the redirection transparency, which is achieved by not displaying the address of the honeypot to the client browser. Instead, by modifying the address of the honeypot provided to the client browser, the redirection server can make the clients believe that they get the contents in which they show interested from their origin target. There is no way to find out the address of the honeypot when the redirection service is used.

Additionally, before describing the redirection process, the following steps outline the general processes of the DNS-based mechanism. When an ordinary client requiring some contents from a web site, there are two steps in the communications between the client and the web site:

– The client's system sends a request to the local DNS server to resolve the domain name into the IP address.
– The resolved IP address is returned to the client by the local DNS server. Then the client communicates with the web site by launching requests (i.e. HTTP requests) to the web site server and gets back the responses.

In the honeyfarm, to redirect the clients, the local DNS server can modify the IP addresses that the local DNS server returns to the client to fulfill the redirection. Compared with the local DNS servers, redirection servers mentioned above also can fulfill their redirection by modifying the resolving IP addresses and return it instead of the real ones.

Considering those hackers who are using some mid-systems, if one was detected to try to penetrate into a protected system illegally by a mid-system, the whole process of the redirection is progressed in the following five steps when the mid-system receives requests from the hacker:

– The mid-system sends a request to its local redirection server to resolve the hackers' destination (usually a domain name) into an IP address, which is the guidance for the mid-system to access to the hacker's target. Since

the local redirection server will send back its own IP address to the client browser instead of the real IP address of the hacker's final target, the hacker's following HTTP requests will be redirected to the local redirection server.
– The redirection server selects the optimal honeypot for the hackers by analyzing the hacker's HTTP requests, comparing all the topological proximities between each honeypot and the hacker.
– If the optimal server exists, the redirection server sends the hacker's requests to this selected honeypot and the honeypot then returns the appropriate responses to the local redirection server in terms of the hacker's requests. Upon receiving the responses, the local redirection server falsifies the responses as if these responses were sent back to the hacker by the hackers' origin destination.
– If there is no honeypot that has simulated the hackers' destination, the redirection server will keep the hackers to stay on them, give them some privilege such as a limited bandwidth to connect to the Internet. In the meantime, the honeyfarm emulates the needed environment. When the emulation is completed, the hacker will be redirected to the honeyfarm by sending the IP address of the emulated environment to the redirection server, then the same communication process between the local redirection server and the chosen honeypot will be implemented as the third step.
– After the local redirection server gets those required responses (i.e. the contents that hackers are interested in or some answers for the hackers' requests), it will send them to the hackers.

Because the redirection server is distributed throughout the Internet and organized as a tree, the task of comparing and computing the topological proximity between hackers and each honeypot in the honeyfam can be distributed to many redirection servers, which enhances the speed of hackers' accessing to the honeypot. This then can remove, or at least decrease to a certain extent, the centralized problem in the traditional honeyfarm so that the latency in the connection between the hackers and the honeyfarm is reduced. Ultimately, each honeypot in the honeyfarm is directly managed by one and only one leaf redirection server. When the information about a honeypot is updated, for instance, its load condition is changed or its emulated contents are refreshed, it can immediately ask its manager (the redirection server) to update the information. This way the requirement for update can be satisfied.

5 Conclusion

Honeyfarm is a newly developed concept. It is some type of high-interaction honeypot, which not only protects the web sites from being intruded in illegally, but also helps researchers to study hackers' behaviors efficiently, especially new and advanced behaviors. In order to direct hackers away from those protected web sites and redirect them into the honeyfarm, redirection mechanism is considered as one of the most important aspects that should be dealt with by researchers. In this paper, a distributed redirection mechanism is added into the traditional honeyfarm to redirect the risky traffic to the honeyfarm transparently. Additionally the distributed redirection mechanism can reduce the load consumption of the honeypots, compared with the traditional redirection mechanism of honeyfarm in the centralized way. There are some future research direction in this topic such as the performance issue of the redirection mechanism and the issue of proxy caches. A possible extension of the redirection mechanism which combines URL rewriting to DNS-based is worth studying. Transport-layer requesting routing and application-layer request routing are two popular URL rewriting mechanisms that can be integrated with DNS-based requesting routing to not only redirect the suspected traffic to the honeyfarm, but also improve the accuracy of selection of the appropriate honeypot [7].

Acknowledgement

This research was partly funded by the National 863 Plan (Projects Numbers: 2003AA148020), P. R. China and the Internet Security project of PrivyLink International Limited (Singapore).

References

1. Athena Vakali, George Pallis, *Content delivery Networks: status and trends Aristotle in IEEE Internet Computing*, 2003, pp. 68-74.
2. Aline Baggio, Maarten van Steen, "Transparent Distributed Redirection of HTTP requests", *Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, 2003.
3. Irwin Lazar, William, *Exploring Content Delivery Networking*, July/August 2001, pp. 47-49.
4. Lance Spitzner, *Honeypots: Tracking Hackers*, Addison Wesley, 2002, Chapter 3.
5. Ognjen Prnjat, Ioannis Liabotis, Temitope Olukemi, Lionel Sacks, "Policy-based Management for ALAN-Enabled Networks", *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, 2002.
6. Lance Spitzner, *Know Your Enemy: Sebek*, 1990.
7. Brad Cain, Fred Douglis, Mark Green, Markus Hofmann, Raj Nair, Doug Potter, Oliver Spatscheck, *Knowing CDN Request-Routing Mechanisms*, 2000.
8. Girish Borkar, *Content Distribution Networks*, 2002.

License Protection with a Tamper-Resistant Token*

Cheun Ngen Chong, Bin Ren, Jeroen Doumen, Sandro Etalle,
Pieter H. Hartel, and Ricardo Corin

EEMCS Faculty, University of Twente, Postbus 217, 7500 AE Enschede, The Netherlands
{chong,ren,doumen,etalle,pieter,corin}@cs.utwente.nl

Abstract. Content protection mechanisms are intended to enforce the usage rights on the content. These usage rights are carried by a *license*. Sometimes, a license even carries the key that is used to unlock the protected content. Unfortunately, license protection is difficult, yet it is important for digital rights management (DRM). Not many license protection schemes are available, and most if not all are proprietary. In this paper, we present a license protection scheme, which exploits tamper-resistant cryptographic hardware. The confidentiality and integrity of the license or parts thereof can be assured with our protection scheme. In addition, the keys to unlock the protected content are always protected and stored securely as part of the license. We verify secrecy and authentication aspects of one of our protocols. We implement the scheme in a prototype to assess the performance.

1 Introduction

In a digital rights management (DRM) system, we use a *license* to specify the rights of a user on digital content [7]. For example, a commercial software license could restrict the execution of the licensed software to a particular number of uses. We must ensure the integrity of this information, so that the usage rights can be enforced correctly.

A license often carries the key to unlock the protected content. Therefore, we must ensure the confidentiality of this key, so that a dishonest user cannot access the content without abiding by the license. Additionally, the license can also carry *metadata* of the content, which may be as valuable as the content itself because metadata is critical for efficient content management. For example, the URI of the film. To ensure the availability of the film, the integrity of the URI must be protected.

Sometimes, we must ensure the confidentiality of some license information so that it cannot be accessed by any unauthorized users. For instance, a bank but not a content distributor can access a user's payment information specified on a license, e.g. credit card number. Therefore, the license, just as the content, requires adequate protection.

Unfortunately, license protection does not attract as much attention as content protection. There are only a few license protection schemes available, and most if not all are proprietary. Our main security objectives are to ensure confidentiality and integrity of a license or parts thereof, so that keys and metadata can be protected.

In addition, we would like to enforce different usage rights on different parts of the content. For instance, a patient record contains sensitive information about a patient.

* This project is funded by Telematica Instituut, The Netherlands.

We want to protect and share this information by using different keys. The doctor is issued all the keys to access the entire patient record, but the insurance agent is only issued the keys needed to access insurance related information. To protect the patient record from being misused, we need to protect these keys.

In this paper, we propose a license protection scheme using a key tree and a tamper-resistant hardware token with which we are able to achieve the aforementioned objectives. A hardware token provides a tamper-resistant environment for storage and for cryptographic operations; while a key tree grants us the flexibility to protect and share a license and content.

Our contributions can be listed as follows:

1. We propose a license protection scheme using a key tree and a hardware token, which is able to protect the license or parts thereof and content parts.
2. We perform an analysis of our protection scheme to justify its security properties.
3. We implement and evaluate the license protection scheme by using some off-the-shelf software tools and a Java iButton.

We have applied our license protection scheme according to a number of usage scenarios. To explain our approach, we choose a scenario, where a provider (e.g. NYSE) issues a license that restricts access of brokers (i.e., paid subscribers) and normal users to specific information on stock prices.

The organization of the remainder of the paper is as follows: Section 2 lists the security requirements. Section 3 briefly explains LicenseScript. Section 4 discusses our license protection scheme. Section 5 explains our prototype implementation. Section 6 reports on a performance evaluation of the prototype to justify the applicability. Section 7 briefly explains some related work. Finally, section 8 concludes and suggests future work.

2 Security Requirements

We assume that some of the system components can be trusted. This is more or less realistic with consumer electronic (CE) devices, but much less realistic when working on personal computers. In particular, we assume that the application interprets a license correctly. We treat this trusted part of the application as a *reference monitor* [13]. For example, as soon as the license expires, the application stops rendering. However, a malicious application can still cheat by tampering with the license. Therefore, we define the following requirements for our license protection scheme:

Requirement 1 License Integrity: The application must verify the integrity of the license when it accesses the license.

Requirement 2 Token Interaction: The application must interact with the hardware token to access the license and content parts.

Requirement 3 Key Confidentiality: The storage keys for accessing the license and content parts must be hidden from the application.

When these requirements are fulfilled, cheating by tampering with the license will be difficult.

3 LicenseScript License

In this section, we discuss our licensing language, LicenseScript. The language is based on multiset rewriting [2] and logic programming [11]. The reader may refer to our previous work [4] for more detailed information.

A license has the following form:

```
license(Content,Clauses,Bindings)
```

Here, `Content` is (a link to) the content to be protected; `Clauses` is a Prolog program that decides if the operations performed are allowed or forbidden; and `Bindings` is a list of attributes that carry the status of the license and metadata of the content.

A clause has the following form:

```
Head :- Body_1,Body_2,...,Body_n.
```

Here, `Head` is the name and arguments of the clause, and the conjunction of `Body_1` up to `Body_n` is the body of the clause.

We use stock price scenario (as discussed in section 1) for illustration. Figure 1 is an example of a LicenseScript license that allows a broker to view a stock price for 10 times. The license also allows the provider to reset the number of times the stock price is viewed. Finally, the provider can update the stock price.

In Figure 1, `stock_price` is the link to the stock price; `get_value(X,Y,Z)` gets the value of the binding `Y` from the binding list `X` and unifies it with the variable `Z`; `set_value(V,X,Y,Z)` sets the value of the binding `X` from the binding list `V` with the value `Y` and stores the binding into the binding list `Z`; `is_member(X,Y)` checks if element `X` is a member of set `Y`; `get_curr_time(X)` gets the current time and stores

```
license(stock_price,
[(canreset(S,B1,B2):-
   S==provider,
   set_value(B1,viewed,0,B2)),
 (canupdate(S,B1,B2) :-
   S==provider,
   get_curr_time(T),
   set_value(B1,updated,T,B2)),
 (canview(S,B1,B2) :-
   get_value(B1,subjects,Ss),
   is_member(S,Ss),
   get_value(B1,viewed,X),
   get_value(B1,maxviews,Y),
   X <= Y, X = X + 1,
   set_value(B1,viewed,X,B2))],
[maxviews=10,
 viewed=0,
 updated=01012004,
 subjects=[broker]])
```

Fig. 1. A license that restricts a broker to access a stock price under 10 times.

```
license(stock_price,
[(canreset(S,B1,B2):-
   cipher("CJ...", skey1)),
 (canupdate(S,B1,B2) :-
   cipher("XY...", skey3)),
 (canview(S,B1,B2) :-
   cipher("AB...", skey4))],
[maxviews=cipher("12...",skey4),
 viewed=cipher("AC...",skey4),
 updated=01012004,
 skey1=cipher("89...",rootkey),
 skey2=cipher("aC...",rootkey),
 skey3=cipher("CC...",skey1),
 skey4=cipher("KL...",skey2),
 mac=cipher("XA...",rootkey),
 subjects=[(provider,rootkey),
           (broker,skey2),
           (alice,skey4)]])
```

Fig. 2. Protected license of Figure 1, storing the storage keys.

it in `X` (primitives for other useful environmental data exist as well); `viewed` is the binding that stores the number of times the stock price has been viewed; `maxviews` stores the maximum number of times the stock price can be viewed; `updated` records the time that `stock_price` is updated; `subjects` is the access control list of subjects that can view `stock_price`.

In all clauses, `S` represents the subject making the query, `B1` is the current set of bindings and `B2` is the set of bindings resulting from a successful query. A failed query does not update the bindings. Clauses are triggered via external actions, for example if the `broker` presses the view button on the user interface, the `canview` clause is activated, with the appropriate settings for `S` (i.e., `broker`) and the bindings `B1` and `B2`.

4 License Protection Scheme

In this section, we introduce our license protection scheme. We use the architecture shown in Figure 3.

Four components are involved: the *application*, *reference monitor*, *token*, and *provider*. The application is a piece of software that interacts with the token, and which is used to access the license and the associated content. The reference monitor, which is a *trusted* part of the application, coordinates the actions of the application and the license. Each of these components has its own public/private key pair.

Two protocols support the communication between the components. Protocol A is used to send a protected license to the application from the provider. The provider generates the protected license and depending on its business model, decides which part of the license needs to be protected.

Protocol B is used when the application starts using the content, and when the reference monitor interprets the protected license. We will elaborate these protocols later in section 4.3. To use the license, the application must interact with the token and the reference monitor.

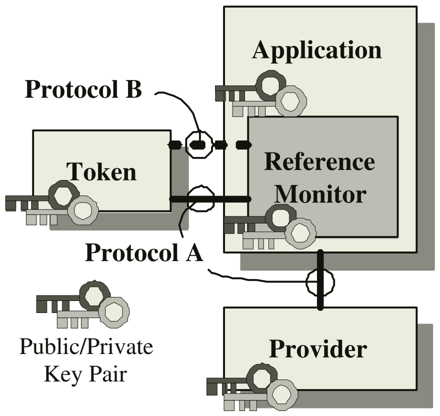


Fig. 3. Overall license protection architecture.

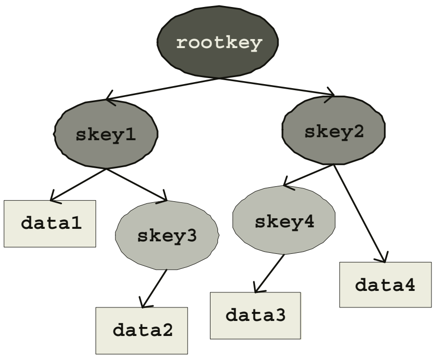


Fig. 4. An example of key tree.

We will explain our scheme as follows: Firstly, we look at protected storage mechanisms, which have inspired our license protection scheme in section 4.1. Secondly, we show the structure of the protected license in section 4.2. Lastly, we illustrate the protection scheme protocols that we have developed in section 4.3.

4.1 Protected Storage Mechanisms

Our license protection scheme is a *protected storage mechanism*. Protected storage is defined by Pearson et al [12] as follows:

> Protected storage is a service to the host platform in which the trusted platform module (TPM) acts as a portal to confidential data stored on an arbitrary, unprotected storage media.

Here, the TPM is a tamper-resistant cryptographic hardware module that is permanently embedded in a computer. The TPM can provide secure storage for keys and other sensitive information and it can perform cryptographic operations. Our license protection scheme uses an external hardware token instead of the TPM because this allows user the freedom to move licenses and content between machines.

In this paper, we use protected storage in the form of a *key tree*. This is a mechanism that has been used in secure group communication for key distribution and management [10].

Figure 4 provides an example of a key tree. A child node is encrypted using the storage key of the parent node. The root key is the “master key” for the whole tree. If say, `skey1` is needed to decrypt `data1`, the former will be decrypted using the `rootkey`. Then, `data1` can be decrypted with `skey1`.

For optimal performance, we use symmetric keys for the root key and the storage keys. The root key is stored on the token when it is issued and never leaves the token. It is sent to the user physically with the token. This root key is the secret key shared between the token and the provider. All decryptions take place on the hardware token for maximum security.

However, when sharing license information with another user, an actual storage key (which has become the root key for a sub-tree) must be transferred to the user’s token. For instance, we can allow a user to *only* access `data3` and `data4` by transferring the actual storage key `skey2` to the user’s token. The process of transferring this storage key to another user’s token falls outside the scope of this paper. However, we believe it can be achieved by using the TPM maintenance mechanism [12], which is intended to transfer a storage key from a TPM to another TPM securely. In addition, we can exploit a secure transfer mechanism, such as the mechanism proposed by Atallah and Li [1]. This deserves further study.

We can selectively deploy the information of the license with other entities by using the key tree. For instance, we can share the information of the license encrypted by `skey3` and hide the other information from another user, by using the key `skey2` as the root key for that user.

4.2 Protected License

By using a key tree, we can protect the license from Figure 1, as shown in Figure 2. Here, `cipher(X,Y)` is a predicate that stores the encrypted value `X` (which looks meaningless to human eyes) with the key `Y`.

Several additional bindings are needed to store the encrypted keys. For example, `skey1` is the encrypted key used to encrypt lines `03–04` of Figure 1, shown as `"CJ..."` in line `03` of Figure 2. The length of `"CJ..."` is the same as the original text shown in lines `03–04`. We use the binding `mac` (line `15`) to store the message authentication code, i.e., `"XA..."` of the license, which can be verified by using the `rootkey`. The `mac` can be used to ensure the integrity of the entire license.

The `provider` has the `rootkey` (on her token) so that the `provider` can access all the encrypted data in the license. Therefore, the `provider` can execute clause `canreset`, which resets the value of the binding `viewed`; clause `canview`, which views the content of `stock_price`; and clause `canupdate`, which updates the content of `stock_price`. On the other hand, the `broker` can only execute the clause `canview`, because her token only has the actual key `skey2`.

In addition, we use the keys in this license to protect some information (i.e., parts) of the `stock_price`. The license only allows an authorized user (with the correct key) to access these protected parts. In Figure 2, `broker`, who is a paid subscriber, can access the `stock_price` information that is encrypted with key `skey2` and `skey4`. The user `alice`, who is not a paid subscriber can only access the information encrypted with key `skey4`.

4.3 Protocols

In this section, we describe the protocols of our protection scheme: Protocol A (for transmitting the license) and Protocol B (for using the license). For the reader's convenience, we list the notation we use to describe the protocols in Table 1.

Protocol A requires interaction between the hardware token, the application, and the license provider. Its two main objectives are : (1) to send the protected license to the application; and (2) to send the public key of the application to the token. The application's public key will certify the trustworthiness of the application when the license is used.

$$\text{A1. } A \rightarrow T : \{A, P, \text{“}name\text{”}\}$$

$$\text{A2. } T \rightarrow A : \{N, MAC(N, K_{(P,T)}), T, A, P, \text{“}name\text{”}\}_{K^{+}_{eP}}$$

$$\text{A3. } A \rightarrow P : \{A, \{N, MAC(N, K_{(P,T)}), T, A, P, \text{“}name\text{”}\}_{K^{+}_{eP}}\}$$

$$\text{A4. } P \rightarrow A : \{Lic, \{N+1, A, K^{+}_{eA}\}_{K^{+}_{eT}}\}$$

$$\text{A5. } A \rightarrow T : \{N+1, A, K^{+}_{eA}\}_{K^{+}_{eT}}$$

A1 Application (A) asks Token (T) to get the desired license (identified by “$name$”) from Provider (P). T must recognize P.

A2 T generates a fresh nonce N, a MAC of N (using the secret key shared with P), $K_{(P,T)}$, concatenates N, $MAC(N, K_{(P,T)})$ and identity T to the message received from A, encrypts the result with Provider's public key K^{+}_{eP}, and sends this to A. The fresh nonce is

Table 1. The notation.

Symbol	Meaning
A	Application
R	Reference Monitor (the trusted part of A)
T	Hardware token
P	Provider
D	Data, i.e., license clause/binding or content
$\{X, Y\}$	Concatenation of X and Y
$\{\cdots\}_K$	Message is encrypted by key K
K_{st}	Storage key
K_{ss}	Session key
$K_{(X,Y)}$	Shared secret key of X and Y
K_{eX}^+, K_{eX}^-	Public and private key of X for encryption
K_{sX}^+, K_{sX}^-	Public and private key of X for signature
$S(M)_{K_{sX}^-}$	Signature of M with K_{sX}^-
$MAC(M, K)$	MAC of message M with K
Lic	License
Key	List of encrypted storage keys

necessary to prevent a replay attack. The secrecy of the message can be assured by encrypting it with K_{eP}^+. The authenticity of the nonce (i.e., that it was produced by T) is guaranteed by $MAC(N, K_{(P,T)})$. Therefore, a malicious application cannot fabricate the message of step A2 without help of the token.

A3 A sends its identity and the received message from T to P.

A4 If P can decrypt the received message, P is implicitly authenticated by T. P increments N, concatenates $N+1$ with A's public key K_{eA}^+ and encrypts the result with T's public key K_{eT}^+. P sends this message and the protected license Lic to A.

A5 A forwards the encrypted message to T and stores Lic. The license can be stored securely because its content is protected by a key tree.

$$\begin{aligned}
&\text{B1. } A \to T : \{A, Lic, MAC(Lic, K_{(P,T)})\}\\
&\text{B2. } T \to A : \{K_{ss_1}\}_{K_{eA}^+}\\
&\text{B3. } A \to T : \{Key, \{D\}_{K_{st}}, \text{“}param\text{”}\}_{K_{ss_1}}\\
&\text{B4. } T \to R : \{\{Lic, D, S(D)_{K_{sT}^-}\}_{K_{ss_2}}, \{K_{ss_1}, K_{ss_2}\}_{K_{eR}^+}\}\\
&\text{B5. } R \to A : \{D\}_{K_{ss_1}}\\
&\text{B6. } A \to R : \{D'\}_{K_{ss_1}}\\
&\text{B7. } R \to T : \{D'\}_{K_{ss_2}}\\
&\text{B8. } T \to A : \{\{D'\}_{K_{st}}\}_{K_{ss_1}}\\
&\text{B9. } A \to T : \{Lic'\}_{K_{ss_1}}\\
&\text{B10. } T \to A : \{MAC(Lic', K_{(P,T)})\}_{K_{ss_1}}
\end{aligned}$$

Protocol B After Protocol A has finished, the application has the protected license. Each time the license is used, Protocol B is run.

As stated before, the reference monitor is assumed to be the trusted part of the application. We trust the reference monitor in the sense that it will correctly interpret each license. Also, we assume that the token has obtained, and trusts, the public key of the reference monitor initially. The steps of Protocol B are:

B1 Application (A) wants to access the license. It initiates the interaction by sending to Token (T) its identity A, the license Lic and the MAC value of the license, $MAC(Lic, K_{(P,T)})$. If validation fails, T terminates the interaction and records the event.

B2 T verifies the integrity of Lic. If the integrity is violated, T terminates the interaction with A, and A cannot access the content. We use secure audit logging to record this incident [6]. If the integrity is validated, T acknowledges A with a randomly generated fresh session key K_{ss}, encrypted with A's public key K^{+}_{eA}. Implicitly, the application is authenticated if the application can read this message using its private key.

B3 A retrieves a list of encrypted storage keys Key needed for the required data $D_{K_{st}}$, and sends it to T. The parameter "$param$" is used to identify the type of D, i.e., if D is a clause, "$param$" is the name of the clause. This message is encrypted with the session key K_{ss} to ensure the authenticity of the session.

B4 Before sending the decrypted data D, two cases are considered:

B4.1 If D is a license binding, T checks it against the previously stored value to assure that D has not been tampered with. If the check fails, the token terminates the transaction. Otherwise, T performs *application-specific* updates on the binding value stored on the token. For instance, the value of binding `played_times` is incremented. In any case, we log the binding values so that when T and A re-connect to P (Protocol A) say, for a new license or content, T sends P the stored binding values, so that P can check if the user has cheated [7]. If D is used for the first time, T will store it (T can trust the integrity of D at the *first time* because D is always encrypted with a storage key).

B4.2 If D is a license clause or a content part, no checking is done due to the limited resources of the token.

Then, T sends D and its signature $S(D)_{K^{-}_{sT}}$ as well as the license Lic encrypted with a new session key K_{ss_2} to the reference monitor R. The new session key is encrypted with the public key of R, i.e., K^{+}_{eR}.

B5 R verifies $S(D)_{K^{-}_{sT}}$ before interpreting the data D. Then, R sends D to A, encrypted with the session key K_{ss_1}. The encryption with K_{ss_1} is to ensure the authenticity of the session.

B6 After A has used and updated D (i.e., D' is generated), A sends D' to R.

B7 R checks if D' was updated correctly. If so, R sends D' to T encrypted with their shared session key K_{ss_2}.

B8 T replies to A with the encrypted D', i.e. $\{D'\}_{K_{st}}$. T encrypts it with the session key K_{ss_1} to ensure the authenticity of the session.

B9 A new license Lic' is re-constructed by A. A asks T to regenerate a new MAC value for the updated license Lic'.

B10 T sends the new $MAC(Lic', K_{(P,T)})$ to A to finish the final re-construction of Lic'.

Steps B3 to B8 may be repeated in a session for different types of data (i.e., license clause, binding or content part) during the use of the license and content.

This completes the description of the protocols.

4.4 Formal Protocol Verification

We have used the protocol verifier CoProVe [8] to verify Protocol A. Basically, what we needed to verify is that a malicious application would not be able to obtain the license without the correct intervention of the token. It is well-known that design of cryptographic protocols is error-prone, and that a great deal of published protocols have later been shown to contain errors prejudicing their safety. CoProVe helps to find possible attacks and at proving that – under certain conditions – a protocol is attack-free. CoProVe works by taking as input a specification of the protocol and a system scenario describing the roles involved in the protocol – in our case token, application and provider of Protocol A – and by analysing all possible interleavings in presence of a malicious intruder.

We adopt two reasonable assumptions to keep our scenario simple yet expressive:

1. The token knows the genuine provider. It is a practical assumption because the token is dispatched by the provider.
2. The provider knows the genuine application. It is also practical because the provider keeps a list of authorized applications that can access the license.

Specifically, we have verified the following properties:

1. Secrecy: A fresh nonce must only be known by the token and the genuine provider. This is to prevent replay attacks.
2. Authentication: The malicious application and provider cannot impersonate the genuine ones. Thereby, the malicious application cannot impersonate a genuine one to decoy the token. We tested that a malicious application could not impersonate the token.

To carry out the verification we had to set up a finite-state scenario (consisting of a finite number of parallel sessions); this is the standard limitation of model-checking approach to verification: while we have checked scenarios with two parallel sessions (we are going to carry out tests with 3 parallel sessions as well) it is possible – though unlikely – that hidden flaws are revealed only by analyzing scenarios with a higher number of parallel sessions.

4.5 Security Analysis

In this section, we review the requirements of section 2 corresponding to our license protection scheme.

Requirement 1 is satisfied by using a message authentication code. The verification of the MAC value is performed on the token with the root key stored on the token (in Protocol B step B1). Therefore, we can ensure the correctness of the MAC verification because the secret key never leaves the token.

Requirement 2 is fulfilled *if* different parts of the content are encrypted by using different keys stored on the license (in Protocol B step B3). Therefore, the application must interact with the token *continuously* as long as the application accesses the content and license. Our protection scheme is aimed for content that is short-lived, i.e., the value of the content is reduced after a short period of time. For instance, stock prices. Therefore, once the content has been decrypted and presumably saved in the clear, we do not insist on communication with the token anymore.

Requirement 3 is satisfied. The keys stored on the license are encrypted. The decryption operations (on the keys, license clauses, bindings, and content parts) are performed on the token (in Protocol B step B4). However, during sharing, an actual storage key must be transferred from one token to another. This process is presumed secure by using additional protocols [1].

5 Prototype

In this section, we discuss a prototype implementation of our license protection scheme. The objective is to establish the applicability, and at the same time to conduct some performance evaluation. The prototype is built on a platform with an Intel Pentium 4, 256 Mbytes of RAM, and a serial port connection with the iButton version 2.2.

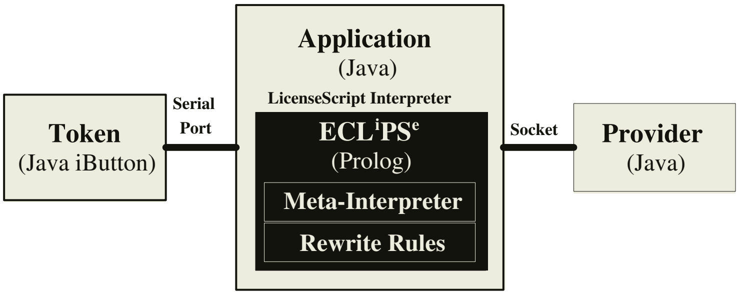


Fig. 5. The architecture and components of the reference implementation.

We use off-the-shelf software tools to implement the components of our prototype, as shown in Figure 5:

1. The LicenseScript Interpreter is responsible for interpreting and calculating licenses. It *acts* as a reference monitor. We have used the LicenseScript Interpreter from our previous work [5] based on:
   (a) ECL^{i}PS^e: To execute the Prolog code retrieved from the LicenseScript licenses.
   (b) Meta-interpreter: To retrieve the clauses and binding values from the licenses and to send these to the ECL^{i}PS^e Prolog interpreter.
   (c) Multi-set Rewrite Rules: To interpret the rights operations performed by the users via the application, for instance, play, copy, etc.
2. The Application is used to access the license and the associated content, while interacting with the token. This is written in Java, using the iB-IDE API, Java Cryptography Extension (JCE), and JavaCard Framework.
3. The Token is a Java iButton version 2.2. It has a higher physical security than a normal smart card because the chip is physically protected by a stainless steel cover, and it supports common cryptographic algorithms.
4. The Provider serves the protected license and sends it to the client via a socket connection. This is written in Java using JCE and Java.net.

After implementing the prototype, we performed several performance evaluations of the prototype.

6 Performance Evaluation

To verify the practicability of our license protection scheme, we perform several tests on our prototype.

From our previous experience, we know that the cryptographic operations on the iButton are slow [6]. As it is used more frequently than Protocol A, we choose to evaluate the performance of Protocol B, in particular the two operations that involve the iButton: (1) decryption of keys and data (includes license clause, binding and content part) on different levels of a key tree, and (2) reconstruction of the license, which involves encrypting the data and generating a message authentication code.

6.1 Test 1: Levels of the Key Tree

The depth of a key tree influences the performance of our license protection scheme. For instance, to retrieve a license binding value, which is encrypted with a storage key at level 10 of the key tree, the token has to perform 10 steps of symmetric decryption (including the step to decrypt the encrypted binding value).

In this test, we measure the decryption time required at various levels, i.e., from level 2 to 10 inclusively (level 1 is the root key). The final result obtained for each level is the average of 5 repeated measurements. We run the test as shown in Figure 6.

The size of the data is less than 128 bytes. We found that it takes roughly 0.2 second for DES decryption (with a 56-bit key) on the iButton, which is consistent with the findings of our previous work [6].

We give a least square fitting (LSQ-Fit) formula to express the results of our measurements:

$$t = 0.06 \pm 0.02 + (111 \pm 3) \times l \tag{1}$$

Here, t is the time in milliseconds required to perform DES decryption on the token for level l on the key tree. $l > 1$ because level 1 is the root key.

The first conclusion is that the depth of the key tree should be kept as low as possible. From Equation 1, it takes approximately 1.22 seconds to decrypt data (of size less than 128 bytes) at level 10 of the key tree. This will cause a delay to the system, which is noticeable to the user.

6.2 Test 2: License Reconstruction

After the data is used and updated, we also need to re-encrypt the data on the token, reconstruct the license on the application and generate a new MAC for the updated license on the token.

We run a test, as shown in Figure 7. We use the same data size as before (100 bytes) to perform our tests, and we run the same test for the same data 5 times. The final result is the average value of these 5 tests.

We use an LSQ-fit formula to express our result:

$$t = 2256 \pm 80 + (2.56 \pm 0.28) \times l \tag{2}$$

Here, t is the time in milliseconds required to reconstruct the license for an updated level l. The time required to reconstruct the license hardly depends on the depth of the data in the key tree because only one DES encryption is performed on the iButton.

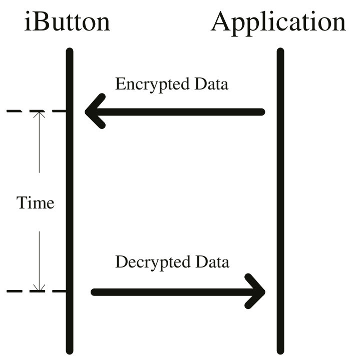


Fig. 6. The procedure for measuring the time needed to perform data decryption at different levels of the key tree.

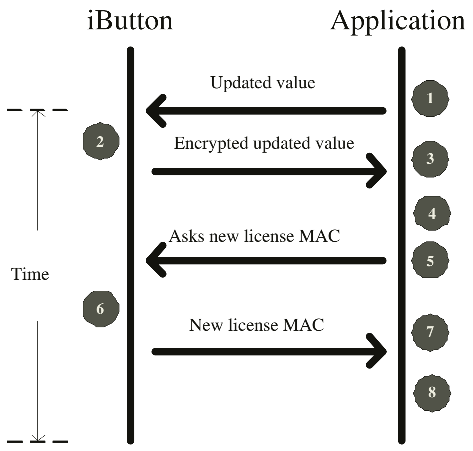


Fig. 7. The procedure for measuring the time needed to perform data re-encryption on the token and reconstruction license on the application, at different levels on the key tree.

Therefore, the time required to reconstruct the license for arbitrary updated level in the key tree is approximately 2.25 seconds.

We decompose the test into 8 parts, as shown in Figure 7, and measure the time required for each part:

Parts 1, 3, 5, and 7. The Application transmits data to the iButton and vice versa. We have run a test of the data transfer rate. It takes about 0.2 second to transmit less than 128 bytes of data, as shown in Figure 8. Our data size is about 100 bytes. In total the communication between the iButton and the application takes **0.8 second**.
The graph shown in Figure 8 leaps drastically at around 120 bytes of data size. This is due to the iB-IDE API. When the data is over 120 bytes, it will be split into chunks for transfer, which causes more transmit time.

Part 2. This corresponds to Protocol B step B5. The iButton needs to perform a DES decryption with a 56-bit session key on the message to retrieve the updated value, which takes about 0.2 second [6]. Then, the iButton encrypts the updated value with 56-bit storage key, and then with the 64-bit session key. Therefore, this process takes in total **0.6 second**.

Part 4. The application reconstructs the license with the encrypted and updated license data. This takes less than **0.05 second**.

Part 6. This corresponds to Protocol B step B7. Similar to step 2, it takes 0.2 second to decrypt an encrypted message with the session key. The iButton generates a MAC for the new license. The iButton needs about 0.15 second to generate a hash of the data size less than 128 bytes [6]. The iButton needs 0.2 second to generate the MAC with the root key (DES encryption of the hash). Lastly, the iButton needs 0.2 second to encrypt the MAC with the session key. Therefore, this step takes in total **0.75 second**.

Part 8. The Application reconstructs the license by embedding the new MAC. Similar to 4, the application takes less than **0.05 second** to finish this final step of license reconstruction.

To update and reconstruct a license, it takes in total approximately 2.25 seconds, which is consistent with the overall measurement reported at the beginning of this section.

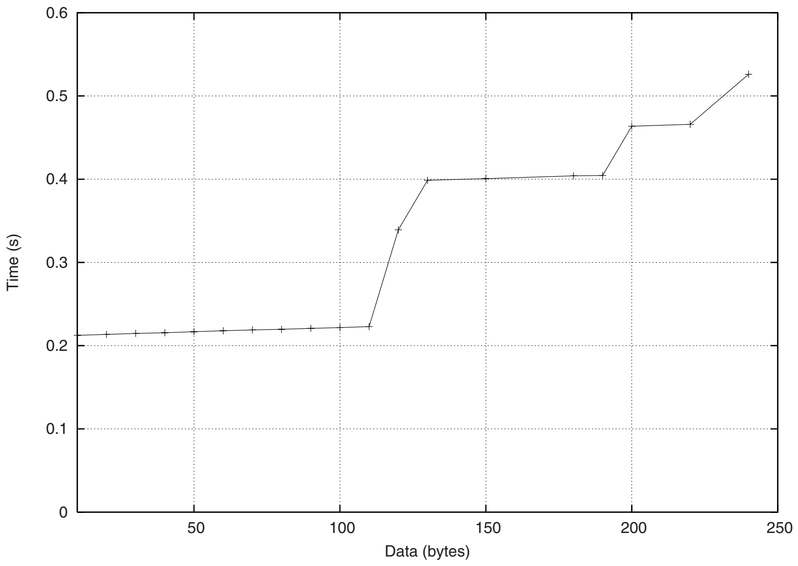


Fig. 8. The data transmission time from the application to iButton.

To conclude, the performance of the license protection scheme is acceptable from the user's perspective, if the data is small (less than 128 bytes). We may need a USB interface and a bigger token memory to handle bigger data. This remains for future investigation.

7 Related Work

In this section, we briefly discuss some related work. We investigate some XML documents security that XML-based rights expression languages exploit. We also discuss some commercial license protection mechanisms by using hardware tokens.

Damiani et al [9] define and implement an authorization model for regulating access to XML documents. They exploit the capabilities of XML, and define an XML markup for a set of security elements describing the protection requirements of XML documents. Bertino et al [3] share the objective of Damiani et al but they focus on controlling the data access and dissemination of XML documents when there are XML documents exchanges between two parties. They discuss main protection requirements posed by XML documents and present a set of authorization and dissemination policies to achieve the aforementioned purpose.

As far as we are aware, the listed authorization models only propose the representation of the protected XML documents, e.g. new structure with new set of XML markups, etc. There is no protection operation mentioned of how these protected XML documents are produced and accessed.

Several commercial proprietary protection schemes using hardware tokens are available. We are only able to scratch the surface of these mechanisms by studying their white papers. Most of them, e.g. Sospita (`http://www.sospita.com/`) and Wibu (`http://www.wibu.com/`) aim to protect software by executing parts of the code on their proprietary hardware tokens. They can unlock this part of the software if the user pays for it. These protection schemes also assume that (parts of) the application that interfaces with the tokens are trusted.

Basically, protecting a license is only a secondary task in their scheme. Different from LicenseScript, their licenses do not have a rich structure to express complex usage scenarios. In addition, our license protection scheme, because of a key tree, allows flexibility in sharing a protected license and content with other users.

8 Conclusions and Future Work

A *license* is an important element of digital rights management (DRM) because it: (1) specifies a user's rights on a digital content, (2) carries a content key, and (3) describes metadata of the content. To protect these valuable assets, we propose a license protection scheme based on a tamper-resistant hardware token and a key tree. The key tree provides flexibility and the hardware token provides tamper-resistance. We apply our license protection scheme to LicenseScript licenses. We analyze the protection scheme in terms of security with respect to some common security assumptions. We also perform a formal protocol verification using CoProVe.

We implement a prototype by using the Java iButton. To justify the practicability, we perform several measurement on the prototype. We conclude that the protection scheme is practical for a shallow key tree and small license size. We intend to extend our protection scheme for protecting fancy media, e.g. music or film. We will also use a USB connection for the iButton to improve the performance. Our scheme is intended for the business model of "one token to one provider" due to the limited resources of the token. However, we can extend our scheme to support "one token to many providers" – by using the public key of the token, we can generate a new shared secret key for a new provider. This remains as our future work.

References

1. M. J. Atallah and J. Li. Enhanced smart-card based license management. In *IEEE International Conference on E-Commerce*, pages 111–119. IEEE Computer Society, June 2003.
2. J-P. Banâtre, P. Fradet, and D. L. Métayer. Gamma and the chemical reaction model: Fifteen years after. In C. Calude, G. Paun, G. Rozenberg, and A. Salomaa, editors, *Workshop on Multiset Processing (WMP)*, volume 2235 of *LNCS*, pages 17–44. Springer-Verlag, Berlin, August 2001.
3. E. Bertino, S. Castano, E. Ferrari, and M. Mesili. Controlled access and dissemination of XML documents. In *Proceedings 2nd ACM Workshop on Web Information and Data Management (WIDM9́9)*, pages 22–27, 1999.
4. C. N. Chong, R. Corin, S. Etalle, P. H. Hartel, W. Jonker, and Y. W. Law. LicenseScript: A novel digital rights language and its semantics. In K. Ng, C. Busch, and P. Nesi, editors, *3rd International Conference on Web Delivering of Music (WEDELMUSIC)*, pages 122–129, Los Alamitos, California, United States, September 2003. IEEE Computer Society Press.
5. C. N. Chong, S. Etalle, P. H. Hartel, R. Joosten, and G. Kleinhuis. Service brokerage with prolog. Technical Report TR-CTIT-04-14, Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands, February 2004.

6. C. N. Chong, Z. Peng, and P. H. Hartel. Secure audit logging with tamper-resistant hardware. In D. Gritzalis, S. D. C. di Vimercati, P. Samarati, and S. K. Katsikas, editors, *18th IFIP International Information Security Conference (IFIPSEC)*, volume 250 of *IFIP Conference Proceedings*, pages 73–84. Kluwer Academic Publishers, May 2003.
7. C. N. Chong, R. van Buuren, P. H. Hartel, and G. Kleinhuis. Security attribute based digital rights management (SABDRM). In F. Boavida, E. Monteiro, and J. Orvalho, editors, *Joint Int. Workshop on Interactive Distributed Multimedia Systems/Protocols for Multimedia Systems (IDMS/PROMS)*, volume 2515 of *LNCS*, pages 339–352. Springer-Verlag, November 2002.
8. R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In M. V. Hermenegildo and G. Puebla, editors, *9th International Static Analysis Symposium (SAS)*, volume 2477 of *LNCS*, pages 326–341. Springer-Verlag, September 2002.
9. E. Damiani, S. de C. di Vimercati, S. Paraboschi, and P. Samarati. Securing XML documents. In *Advances in Database Technology - EDBT 2000, Proceedings 7th International Conference on Extending Database Technology*, volume 1777 of *Lecture Notes of Computer Science*, pages 121–135. Springer, March 2000.
10. J. Goshi and R. E. Ladner. Algorithms for dynamic multicast key distribution trees. In *Proceedings of of the twenty-second annual symposium on Principles of distributed computing*, pages 243–251. ACM Press, 2003.
11. J. W. Lloyd. *Foundations of Logic Programming*. Symbolic Computation – Artificial Intelligence. Springer-Verlag, 1987. Second edition.
12. S. Pearson, B. Balacheff, L. Chen, D. Plaqui, and G. Proudler. *Trusted Computing Platforms – TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, New Jersey 07458 United States, 2003.
13. R. Sandhu and J. Park. Towards usage control models: beyond traditional access control. In *7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 57–64. ACM, June 2002.

An Adaptive Approach to Hardware Alteration for Digital Rights Management

Yinyan Yu and Zhi Tang

National Key Laboratory of Text Processing Technology
Institute of Computer Science and Technology
Peking University, Beijing 100871, China
{yuyinyan,tangzhi}@icst.pku.edu.cn

Abstract. In most current digital rights management solutions, digital license is bound to the content rendering device using its hardware configuration. However, this strategy introduces an adaptive problem: protected contents can't be used any longer once the user changes the hardware configuration. This paper focuses on the problem and presents a feasible adaptive approach. With our approach, digital content buyers can still smoothly render the bought digital content without any extra operation when partial hardware replacement arises. Our approach balances the needs of customers' hardware alteration and providers' copyright protection on a reasonable level and increases the flexibility of DRM system.

1 Introduction

With the rapid development of digital network technology, digital content such as software, digital books, images, music and video can be easily duplicated without loss of quality and can be distributed instantaneously and extremely cheaply across the Internet to end-users. It becomes an immediate and important need to control and manage copyright of digital intellectual property. Digital rights management is regarded as a desirable solution to protect digital assets, control their distribution and usage and protect the rights of both the provider and the consumer.

In typical DRM solutions, digital content is always encrypted before distribution and a valid digital license containing the content decryption key is needed to render the content [1–7]. In order to protect the content from illegal copying, digital license is always tied to a specific rendering device using the unique identification of its hardware configuration [5–7]. But this approach introduces an unpleasing adaptive problem concerning the hardware alteration. Since the digital license is bound to a specific hardware configuration and the digital content can only be used on the device with the same hardware configuration used for creating the digital license, any change of the correlative hardware components may result in failure to use the bought digital content forever [6]. This adaptive problem apparently frustrates the consumers, deprives them of profit and causes obstacles in the success of DRM. To protect the digital content providers' profits and its users' interests, an adaptive solution that balances the requirements of both sides is under urgent need.

We propose a feasible adaptive approach by applying fault tolerance techniques for the first time. In particular, we use secret sharing to enable partial hardware replacement without the need for a license change. With our approach, digital content buyers can still smoothly render the bought digital content without any extra operation when partial hardware replacement arises. Our approach balances the needs of customers' hardware alteration and providers' copyright protection on a reasonable level and increases the flexibility of DRM system. As compared to the other solutions shown in Section 2, our approach has the following virtues:

1. The hardware alteration is transparent to digital content consumers and there is a grace tolerance scope. Within this scope, the consumer can change any of the hardware components of his rendering device and can still flowingly use the bought digital content without any extra operation as before;
2. The digital license itself is adaptive to hardware modification and no hardware configuration information is kept in the end-user's machine. And thus avoid the possible attack on the comparison point of the current approach.

The remainder of this paper is organized as follows. In Section 2, we analyze some related works. Then, we present our hardware adaptive approach in detail and prove that our scheme really protects rights of both providers and users in Section 3. Next, some discussions about our approach are given in Section 4. Finally, we present the main results of this paper in Section 5.

2 Related Works

Though there are many different DRM systems, the core idea is the same, i.e., using digital license to protect digital contents. After consumers get digital contents, corresponding digital license must be acquired before using the digital contents. Most DRM solutions are variations on a DRM reference architecture shown in Fig. 1 [1].

There are three major components in the reference architecture: the content server, the license server, and the client. **The content server** includes a digital content repository, a product information database and a DRM packager. The DRM packager has the functionality for encrypting and preparing content for distribution through the system. Another task of the DRM packager is to create descriptions of the rights that the content provider wants to allow the user to exercise on the content. The content encryption key and rights descriptions are sent to the license server. **The license server** includes a rights specification repository, an encryption keys repository, an identities repository and a DRM license generator. The generator is responsible for creation and distribution of digital license, which contains information about the identity of the user or device that wants to exercise rights to content, identification and encryption key of the content to which the rights apply, specification of those rights and the

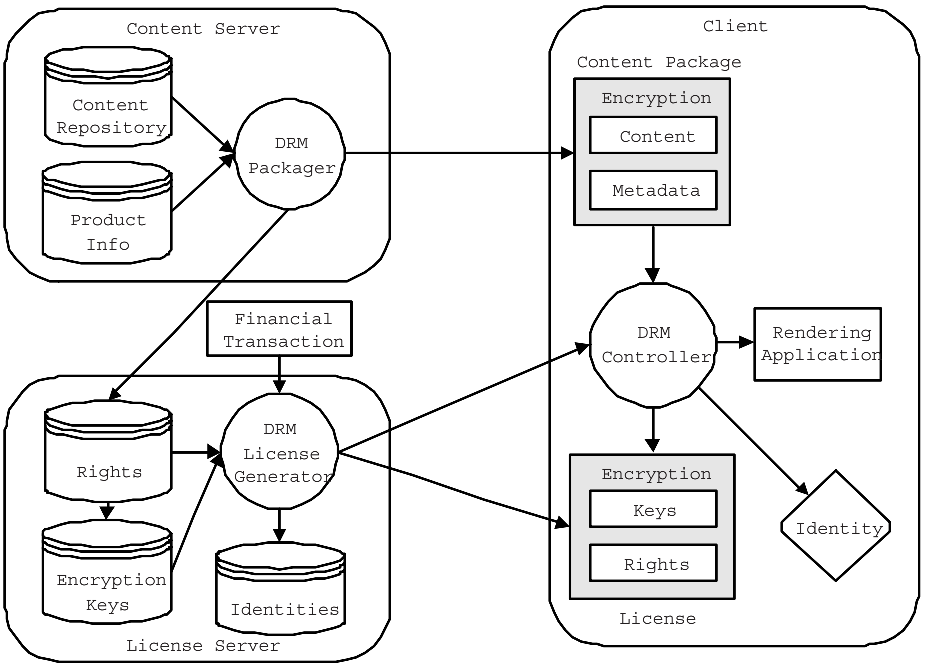


Fig. 1. The DRM reference architecture.

information of license issuer. Components of the DRM reference architecture that reside on **the client**'s side are the DRM controller, the rendering application, and the user's identification mechanism. The DRM controller is responsible for the lawful use of digital content and requesting digital license from the license server if no corresponding license exists in the client side.

Two strategies are often adopted in today's DRM solutions to prevent digital content from illegal copying. One is to bind the digital license to an extra secure physical device such as a smart-card [2–4]. The other is to tie the digital license to the rendering device by using the unique identification of its hardware configuration [5–7]. The advantage of these two strategies is that the digital content can not be shared among different rendering devices and pirate actions such as unlawful copy can be effectively controlled.

However, extra physical devices turn out to be unfriendly and unacceptable for mass market considering the consequent added costs for content providers and consumers, especially the digital content with lower value [8]. It becomes worse if these devices are broken or lost. Hence, binding digital license to a specific rendering device seems more reasonable. But this approach will lead the consumer fail to use the bought contents if he changes the hardware configuration of his rendering device [6].

To our knowledge, there is no solution to solve this adaptive problem in DRM system used to protect general digital contents such as e-books. But there exist two solutions in DRM system used to protect software. One solution to this problem is that the buyer contacts with the license server every time he changes the hardware configuration of his rendering device and requests the digital license once more, i.e. a license change is needed. This solution needs ongoing contact between the user and the license server. The license server will have to save and periodically update a record of the user and the device set. Users have to bear the unpleasing additional contact with the license server and can thus be tracked to some extent [9]. It will be worse if the user can't connect his device to the Internet in time.

Microsoft has adopted another approach to tolerate a certain degree of change in a hardware configuration in its Product Activation techniques. In Microsoft's Windows XP, the information of hardware configuration at the time of product activation is stored in the local host and the current hardware configuration is compared to the original hardware configuration at each login. Windows XP can run well if it judges that the hardware is not "substantially different" [10–11]. However, this manner may not be secure since the original hardware configuration is stored in the consumer's machine and the comparison point may be a hidden weak point, especially when we apply it to protect general digital content such as e-book, digital music and movie etc.

3 The Adaptive Approach to Hardware Alteration

Our approach is based on the architecture described in Fig. 1. To realize our adaptive object for the hardware alteration, a new approach to create digital license and recover content decryption key using hardware configuration will be employed in the architecture.

In the license creation stage, the license server fetches the corresponding content decryption key, divides it into n shares and encrypts each share with the relative piece of information of the hardware component received from the client, and then encapsulates the encrypted shares with other necessary information in a digital license file and delivers it to the client. In the key recovery stage, DRM controller in the client will retrieve at least t valid shares from the digital license file and reconstruct the content decryption key.

In this section, we will first give some notations used in our approach, and then detail our approach step by step. Finally, we will prove that our approach can really balance the needs of hardware alteration and copyright protection.

3.1 Notations

The notations in table 1 are used throughout this paper.

3.2 Our Approach

This section will describe the proposed adaptive approach to hardware alteration for DRM in detail. In the approach, we apply fault tolerance techniques to satisfy the needs of hardware alteration and copyright protection. In particular,

Table 1. Notations appeared in our approach.

Notations	**Meanings**
K_p, K_s	Public/private key pair of the license server
k_0	Shared secret key of the DRM controller and the license server
Ω	Set of the hardware components selected to bind digital license
Ω_0	Subset of Ω, consists of the hardware components that don't exist in the current rendering device
$m(m_0)$	Total number of the hardware components in $\Omega(\Omega_0)$
e_i	The ith hardware component of Ω
n_i	Number of shares bound to $e_i, n_i > 0$
$n(n_0)$	Total number of shares bound to the hardware components in $\Omega(\Omega_0)$, i.e. $n = \sum_{i=1}^{m} n_i, n_0 = \sum_{e_i \in \Omega_0} n_i$
$H(\cdot)$	One way hash function
$G(\cdot)$	Key generation function
$E(\cdot, K_p)$, $D(\cdot, K_s)$	Encryption of a message with an asymmetric encryption algorithm with public key K_p, and its relative decryption function
$\dot{E}(\cdot, K)$, $\dot{D}(\cdot, K)$	Encryption of a message with a symmetric encryption algorithm with symmetric key K, and its relative decryption function
$Sig_{K_s}(M)$	Signature over message M with private key K_s

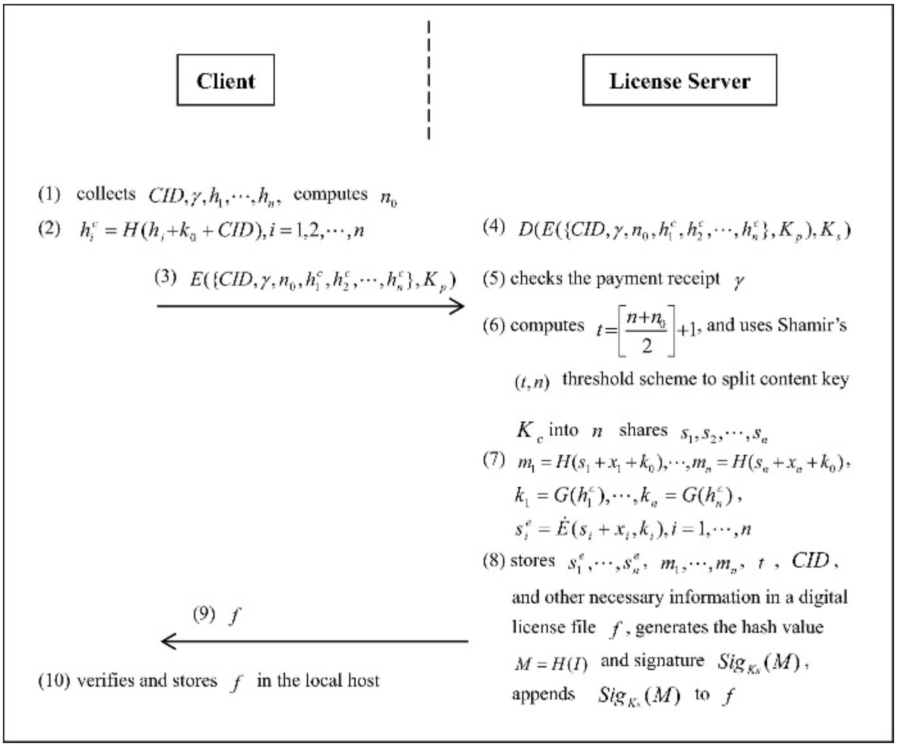
we adopt Shamir's (t, n) threshold sharing scheme [12]. Instead of binding the whole license or the content decryption key to the whole hardware configuration identification, we obtain identities of corresponding hardware components in Ω, split the content decryption key into multiple parts and tie them respectively to these identifies.

In our scheme, we make the following assumptions:
(H1): The length of the key of the selected symmetric encryption algorithm in our approach is not less than the length of the hash value $H(\cdot)$.
(H2): $G(\cdot)$ is an injective function, i.e. $G(x_1) \neq G(x_2)$ if $x_1 \neq x_2$.
(H3): $n > n_0 \geq 0$.

As noted before, our approach has two stages: the license creation stage and the key recovery stage. Details of these stages are described in the following.

License Creation. The process of license creation is described as follows (see Fig. 2):

(1) The DRM controller of the client collects digital content's identity CID, corresponding payment receipt γ (set to null if it doesn't exist) and n pieces of hardware information $h_1, h_2, \cdots, h_n$ of the hardware components in Ω in current device, where $h_1, h_2, \cdots, h_{n_1}$ is the information of the first hardware component $e_1 \in \Omega$, $h_{n_1+1}, h_{n_1+2}, \cdots, h_{n_1+n_2}$ is the information of the second hardware component $e_2 \in \Omega$, and so on, and computes $n_0 = \sum_{e_i \in \Omega_0} n_i$.
(2) The DRM controller computes the hash value of each piece of hardware information, $h_i^c = H(h_i + k_0 + CID), i = 1, 2, \cdots, n$.



(1) collects $CID, \gamma, h_1, \cdots, h_n$, computes n_0

(2) $h_i^c = H(h_i + k_0 + CID), i = 1, 2, \cdots, n$

(3) $E(\{CID, \gamma, n_0, h_1^c, h_2^c, \cdots, h_n^c\}, K_p)$

(4) $D(E(\{CID, \gamma, n_0, h_1^c, h_2^c, \cdots, h_n^c\}, K_p), K_s)$

(5) checks the payment receipt γ

(6) computes $t = \left[\frac{n+n_0}{2}\right] + 1$, and uses Shamir's (t, n) threshold scheme to split content key K_c into n shares $s_1, s_2, \cdots, s_n$

(7) $m_1 = H(s_1 + x_1 + k_0), \cdots, m_n = H(s_n + x_n + k_0)$, $k_1 = G(h_1^c), \cdots, k_n = G(h_n^c)$, $s_i^e = \dot{E}(s_i + x_i, k_i), i = 1, \cdots, n$

(8) stores $s_1^e, \cdots, s_n^e$, $m_1, \cdots, m_n$, t, CID, and other necessary information in a digital license file f, generates the hash value $M = H(I)$ and signature $Sig_{K_s}(M)$, appends $Sig_{K_s}(M)$ to f

(9) f

(10) verifies and stores f in the local host

Fig. 2. The process of license creation and information transfer.

(3) The DRM controller encrypts CID, n_0, γ and $h_1^c, h_2^c, \cdots, h_n^c$ with the public key of the license server: $E(\{CID, \gamma, n_0, h_1^c, h_2^c, \cdots, h_n^c\}, K_p)$, and submits the result to the license server.

(4) The license server decrypts the received information with its private key: $D(E(\{CID, \gamma, n_0, h_1^c, h_2^c, \cdots, h_n^c\}, K_p), K_s)$.

(5) The license server checks the payment receipt γ. If the digital content is not free and the payment receipt γ is null or the receipt γ can't prove that digital content CID has been paid or the receipt γ is not generated by an authorized credit organization such as a bank or retailer, the license server will kick off a financial transaction and the client will securely receive and store a payment receipt signed by an authorized credit party after the consumer pays for the content.

(6) The license server verifies the validity of the digital content. If the verification fails, a failure message will be delivered to the DRM controller and this stage stops. Otherwise, the license server fetches the decryption key K_c of content CID from its key repository and computes $t = \left[\frac{n+n_0}{2}\right] + 1$. Then it selects a prime p such that $K_c \in Z_p, n < p$ and chooses n nonzero variants $x_1, x_2, \cdots, x_n$, which are different to each other, and uses Shamir's (t, n) threshold scheme [12] to split the secret K_c into n shares $s_1, s_2, \cdots, s_n$.

(7) The license server computes $m_1 = H(s_1+x_1+k_0), \cdots, m_n = H(s_n+x_n+k_0)$, generates n symmetric keys $k_1 = G(h_1^c), \cdots, k_n = G(h_n^c)$ and encrypts the concatenation of each share and its variant: $s_i^e = \dot{E}(s_i + x_i, k_i), i = 1, \cdots, n$.
(8) The license server stores $s_1^e, \cdots, s_n^e, m_1, \cdots, m_n, t, CID$, license version, the information of the license server and the information about authorized rights and conditions with other necessary information in a digital license file f, generates the hash value $M = H(I)$, where we denote as I the whole information in current digital license file f, signs the result with its private key $Sig_{K_s}(M)$, and then appends $Sig_{K_s}(M)$ to f.
(9) The license server delivers the resultant digital license file f to the client.
(10) The DRM controller of the client verifies the digital license file by using the license server's public key, and then saves f in the local device. If the verification fails, a resending request will be sent to the license server.

Key Recovery. Key recovery process can be regarded as the reverse process of the license creation stage given above.

(1) The DRM controller fetches content CID's digital license file f, verifies its integrity and collects n corresponding hardware information $h_1^{'}, h_2^{'}, \cdots, h_n^{'}$ with the same way adopted by step 1 of the license creation stage, and computes $h_i^{c'} = H(h_i^{'} + k_0 + CID), i = 1, 2, \cdots, n$.
(2) The DRM controller obtains $s_1^e, \cdots, s_n^e, m_1, \cdots, m_n$ and t from f.
(3) The controller generates n symmetric keys $k_1^{'} = G(h_1^{c'}), \cdots, k_n^{'} = G(h_n^{c'})$ and obtains n shares $s_1^{'}, s_2^{'}, \cdots, s_n^{'}$ and their corresponding variants $x_1^{'}, x_2^{'}, \cdots, x_n^{'}$ by decrypting the encrypted values $s_1^e, \cdots, s_n^e$: $\dot{D}(s_1^e, k_1^{'}), \cdots, \dot{D}(s_n^e, k_n^{'})$.
(4) The controller computes $m_1^{'} = H(s_1^{'} + x_1^{'} + k_0), \cdots, m_n^{'} = H(s_n^{'} + x_n^{'} + k_0)$, and then compares these resultant values with $m_1, \cdots, m_n$ and concludes that share $s_i^{'}$ and its variant $x_i^{'}$ valid if $m_i^{'} = m_i$, i.e. $s_i^{'} = s_i$ and $x_i^{'} = x_i$.
(5) If there are at least t valid coordinate pairs $(x_i^{'}, s_i^{'})$, content decryption key K_c can be reconstructed using Shamir's (t, n) threshold scheme [12]. And key recovery fails if there are less than t valid coordinate pairs.

In our scheme, each piece of the hardware information of the hardware component $e_i \in \Omega$ will be set to zero (or some other default value) if e_i does not exist in the current rendering device, i.e. $h_{n_1+\cdots+n_{i-1}+1} = \cdots = h_{n_1+\cdots+n_{i-1}+n_i} = 0$ if $e_i \in \Omega_0$. And when it comes to the rendering device that has more than one hardware component of the same type, we will choose the one that operates currently in the license creation stage and will try each in the key recovery stage. For example, if there is more than one hard disk in the rendering device, the one where the running operating system is installed will be selected by our approach in the first stage. And in the key recovery stage, each disk will be tried to get the relative share.

3.3 Proof

We will prove here that our approach can really balance the needs of hardware alteration and copyright protection. Without loss of generality, we assume:
(H4): $0 < n_1 \leq n_2 \leq \cdots \leq n_m$.

Theorem 1 (Hardware Alteration). *With the hardware adaptive approach, we have the following conclusions if $n_m \leq n - t$ holds:*

i. As compared with the hardware configuration of the rendering device the time the digital license is requested and generated, the consumer can change any $m - j + 1$ or less hardware components of the rendering device without influencing the use of the digital content;
ii. The consumer may fail to render the protected digital content if he changes not less than $m - j + 2$ hardware components,

where $1 < j \leq m$ satisfies:

$$\begin{cases} \sum\limits_{i=j}^{m} n_i \leq n - t \\ \sum\limits_{i=j-1}^{m} n_i > n - t \end{cases} \tag{1}$$

Proof. With $n = \sum\limits_{i=1}^{m} n_i, t = \left[\frac{n+n_0}{2}\right] + 1$ and assumption (H3), we can easily conclude that such an integer j satisfying $1 < j \leq m$ and inequations (1) exists if $n_m \leq n - t$ holds.

i. Assume that $m - j + 1$ or less hardware components of the rendering device have been changed since the time the digital license is generated, at least $j - 1$ hardware components in Ω remain unaltered. According to assumption (H4), at least $\sum\limits_{i=1}^{j-1} n_i$ pieces of hardware information remain the same. We can conclude from the stages of our adaptive approach that the system can obtain at least $\sum\limits_{i=1}^{j-1} n_i$ valid coordinate pairs. For

$$\sum_{i=1}^{j-1} n_i = \sum_{i=1}^{m} n_i - \sum_{i=j}^{m} n_i = n - \sum_{i=j}^{m} n_i,$$

we can confirm from inequations (1) that $\sum\limits_{i=1}^{j-1} n_i \geq t$. So the content encryption key can be recovered using the (t, n) threshold scheme [12] and the digital content can still be smoothly used.
ii. Assume that $m - j + 2$ or more hardware components of the rendering device have been changed since the time the digital license is generated, the last $m - j + 2$ hardware components in Ω may be changed. That is to say, $\sum\limits_{i=j-1}^{m} n_i$ pieces of hardware information used in creating the digital license may be changed. According to assumption (H1) and (H2), different hardware information will generate different key, thus a valid share and its variant can only be recovered by using the same hardware information used to encapsulate them. So the system may fail to recover $\sum\limits_{i=j-1}^{m} n_i$ valid coordinate pairs,

i.e. perhaps the system can only recover $n - \sum_{i=j-1}^{m} n_i$ valid coordinate pairs. Hence, the system may fail to reconstruct the content decryption key according to inequations (1) and (t, n) threshold scheme [12] and the consumer then fail to render the protected digital content. □

Theorem 2 (Hardware Alteration). *Suppose that each piece of the hardware information collected in our approach can uniquely identify the corresponding hardware component and $n_1 \le n - t$, the following conclusions hold:*

i. *As compared with the hardware configuration of the rendering device used to generate the digital license, the relative digital content can't be used if the consumer changes k or more hardware components in Ω;*
ii. *The digital content may be well rendered if the consumer changes not more than $k - 1$ hardware components,*

where $1 < k \le m$ satisfies:

$$\begin{cases} \sum_{i=1}^{k-1} n_i \le n - t \\ \sum_{i=1}^{k} n_i > n - t \end{cases} \tag{2}$$

Proof. With $n = \sum_{i=1}^{m} n_i, t = \left[\frac{n+n_0}{2}\right] + 1$, and assumption (H3), we can easily conclude that such an integer k satisfying $1 < k \le m$ and inequations (2) exists if $n_1 \le n - t$ holds.

i. Assume that the consumer has changed k hardware components in Ω of his rendering device since the time the digital license is generated, $m - k$ hardware components in Ω remain unchanged. Since each piece of the hardware information collected in the hardware adaptive approach can uniquely identify the corresponding hardware component, i.e. different hardware component has different hardware information, $\sum_{i=k+1}^{m} n_i$ pieces of hardware information collected by the system remain the same at best according to assumption (H4). Since a valid share and its variant can only be recovered by using the same hardware information used to encapsulate them (see the proof of theorem 1), the system can obtain at most $\sum_{i=k+1}^{m} n_i$ valid coordinate pairs. For $\sum_{i=1}^{k} n_i > n - t$ and $\sum_{i=1}^{m} n_i = n$, $\sum_{i=k+1}^{m} n_i < t$. So content encryption key cannot be recovered according to (t, n) threshold scheme and the digital content cannot be used. We can obtain the same conclusion if the client changes more than k hardware components in Ω of his device.
ii. Assume that the consumer has changed not more than $k - 1$ hardware components of his rendering device since the time the digital license is generated, the last $m - k + 1$ hardware components in Ω may remain unchanged. That

is to say, $\sum_{i=k}^{m} n_i$ pieces of hardware information used in creating the digital license may remain the same. We can conclude that the system may obtain $\sum_{i=k}^{m} n_i$ valid coordinate pairs. For $\sum_{i=1}^{m} n_i = n$ and $\sum_{i=1}^{k-1} n_i \leq n - t$, $\sum_{i=k}^{m} n_i \geq t$. So the content encryption key may be recovered using the (t, n) threshold scheme and the digital content can be well used. □

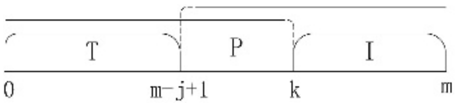


Fig. 3. Tolerance scope in hardware alteration.

Fig. 3 presents the results of theorem 1 and theorem 2, i.e. the tolerance scope in hardware alteration of our hardware adaptive approach upon the premises that $n_m \leq n - t$ and each piece of the hardware information collected in the approach can uniquely identify the corresponding hardware component. In Fig. 3, natural number j satisfies inequations (1) and k satisfies inequations (2). The field $[0, m - j + 1]$ occupied by the capital character "T" is the tolerance scope, which means that the protected content can be well used as long as the total number of the altered hardware components belonging to Ω is within this scope; the field $[k, m]$ occupied by the character "I" is the intolerance scope; and the filed $(m - j + 1, k)$ occupied by the character "P" is the possible tolerance scope.

Theorem 3 (Copyright Protection). *With the hardware adaptive approach, the generated digital license can't be shared among different rendering devices if all pieces of the hardware information collected in the approach can uniquely identify the corresponding hardware component.*

Proof. Assume that the digital license generated by the proposed scheme can be shared between two different rendering devices $\mathcal{A}$ and $\mathcal{B}$, i.e. both hardware configuration of device $\mathcal{A}$ and $\mathcal{B}$ can be used to reconstruct the content decryption key, we will show that this assumption leads to a contradiction.

Since a valid share and its variant can only be recovered by using the same hardware information used to encapsulate them (see the proof of theorem 1), only the device with at least t pieces of same hardware information used to generate the digital license can retrieve the content decryption key. So $\mathcal{A}$ and $\mathcal{B}$ totally have at least $2t$ pieces of same hardware information used to generate the digital license.

On the other hand, different hardware component has different hardware information upon the premise that all pieces of the hardware information collected in the proposed approach can uniquely identify the corresponding hardware component. For $\mathcal{A}$ and $\mathcal{B}$ can have at most $m + m_0$ same hardware components used to create the digital license, $\mathcal{A}$ and $\mathcal{B}$ have totally $n + n_0$ pieces of same hardware information used to create the digital license at best.

Summing up the results above, we can conclude $2t \leq n + n_0$. This is contrary to $t = \left[\frac{n+n_0}{2}\right] + 1$. Thus, the digital license generated by our scheme can not be shared among different rendering devices. □

We can easily deduce the following corollary from the proofs of the theorems above:

Corollary 1. *If we replace the value of parameter t in the hardware adaptive approach with any integer in the semi-close set $[\left[\frac{n+n_0}{2}\right] + 1, n)$, theorem 1 and theorem 2 will still hold. And theorem 3 holds if $t \in [\left[\frac{n+n_0}{2}\right] + 1, n]$.*

4 Discussions

Our proposed approach relies mainly on the security of the underlying encryption schemes and the use of Shamir's (t, n) threshold sharing scheme will not lessen the security of the system. Well known secure encryption schemes can be used to ensure the security of the protocol and the content. Besides, different encryption algorithms and different hash functions can be adopted to increase the security of the secret shares. Since license is bound to a specific device, replay attack will not help the attacker access the protected content and mechanisms can also be introduced to pretend the license server from deny of service attack. A forgery attack will also fail for a valid payment receipt can only be generated by an authorized credit organization. Though the digital license file f is sent in plain form from the license server to the client, an attacker must first attack the DRM controller, obtain its secret key and learn the mechanism of collecting the identification information of the hardware components. Tamper resistance mechanisms can also be adopted to ensure the security of the system.

As compared with the existing adaptive solutions, our approach needs no additional connection between the client and the license server and keeps no hardware information in the local machine. It is transparent to the consumer, and partial modification of the hardware configuration of the rendering device will not influence the use of the bought contents. Though our approach also has check points, which are used to judge whether the decrypted shares are valid, the attack of these points makes no-good and the copyrighted content is still secure. Unlike the proposed adaptive approach, the attack of the hardware check point in the adaptive approach used in Windows XP will help the malicious user access the protected content.

However, security and flexibility does not come without costs. This is the case not only in the present context, but also in general. One shortcoming in our approach is that splitting encryption key into multiple pieces will put additional workload on the license server and increase the degree of complexity of the system. An in-depth study on this issue is underway. It is also important to note that privacy should be enhanced in our proposed approach. Though hardware information is concealed with a hash function and different hardware information will be sent to the license server when the user requests digital licenses for different digital contents in our approach, the use of the open content identity

CID can still help the license server or a malicious user build link between the hashed hardware information and the bought contents to some extent. Studies are being made to apply the randomization method [4] or some other "blinded" scheme to our approach so that the privacy of the hardware information of the consumer's device and the information of the bought contents can be safer.

5 Conclusions and Future Work

In this paper, we have presented a new approach using Shamir's (t, n) threshold sharing scheme to solve the hardware adaptive problem in digital rights management. In our approach, the consumer can change any $m - j + 1$ or less hardware components if $n_m \leq n - t$ and the bought digital content can still be flowingly used without any extra operation, where j satisfies inequations (1). Meanwhile, the generated digital license can't be shared among different rendering devices if all the hardware information of the selected hardware components in the proposed scheme can uniquely identify the hardware component. So the proposed approach balances the requirements of hardware alteration and copyright protection on a reasonable level. As discussed in Section 4, the analysis of the complexity of our approach is our future work. And to enhance patron's privacy safety, we will also improve the approach by applying some "blinded" scheme.

References

1. William Rosenblatt, William Trippe, Stephen Mooney: Digital Rights Management: Business and Technology, New York: M&T Books, (2002).
2. T. Aura, D. Gollmann: Software License Management with Smart Cards. In: Proceedings of the USENIX Workshop on Smart Card Technology, Chicago, Illinois, USA, (1999), 75-85.
3. M. J. Atallah, Jiangtao Li: Enhanced Smart-card based License Management. In: Proceedings of the IEEE International Conference on E-Commerce (CEC'03), IEEE Computer Society, Newport Beach, California, (2003), 111-119.
4. C. Conrado, F. Kamperman, G. J. Schrijen, W. Jonker: Privacy in an Identity-based DRM System. In: Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03), IEEE Computer Society, Prague, Czech Republic, (2003), 389-395.
5. Qiong Liu, Reihaneh Safavi-Naini, Nicholas Paul Sheppard: Digital Rights Management for Content Distribution. In: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, Adelaide, Australia, (2003).
6. R. Davis: The Digital Dilemma. Communications of the ACM, 44(2), (2001), 77-83.
7. Microsoft Corporation, Andrea Pruneda: Windows Media Technologies: Using Windows Media Rights Management to Protect and Distribute Digital Media. http://msdn.microsoft.com/msdnmag/issues/01/12/DRM/default.aspx.
8. Tomas Sander: Good Times for Digital Rights Management? In: Proceedings of the 5th International Conference on Financial Cryptography. Lecture Notes in Computer Science 2339, Berlin: Springer-Verlag, (2002), 55-65.

9. Joan Feigenbaum, Michael J. Freedman, Tomas Sander, Adam Shostack: Privacy Engineering for Digital Rights Management Systems. In: Proceedings of the 2001 ACM Workshop on Security and Privacy in Digital Rights Management. Lecture Notes in Computer Science 2320, Berlin: Springer-Verlag, (2002), 76-105.
10. Microsoft Corporation: Technical Details on Microsoft Product Activation for Windows XP. http://www.microsoft.com/technet/prodtechnol/winxppro/evaluate/xpactiv.mspx.
11. Fully Licensed GmbH: Inside Windows Product Activation. A Fully Licensed Paper. (2001). http://www.licenturion.com/xp/fully-licensed-wpa.txt.
12. A. Shamir: How to Share a Secret. Communications of the ACM, 22(11), (1979), 612-613.

Dynamic Fingerprinting over Broadcast Using Revocation Scheme

Mira Kim[1], Kazukuni Kobara[2], and Hideki Imai[2]

[1] Institute of Information Security,
2-14-1, Tsuruya-cho Kanagawa-ku, Yokohama 221-0835, Japan
mira@iisec.ac.jp

[2] Institute of Industrial Science, The University of Tokyo,
4-6-1, Komaba Meguro-ku, Tokyo 153-8505, Japan
{kobara,imai}@iis.u-tokyo.ac.jp

Abstract. In this paper, we deal with the problem of how to embed unique fingerprints into broadcasted contents or packaged contents, such as CD and DVD, without giving the watermarking algorithm to the decoders at the receivers. We propose a new model using around-half-rate dynamic revocation scheme in broadcast encryption. Our model achieves the following properties: (1) No watermarking algorithm at the decoders; (2) Dynamic fingerprinting; and (3) Dynamic revocation. Revocation schemes allow a center to broadcast an encrypted message so that only a particular subset of the users can obtain the contents of the message. However, when dealing with around-half-rate revocation, most past proposed schemes failed to obtain a good efficiency, i.e. the message length that must be transmitted by the sender or the number of storage keys at a user is too large. In addition, we propose an efficient algorithm of revocation that reduces both the message length and the size of storage keys at a user while maintaining both collusion-freeness and a single decryption at a user.

1 Introduction

Broadcast encryption schemes are techniques that allow a center to transmit encrypted data to a large set of receivers so that only a particular subset of privileged receivers can decrypt it. Such schemes are useful for pay-TV systems, multicast communication, distribution of copyrighted material (e.g. video, music and so on) on encrypted CD/DVD disks, among others. Broadcast encryption schemes deal with the tracing mechanism, *traitor tracing schemes*, which enables to trace the source of keys used by an illegal device such as pirate decoders. However, those do not deal with the illegal redistribution of the decrypted content by a legitimate user (e.g. to rebroadcast the decrypted content). If any legitimate user redistributes the decrypted content by himself/herself, a center cannot trace the source of the illegal redistribution.

Fingerprinting schemes are helpful techniques in protecting digital contents against illegal redistribution. These schemes embed unique data per user (i.e.

an ID) into copies of a digital content and if an illegal copy appears, allow the content distributor to trace the source of the illegal redistribution. However, it is hard to straightforwardly combine fingerprinting and broadcast encryption schemes. While in broadcast encryption schemes each user receives the same content, in fingerprinting schemes each user receives slightly different contents.

In this paper, we discuss a broadcast encryption scheme which prevents the illegal redistribution of contents. [3] and [5] have been proposed as answers to such problem. In [3], Boneh and Shaw proposed a distribution scheme in which a user receives a bulk of data common to all users, and a small number of extra bits unique to him. The bulk of data refers to the content which is watermarked and encrypted and the extra bits refers to the private key which can decrypt some part of the broadcasted common data. In [5], Fiat and Tassa proposed a dynamic model that fingerprints are generated on the fly according to the feedback from the pirate distribution network.

In this paper, we propose a novel way to fingerprint broadcasted data using a *revocation scheme*. Revocation schemes allow a center to exclude a subset of users from obtaining a message. Our model achieves the following properties: (1) No watermarking algorithm at the decoders; (2) Dynamic fingerprinting that the embedding fingerprinting codes can be changed dynamically; and (3) Dynamic revocation that any subset of users can be revoked temporality or eternally. For our model, it is required that on the order of 1/2 the users of a broadcast encryption scheme can be revoked at all times. While our idea can in principle employ any revocation scheme, such as [9], [6] and [8], they are not suitable for it since they do not have a good performance when around half of users is revoked. So that we propose a new scheme supporting efficient revocation of half of the participants.

1.1 Classification of Fingerprinting Schemes

Now, we classify fingerprinting schemes into four types (Figure 1) and describe each type below in detail.

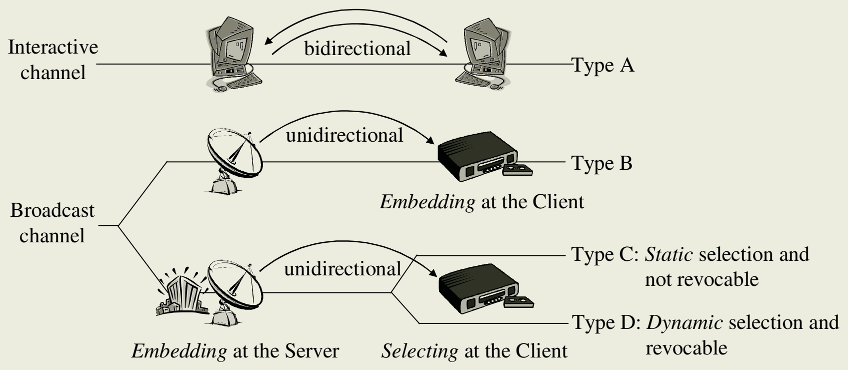


Fig. 1. The classification of fingerprinting schemes.

Type A. This scheme is the simplest one. Users are connected to a content distributor with interactive channels such as the Internet. Whenever a user requests a content, the distributor authenticates the user first, embeds the user's ID into the content and then sends it to the user. This scheme can hide the watermarking algorithm from the users, but is not suitable for dealing with heavy requests since the embedding is done after request.

Type B. A distributor broadcasts clear contents and then each decoder embeds its ID into the contents. For example, the watermarking algorithm is built in the decoder at a user and whenever the received content is decoded, a fingerprint is embedded into that. The big disadvantage of this is that the watermarking algorithm is in the users' side. Some users may reverse-engineer them, create and distribute the watermark erasing algorithm. Note that it is not so difficult to make an erasing algorithm if the watermarking algorithm is known. While good tamper resistant modules may prevent the reverse-engineer, it is still hard to make perfectly tamper resistant modules with low costs.

Type C. Unlike Type B, the watermarking algorithm is kept in the server side (and then the users select the watermarked contents according to their ID). This can hide the watermarking algorithm from the users. When a content, say a movie, is broadcasted to a group of N users, the simplest way to achieve that would be as follows:

Step 1. Make N copies of the movie and then embed unique fingerprints in each copy.
Step 2. Encrypt each copy with each user's key.
Step 3. Broadcast all the N encrypted copies to all the users.

It is easy to see that this construction achieves a fingerprinting scheme over broadcast without giving the watermarking algorithm to the decoders at the receivers. Only the legitimate N users receive the fingerprinted movie, and then even if some of them redistribute the fingerprinted one they can be traced since unique fingerprinters are associated with the users.

This scheme, however, must broadcast N copies of the movie, which is almost impossible for large N, say one million. Here, we present an improved scheme to reduce the broadcast overhead.

Step 1. Divide the movie into l segments.
Step 2. Make two copies of each segment. We arrange them in a $2 \times l$ matrix as follows:

$$S = \begin{bmatrix} seg_{11} \; seg_{12} \; seg_{13} \cdots seg_{1l} \\ seg_{21} \; seg_{22} \; seg_{23} \cdots seg_{2l} \end{bmatrix},$$

where seg_{ij} denotes the j-th segment in the i-th copy of the movie.
Step 3. Embed a watermark denoting '0' in seg_{1j} $(1 \leq j \leq l)$ and '1' in seg_{2j} $(1 \leq j \leq l)$, respectively. We call each watermarked segment a *variant* and denote it var_{ij} for i $(1 \leq i \leq 2)$ and j $(1 \leq j \leq l)$.

Step 4. Encrypt each variant var_{ij} with a key K_{ij} that is pre-assigned to each user, i.e. each user has a set of l keys $\{K_{i1}, \cdots, K_{il}\}$ that is associated with his ID:

$$V_{enc} = \begin{bmatrix} E_{K_{11}}(var_{11}) \cdots E_{K_{1l}}(var_{1l}) \\ E_{K_{21}}(var_{21}) \cdots E_{K_{2l}}(var_{2l}) \end{bmatrix},$$

where $E_K(M)$ denotes an encryption function which encrypts M with the key K.

Step 5. Broadcast V_{enc}.

It is easy to see that l bit ID is embedded in the decrypted content from each user's decoder using 2 copies of the movie instead of N copies in the previous solution.

Unfortunately, this scheme still has disadvantages. One is that there is no way to exclude compromised keys from receiving new broadcast contents. Another disadvantage is that only static fingerprints are embedded repeatedly but dynamic ones are not. For example, let $l = 5$ and a set of keys $\mathcal{K}_1 = \{K_{11}, K_{12}, K_{23}, K_{14}, K_{25}\}$ be pre-assigned to a user U_1 in the former case. Then U_1 will always get contents where the fixed fingerprint of "00101" is embedded repeatedly. Moreover the center cannot revoke $\mathcal{K}_1$ even if it is disclosed to the Internet since to stop encrypting with it means to block the honest users receiving parts of the contents. Thus this scheme cannot revoke compromised keys.

Type D. This is our proposal that can revoke compromised keys and can vary embedding IDs dynamically. To realize this, we propose to broadcast K_{ij} to each half of the users using a revocation scheme that can revoke half of the users. For example, if the center broadcasts K_{11} using a half-rate revocation scheme to users U_1 and U_2 excluding U_3 and U_4 and then broadcasts K_{21} to U_3 and U_4 excluding U_1 and U_2, it can embed '0' to the contents from both U_1 and U_2 and '1' from U_3 and U_4, respectively. If it broadcasts K_{11} to U_2 and U_3 and K_{21} to U_1 and U_4, then it can embed '0' to the contents from U_2 and U_3 and '1' from U_1 and U_4. Thus it can change the embedding fingerprint dynamically. It can also exclude malicious users by giving them neither K_{1i} nor K_{2i}. In spite of the simplicity of our idea we will see in the subsequent section that it is quite effective.

1.2 Related Work

Revocation schemes allow a center to exclude a subset of users from obtaining a message. These schemes can be realized using broadcast encryption schemes that allow the center to deliver the message only to the compliment of a revoked subset. Several works in this topic can be found[11–13, 9, 6, 8, 1] to improve issues such as

- transmission overhead (i.e. message length)
- key size (i.e. the size of storage keys for one receiver)

– operation costs (to retrieve a session key[1])
– collusion immunity to revoked users

and so on.

In this paper, we focus on the schemes that are collusion-free and that do not require any modular exponentiation. Works dealing with a similar setting are [9, 6, 8].

In [9], Naor et al. proposed two efficient schemes using the full binary tree structure. They named them the Complete Subtree (CS) method and the Subset Difference (SD) method, respectively. Later on, Halevy et al proposed in [6] the Layered Subset Difference (LSD) method that improves the size of storage keys of SD while maintaining almost the same message size[2] and operation costs for a receiver. In [8], Nakano et al proposed the Power Set (PS) method that reduces the message length by extending CS to the a-ary tree structure.

Even though all of them, CS, SD, LSD and PS, have a good performance for a small (and a large) number of revocations, unfortunately these are not the case for half-rate revocations (see Section 4 for more details). Thus we propose a new scheme that is suitable for half-rate revocations.

1.3 Our Results

In order to embed fingerprinting codes in broadcasted contents, we propose a new model having the following features:

No Watermarking Algorithm at the Decoders: A broadcaster (or a center) can embed unique fingerprinting codes in the decoded contents without giving any watermarking algorithm to them.

Dynamic Fingerprinting: A broadcaster can change the embedding fingerprinting codes dynamically. This means the model can employ any dynamic fingerprinting codes, such as [5], as well as static ones, such as [3].

Dynamic Revocation: A broadcaster can revoke any subset of users temporality or eternally.

Even though the new model requires a broadcast encryption scheme that can revoke half of the subscribers dynamically, past schemes, i.e. CS, SD, LSD and PS, are not suitable for this purpose since they do not have good performance for half-rate revocations[3].

Thus we also propose a new broadcast encryption scheme that is suitable for the half-rate dynamic revocation. Our scheme is efficient in terms of the message length and the size of storage keys at a user while maintaining collusion-freeness and a single decryption (of a symmetric-key encryption scheme) at a user.

[1] A session key in this paper means a key of encrypting the whole or a part of a content.

[2] The exact message length of LSD is slightly larger than that of SD even though they have the same order.

[3] This is not a disadvantage for the normal usage of these protocols since the number of revoked decoders is usually small.

This paper is organized as follows: In Section 2, we describe a new model with the contents traceability using a half-rate dynamic revocation scheme. In Section 3, we describe the framework for our revocation scheme and a revocation scheme suitable for a half-rate dynamic revocation. We evaluate the efficiency of our scheme in Section 4. Finally we conclude in Section 5.

2 A Model for Fingerprinting Broadcasted Contents

In this section, we describe our broadcasting model that can embed a unique fingerprint in every decoded content of every decoder. As described in Subsection 1.3, it has nice features: (1) no watermarking algorithm at the decoders, (2) dynamic fingerprinting and (3) dynamic revocation.

Let N be the total number of users. We assume that every user has a unique secret information which is predistributed through a secure channel or secure storage such as IC card, individually. The unique secret information consists of keys of a revocation scheme for broadcast encryption which is not changed until the termination of the system. Let *center* be the content provider or the content owner. Let *content* be the original content to be sent by the center. We assume that the content is composed of l pieces called *segments*, i.e. $content = (seg_1\ seg_2\ \cdots\ seg_l)$.

For such a content, the center makes contents to be broadcasted (or packaged) as follows.

Step 1. Divide the content into l segments.

Step 2. Make t ($t \geq 2$) copies of each segment. We arrange them in a $t \times l$ matrix as follows:

$$S = \begin{bmatrix} seg_{11} & seg_{12} & \cdots & seg_{1l} \\ seg_{21} & seg_{22} & \cdots & seg_{2l} \\ \vdots & \vdots & \cdots & \vdots \\ seg_{t1} & seg_{t2} & \cdots & seg_{tl} \end{bmatrix},$$

where seg_{ij} denotes the j-th segment in the i-th copy of the content.

Step 3. Embed a watermark denoting '0' in seg_{1j} $(1 \leq j \leq l)$, '1' in seg_{2j} $(1 \leq j \leq l)$ and $i-1$ in seg_{ij} $(1 \leq j \leq l)$, respectively. We call each watermarked segment a *variant* and denote it var_{ij} for i $(1 \leq i \leq t)$ and j $(1 \leq j \leq l)$.

Step 4. Encrypt every variant var_{ij} with a unique session key[4] K_{ij}.

$$V_{enc} = \begin{bmatrix} E_{K_{11}}(var_{11}) & \cdots & E_{K_{1l}}(var_{1l}) \\ E_{K_{21}}(var_{21}) & \cdots & E_{K_{2l}}(var_{2l}) \\ \vdots & \cdots & \vdots \\ E_{K_{t1}}(var_{t1}) & \cdots & E_{K_{tl}}(var_{tl}) \end{bmatrix},$$

where $E_K(M)$ denotes an encryption function which encrypts M with the key K.

[4] A session key in this paper means a key of encrypting the whole or a part of a content.

Step 5. Broadcast V_{enc}.
Step 6. For $1 \leq j \leq l$,
  Step 6-1. Divide N users into t disjoint subsets $\boldsymbol{U}_{i,j}$ $(1 \leq i \leq t)$.
  Step 6-2. Broadcast K_{ij} so that all the users in $\boldsymbol{U}_{i,j}$ can decrypt it using a revocation scheme that revokes users in $\boldsymbol{R} = \boldsymbol{N} \backslash \boldsymbol{U}_{i,j}$, where $\boldsymbol{N}$ is a set of all the N users.

In Step 6-1, if the center wants to permanently revoke some malicious users who e.g. conspired to generate pirated decoders or contents, it divides N users into $t+1$ disjoint subsets $\boldsymbol{U}_{i,j}$ $(1 \leq i \leq t+1)$ where $\boldsymbol{U}_{t+1,j}$ is the set for the permanently revoked users who cannot receive any contents.

In this model, we assume that $N \leq t^l$. t can be any positive integer greater than 1, but large t is not appropriate since broadcasting t copies of a content consumes a channel. While larger t can shorten the length of embedding fingerprinting codes, this advantage is small since the center embed long codes dynamically using a lot of contents. Thus we focus on the case of $t = 2$ below.

Figure 2 shows an example of the model for $t = 2$. In this figure, white squares containing '0' or '1' express segments where '0' or '1' is embedded, i.e. variants. The variants are encrypted with session keys K_{ij} and then the keys K_{ij} are broadcasted using *Header* generated by a revocation scheme. The decoder of a user u has a unique set $\boldsymbol{I}_u$ of predistributed keys, and using it either K_{1j} or K_{2j} is decrypted as long as he/she is not revoked permanently. You can see that a binary sequence of length l is embedded in the decoded content of each user.

The center not only can change the embedding sequence by changing the header each time, but also can revoke compromised decoders from receiving newly distributed contents.

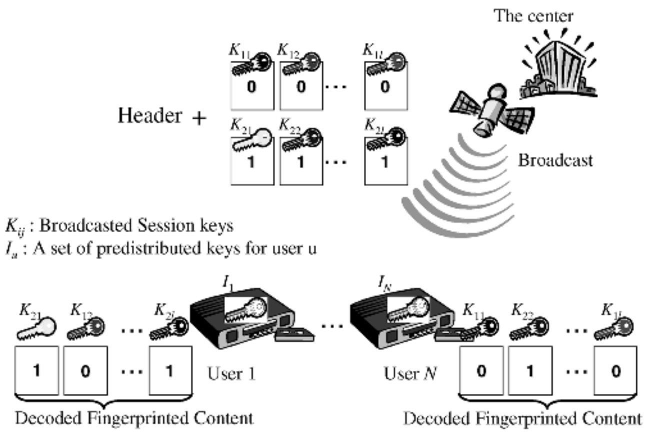

The center
Header +
K_{11}
K_{12}
K_{1l}
0
K_{21}
K_{22}
K_{2l}
1
Broadcast
K_{ij} : Broadcasted Session keys
I_u : A set of predistributed keys for user u
I_1
I_N
User 1
User N
Decoded Fingerprinted Content
Decoded Fingerprinted Content

Fig. 2. The model: $t = 2$.

3 Our Revocation Scheme

The new model in Section 2 requires a revocation scheme in Step 6-2. The past schemes, i.e. CS, SD, LSD and PS, are however not suitable for this purpose especially for $t = 2$ since they do not have good performance for the half-rate revocation. Thus, we propose an efficient broadcast encryption scheme that is suitable for the half-rate dynamic revocation. We explain the framework first.

3.1 Framework

Let a letter in boldface, e.g. $\boldsymbol{A}$, denote a set and $|\boldsymbol{A}|$ denote the cardinality of $\boldsymbol{A}$.

Let $\boldsymbol{N}$ denote a set of all the N users and $\boldsymbol{R}$ denote a set of revoked users in $\boldsymbol{N}$. $\boldsymbol{R}$ may be dynamically changed. Let $r = |\boldsymbol{R}|$, then $r \approx N/2$ in the previous model of $t = 2$. Let $\boldsymbol{N}_1, \boldsymbol{N}_2, \cdots, \boldsymbol{N}_{N/n}$ be disjoint groups of users such that $\boldsymbol{N}_i \subset \boldsymbol{N}$ and $|\boldsymbol{N}_i| = n$. For simplicity, we assume n divides into N, i.e. $n|N$. Let $\boldsymbol{S}_i$ be the set of non-empty subsets of $\boldsymbol{N}_i$ and thus $|\boldsymbol{S}_i| = 2^n - 1$ for $1 \leq i \leq N/n$. Let $\boldsymbol{S}_i[j]$ for $1 \leq j \leq 2^n - 1$ denote $2^n - 1$ distinct members of $\boldsymbol{S}_i$. Finally, we assign a unique session key $L_i[j]$ to $\boldsymbol{S}_i[j]$ for $1 \leq i \leq N/n$ and $1 \leq j \leq 2^n - 1$, respectively.

User Key Assignment. Upon initializing the system, the center distributes private information $\boldsymbol{I}_u$, which is a set of user keys, to a user u. The set of keys assigned to each user u, $1 \leq u \leq N$ is defined as follows.

$$\begin{aligned}\boldsymbol{I}_u := \{&\text{For } 1 \leq i \leq N/n \text{ and } 1 \leq j \leq 2^n - 1,\\ &\quad\text{the key } L_i[j] \text{ corresponding to } \boldsymbol{S}_i[j] \text{ s.t. } u \in \boldsymbol{S}_i[j]\}.\end{aligned}$$

Message Generation for Revocation. Given a session key K and a group $\boldsymbol{R}$ of users to be revoked, the center generates a header as follows.

Step 1. The center finds a collection of subsets $\boldsymbol{S}_i[j]$ such that $\boldsymbol{N}\backslash\boldsymbol{R} = \boldsymbol{S}_{i_1}[j_{i_1}] \cup \boldsymbol{S}_{i_2}[j_{i_2}] \cup \cdots \cup \boldsymbol{S}_{i_m}[j_{i_m}]$ for $1 \leq i_1, i_2, \cdots, i_m \leq N/n$ and $1 \leq j_1, j_2, \cdots, j_m \leq 2^n - 1$ where m denotes the number of subsets $\boldsymbol{S}_i[j]$ needed to form $\boldsymbol{N}\backslash\boldsymbol{R}$ and $0 \leq m \leq N/n$.

Step 2. The center encrypts the session key K with the key $L_i[j]$ corresponding to the found subset $\boldsymbol{S}_i[j]$ in Step 1. Its ciphertext is given by $E_{L_i[j]}(K)$.

Step 3. The center broadcasts

$$\begin{aligned}\{&\{i_1, j_{i_1}\}, \{i_2, j_{i_2}\}, \cdots, \{i_m, j_{i_m}\},\\ &\quad E_{L_{i_1}[j_{i_1}]}(K), E_{L_{i_2}[j_{i_2}]}(K), \cdots, E_{L_{i_m}[j_{i_m}]}(K)\}.\end{aligned}$$

Message Decryption. Upon receiving the broadcasted data such as

$$\{\{i_1, j_{i_1}\}, \{i_2, j_{i_2}\}, \cdots, \{i_m, j_{i_m}\}, C_1, C_2, \cdots, C_m\},$$

each user u $(1 \leq u \leq N)$ performs the following:

Step 1. The user u finds an index $\{i, j\}$ such that $u \in \boldsymbol{S}_i[j]$.

Step 2. The user u retrieves the corresponding key $L_i[j]$ from his/her private information $\boldsymbol{I}_u$ and then decrypts the corresponding ciphertext C_i with it. Thus u can obtain the session key K.

Tracing of Traitors. If pirated decoders are found, we can take the following two approaches to trace the traitors who generated the pirated decoders (even if the keys inside are protected by some tamper resistant modules.) One approach is to use the generic traitor tracing method proposed in [9], which is applicable to any tree-based revocation schemes including ours. The other one is to analyze the fingerprinting codes embedded in the contents decoded from the pirate decoder in our model of Section 2. To trace its supporting traitors based on the fingerprinting codes found in the pirate copy, we assume that the watermark algorithm is robust and the embedded mark cannot be changed or removed such as [5] and [10]. Since our model can dynamically vary embedding fingerprinting codes using a revocation scheme, any fingerprinting codes including static one [3] and dynamic one [5] are available.

3.2 The Method

Initializing System. We consider a tree of height 2 with N leaves. In such a tree, each internal node has n children and each leaf corresponds to each user. Figure 3 shows an example of the tree structure for $N = 9$ and $n = 3$.

Let v_i $(1 \leq i \leq N/n)$ denote an internal node of the tree. The leaves of the subtree rooted at an internal node v_i corresponds to the members of the user group $\boldsymbol{N}_i$ for i $(1 \leq i \leq N/n)$. For i $(1 \leq i \leq N/n)$ and j $(1 \leq j \leq 2^n - 1)$, the center associates every subset $\boldsymbol{S}_i[j]$ with a key $L_i[j]$ which is chosen randomly and independently. Then a user u receives the keys associated with the subsets $\boldsymbol{S}_i[j]$ which he/her belongs to. Table 1 shows an example of the key assignment method for $n = 3$. Thus the number of keys given to each user is 2^{n-1}. In the example depicted in Figure 3, the set of keys given to the user u_2 and the user u_6 are $\boldsymbol{I}_2 = \{L_1[2], L_1[4], L_1[6], L_1[7]\}$ and $\boldsymbol{I}_6 = \{L_2[3], L_2[5], L_2[6], L_2[7]\}$, respectively.

The key size of our scheme can be reduced by applying the key derivation scheme[2]. In [2], Attrapadung et al have introduced a key derivation scheme

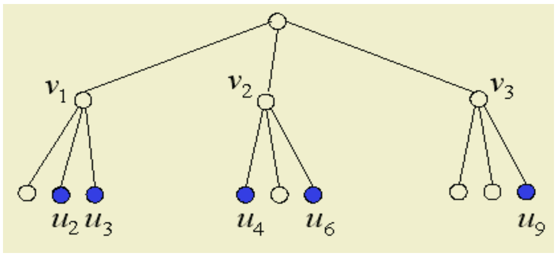


Fig. 3. The subset scheme: The gray circles express the users to be revoked. $N = 9$ and $n = 3$.

Table 1. The key assignment for each user (in the case of $n = 3$): Each user is assigned $2^{3-1} = 4$ keys. "◯" means the user belongs to the subset, and thus the corresponding key is assigned to the user. "×" means the key is not assigned to the user.

Subset	Key	1st User	2nd User	3rd User
$\boldsymbol{S}_i[1]$	$L_i[1]$	◯	×	×
$\boldsymbol{S}_i[2]$	$L_i[2]$	×	◯	×
$\boldsymbol{S}_i[3]$	$L_i[3]$	×	×	◯
$\boldsymbol{S}_i[4]$	$L_i[4]$	◯	◯	×
$\boldsymbol{S}_i[5]$	$L_i[5]$	◯	×	◯
$\boldsymbol{S}_i[6]$	$L_i[6]$	×	◯	◯
$\boldsymbol{S}_i[7]$	$L_i[7]$	◯	◯	◯

Table 2. Comparison of the performance (N, a and n are the numbers of total users, branches at each vertex of the tree and the size of the group of users, respectively. Message length is the worst case for the number of revoked users $r = N/2$).

	# of User keys	Message Length (worst case)	# of Decryption	Collusion Threshold
Trivial	1	$N/2$	1	N
CS[9]	$\log_2 N + 1$	$N/2$	1	N
SD[9]	$\log_2 N(\log_2 N + 1)/2 + 1$	$N/2$	1	N
LSD[6]	$\log_2^{3/2} N + 1$	$N/2$	1	N
PS[8]	$(2^{(a-1)} - 1) \log_a N + 1$	N/a	1	N
The subset scheme	$\lceil (2^n - 1)/n \rceil$	N/n	1	N

based on a pseudo random generator which is a cryptographic primitive equivalent to private key encryption. The result of them is as follows: Let n be the total number of users. The number of keys given to each user is at most $\lceil (2^n - 1)/n \rceil$ and the required computational overhead is $O(n)$.

Message Generation for Revocation. For a given set $\boldsymbol{R}$ of revoked users, let $\boldsymbol{R}$ be divided into $\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_{N/n}$, where $\boldsymbol{R}_i$ is a group of revoked users of $\boldsymbol{N}_i$. Each set $\boldsymbol{N}_i \setminus \boldsymbol{R}_i$ is associated with at least one key. Therefore, the session key is encrypted with at most N/n keys. In the example depicted in Figure 3, the center generates the message with the key $L_1[1], L_2[2]$ and $L_3[4]$ to revoke $\boldsymbol{R}_1 = \{u_2, u_3\}$, $\boldsymbol{R}_2 = \{u_4, u_6\}$ and $\boldsymbol{R}_3 = \{u_9\}$, respectively.

Message Decryption. When receiving the broadcasted message, each user $u \in \boldsymbol{N}$ finds whether subset including it is among indices. If a table lookup structure is used, the evaluation of the subset key takes one. Therefore, the required computational overhead is $O(1)$ and the session key can be retrieved by only one decryption.

4 Comparison

In this section, we compare the performance of Trivial, CS, SD, LDS, PS and the subset scheme where Trivial simply encrypts a session key with each individual

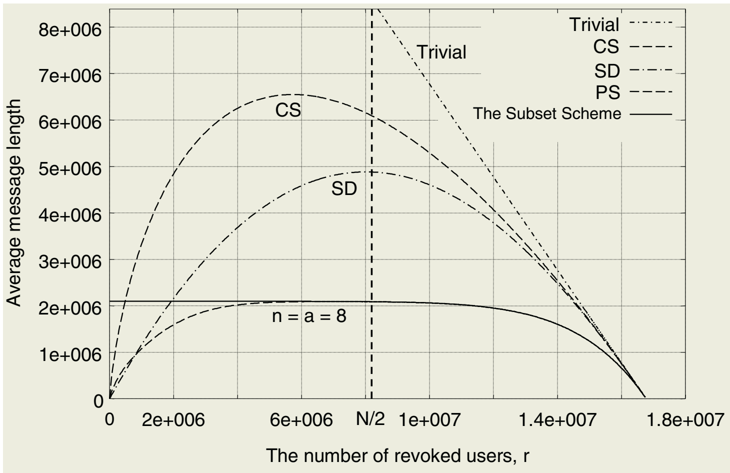


Fig. 4. The average message length (obtained by simulation) vs. r for $N = 2^{24}$ and $a = n = 8$.

key and lists up all the encrypted data. We assume the size of the session-key K and the block size of the symmetric block cipher $E_K()$ are 128 bits, i.e. 16 bytes, respectively. We use the term "message length" to denote the number of blocks where a session-key is encrypted using the symmetric block cipher, according to the previous papers. The size of messages = block size × message length.

Table 2 illustrates the number of user keys, the message length of the worst case for the number of revoked users $r = N/2$, the number of decrypting operations and the collusion threshold of the schemes. As shown in Table 2, all of them are collusion-free and require only one decryption operation of a symmetric cipher to retrieve the session key.

The worst message lengths of CS, SD and LSD are $N/2$ for $r = N/2$. Even though the worst message lengths of SD and LSD are shown to be $2r-1$ and $4r-2$ respectively in [9, 6], these are for small r. We show the average message length versus r obtained by simulation in Figure 4. Note that this figure shows how the average message length behaves according to r, our aim is not to compare the message length among them since the corresponding key sizes are different. The key sizes of CS, SD, PS and the subset scheme in Figure 4 are 400 bytes, 4816 bytes, 12208 bytes and 512 bytes, respectively. As you can see, the message length of CS, SD and LSD [5] are almost maximized around $r = N/2$.

The relationship between the message size and the storage size for $r = N/2$ is illustrated in Figure 5. PS and the subset scheme have a variable tradeoff between the message size and the storage size with the parameter a and n, respectively, whereas CS, SD and LSD have a fixed performance. As shown in Figure 5, the sizes of both messages and storage keys of subset scheme for $n = 4$

[5] The message length of LSD is almost the same as SD [6], even though it is slightly larger than that of SD.

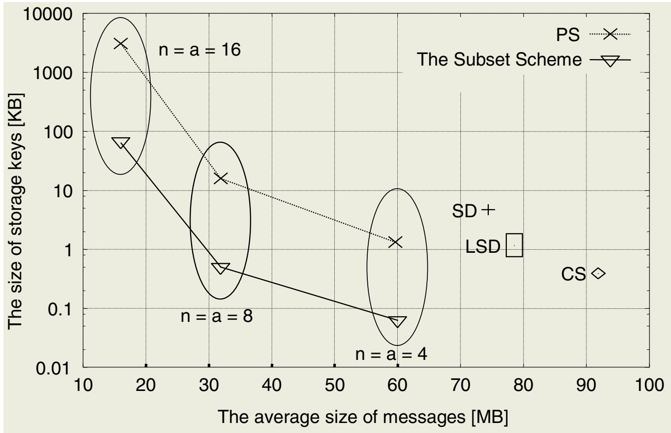


Fig. 5. The tradeoff between the average size of messages (obtained by simulation) and the size of storage keys for $N = 2^{24}$ and $r = N/2$.

is smaller than those of CS, SD and LSD, respectively. Also the size of storage keys of the subset scheme is smaller than that of PS for $n = a$ maintaining the same message size. Thus, the subset scheme has a better performance than the previous schemes on the tradeoff between the size of messages and that of storage keys.

5 Conclusions

In this paper, we have dealt with the problem of how to embed unique fingerprints into broadcasted contents or packaged contents, such as CD and DVD, without giving the watermarking algorithm to the decoders at receivers. We proposed a novel way to fingerprint broadcasted data using around-half-rate dynamic revocation scheme. Our model achieved the following features: (1) No watermarking algorithm at the decoders; (2) Dynamic fingerprinting; and (3) Dynamic revocation.

In addition, we pointed out that when dealing with around-half-rate revocation, most past proposed schemes failed to obtain a good efficiency, i.e. the message length that must be transmitted by the sender or the number of storage keys at a user is too large. We showed that the subset scheme is an efficient algorithm of revocation that reduces both the message length and the size of storage keys at a user while maintaining both collusion-freeness and a single decryption at a user.

Acknowledgment

The authors would like to thank anonymous referees who provided useful comments.

References

1. T. Asano, "A revocation scheme with minimal storage at receivers", In *Advanced in Cryptology - ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 433–450, Springer-Verlag, 2002.
2. N. Attrapadung, K. Kobara, and H. Imai, "Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes", In *Advances in Crytology - ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 374–391, Springer-Verlag, 2003.
3. D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data", *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.
4. A. Fiat and M. Naor, "Broadcast Encryption", In *Advances in Cryptology - CRYPTO'93*, volume 773 of *LNCS*, pages 480–491, Springer-Verlag, 1994.
5. A. Fiat and T. Tassa, "Dynamic traitor tracing", In *Advances in Cryptology - CRYPTO'99*, volume 1666 of *LNCS*, pages 354–371, Springer-Verlag, 1999.
6. D. Halevy and A. Shamir, "The LSD broadcast encryption scheme", In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 47–60, Springer-Verlag, 2002.
7. H. Muratani, "A collusion-secure fingerprinting code reduced by chinese remaindering and its random error resilience", In *Proceedings of Information Hiding Workshop (IHW'01)*, volume 2137 of *LNCS*, pages 315–327, Springer-Verlag, 2001.
8. T. Nakano, M. Ohmori, N. Matsuzaki, and M. Tatebayashi, "Key management system for digital content protection -tree pattern division method-", In *Proceedings of Symposium on Cryptography and Information Security (SCIS 2002)*, volume 2, 2002(in Japanese).
9. D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers", In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62, Springer-Verlag, 2001.
10. R. Safavi-Naini and Y. Wang, "Sequential Traitor Tracing", *IEEE Transactions on Information Theory*, 49(5):1319–1326, 2003.
11. M. Naor and B. Pinkas, "Efficient Trace and Revoke Schemes", In *Proc. of Financial Cryptography '00*, February 2000.
12. M. Yoshida and T. Fujiwara, "An Efficient Traitor Tracing Scheme for Broadcast Encryption", In *Proc. of IEEE International Symposium on Information Theory (ISIT '00)*, June 2000.
13. T. Matsushita, K. Kobara, and H. Imai, "Revocable black-box tracing against self-defensive pirate decoders", In *Proc. of Workshop on Information Security Applications (WISA 2001)*, pp. 335-354, 2001.

Practical Pay-TV Scheme Using Traitor Tracing Scheme for Multiple Channels[*]

Chong Hee Kim[1], Yong Ho Hwang[2], and Pil Joong Lee[2,**]

[1] Samsung Electronics Co., LTD, Korea
chonghee.kim@samsung.com
[2] IS Lab., Dept. of Electronic and Electrical Eng., POSTECH, Korea
yhhwang@oberon.postech.ac.kr, pjl@postech.ac.kr

Abstract. A Pay-TV scheme broadcasts a set of services or streams instead of one. That is, a pay-TV scheme is a specific application of a broadcast encryption scheme in which the secret to be broadcast is associated with a number of services. For example, a pay-TV broadcaster offers various channels such as a sports channel, a movie channel, and so on. A traitor tracing scheme is a variant of a broadcast encryption scheme, so can be applied to construct a pay-TV scheme. However, because most known traitor tracing schemes deal with the broadcast of a single stream, a direct extension to multiple streams is too inefficient, i.e., direct extension to an m-stream case would involve an m-fold increase in the user's secret keys. In other words, if the number of streams to be sent increases, the number of secret keys each user must store in a secure memory also linearly increases. Therefore, we require a specific traitor tracing scheme which can be efficiently applied to a pay-TV scheme. We propose a new traitor tracing scheme for multiple channels and its application to a pay-TV scheme. The number of secret keys each user must store in a secure memory is just one regardless of the number of channels and it cannot be changed. Our scheme has a *revocation* property, i.e., we can revoke some users without redistributing a new secret key to other un-revoked users. Our scheme also provides a so called *holding property* – we can revoke some users and un-revoke them without redistribution of new keys after some period. This is very useful in a pay-TV scheme since a user may wish to un-subscribe from a channel for some periods and re-subscribe again later. Moreover, our pay-TV scheme is based on a *public key* traitor tracing scheme. Therefore, any content provider can send encrypted data to users with the public key the system manager provides. Our scheme can also be extended to provide *asymmetric* property and be secure against the adaptive chosen ciphertext attack.

[*] This research was supported by University IT Research Center Project and the Brain Korea 21 Project.
[**] On leave at KT Research Center, Korea.

1 Introduction

A broadcast encryption scheme is a multiple user encryption scheme, where a sender broadcasts encrypted data to users through a public inseucre channel and only legitimate users can decrypt it. A broadcast encryption scheme has numerous applications, such as a pay-TV system, the distribution of copyrighted materials, internet multicasting of video, music, and magazines, and so on.

Fiat and Naor first formalized the basic definitions and paradigms of the broadcast encryption scheme [10]. Afterwards, many variants have been researched. One is the scheme of tracing traitors. A traitor tracing scheme (TTS) deters traitors from giving away their secret keys to decrypt the transmitted data by enabling the system manager to trace at least one of the traitors who participated in the construction of a pirate decoder after confiscating it. We can divide TTS into two categories. One scheme uses a secret key and coding approach [3, 4, 11, 19, 22–24] and the other uses a public key [2, 12, 25, 15]. In the secret key scheme, the keys in a pirate decoder can be identified by combinatorial methods. In the public key approach, the size of the enabling block is independent of the number of users. In addition, the public key TTS enables the system manager to prepare a public key that allows all content providers to broadcast data to users. A TTS is called *t-resilient* if we can find at least one traitor after confiscation of a pirate decoder which is constructed by at most t users.

In a (*symmetric*) TTS, users share all secret keys with the system manager. Therefore, non-repudiation cannot be offered. That is, a malicious system manager can implicate an innocent user in the construction of a pirate decoder, and users can always deny their implication in the construction of a pirate decoder and claim that it was the work of a malicious system manager. A solution to this problem is an *asymmetric* TTS, first introduced by Pfitzmann and Waidner [22, 23]. In an *asymmetric* TTS, the system manager does not know the entire secret key of the user.

An *asymmetric public key* TTS was presented in [16, 27]. These schemes used an *oblivious polynomial evaluation* (OPE) protocol [20] to achieve asymmetric property. Recently, Kiayias and Yung showed a weakness of [16, 27] and proposed a new scheme [14]. They used the idea of the use of the OPE protocol as in [16, 27], especially the malleable OPE protocol [14], to achieve an asymmetric property and the idea of [2] to construct a tracing algorithm

A Pay-TV scheme broadcasts a set of services or streams instead of one. That is, a pay-TV scheme is a specific application of a broadcast encryption scheme in which the secret to be broadcast is associated with a number of services. For example, a pay-TV broadcaster offers various channels such as a sports channel, a movie channel, and so on. A traitor tracing scheme is a variant of a broadcast encryption scheme, so can be applied to construct a pay-TV scheme. However, because most known traitor tracing schemes deal with the broadcast of a single stream, direct extension to multiple streams is too inefficient, i.e., direct extension to an m-stream case would involve an m-fold increase in the user's secret keys [21]. In other words, if the number of streams to be sent increases, the number of secret keys each user must store in a secure memory also linearly increase. Therefore, we require a specific traitor tracing scheme that can be efficiently applied to a pay-TV scheme.

Recently, Narayanan *et al.* proposed a practical pay-TV scheme [21] that deals with broadcasting multiple streams. In [21], the number of secret keys each user must store in a secure memory is independent of the number of streams to be sent. The number of secret keys of each user must be stored in a secure memory is $t + 4$, and where t is the maximum number of colluded users. We note that each user also has to store additional channel keys for each channel and he/she does not need to store these channel keys in a secure memory [21]. Therefore the user's key consists of $t + 4$ secret keys and m non-secure channel keys if the user subscribes m channels. Narayanan *et al.*'s scheme does not offer *revocation* property and *asymmetric* property. Every time a user un-subscribes from channel j, then the channel key for the channel j must be changed and all other users who subscribe to the channel j must be received a new key.

Our Results. We propose a new traitor tracing scheme for multiple channels and its application to a pay-TV scheme. The number of secret keys each user must store in a secure memory is merely one regardless of the number of channels. Our scheme has a *revocation* property, i.e., we can revoke some users without redistributing a new secret key to other un-revoked users. Our scheme also provides so called *holding property* - we can revoke some users and un-revoke them without redistribution of new keys after some period. This is very useful in a pay-TV scheme since a user may wish to un-subscribe from a channel for a period of time and re-subscribe again later. Moreover, our pay-TV scheme is based on a *public key* traitor tracing scheme. Therefore, any contents provider can send encrypted data to users with the public key the system manager provides. Our scheme can also be extended to provide *asymmetric* property and be secure against the adaptive chosen ciphertext attack. Table 1 shows a simple comparison between Narayanan *et al.*'s scheme and ours.

Table 1. Comparison between Narayanan *et al.*'s and ours.

	# of secret keys	Revocation	Holding	System type
[21]	$t + 4$	NO	NO	Symmetric key TTS
Our scheme	1	YES	YES	Public key TTS

2 Preliminaries

In this section, we briefly review the Lagrange interpolation in the exponent and a malleable oblivious polynomial evaluation.

The Lagrange Interpolation in the Exponent. Let q be a prime and $f(x)=\sum_{t=0}^{z} a_t x_t$ a polynomial of degree z over Z_q. Let $x_0, \ldots, x_z$ be distinct elements in Z_q. Then using the Lagrange interpolation, we can express $f(x)$

as $\sum_{t=0}^{z}(f(x_t)\cdot\lambda_t(x))$, where $\lambda_t(x) = \prod_{0\le j\ne t\le z}\frac{x_j - x}{x_j - x_t}$, $0 \le t \le z$. We define the Lagrange interpolation operator as: $LI(x_0,\ldots,x_z; f(x_0),\ldots,f(x_z))(x) = \sum_{t=0}^{z}(f(x_t)\cdot\lambda_t(x))$.

Next, we consider a cyclic group G of order q and a generator g of G. Let $v_t = g^{f(x_t)}$, $0 \le t \le z$, where $x_t \in Z_q$ and $v_t \in G$. Then we define the Lagrange interpolation operator in the exponent as: $EXP{-}LI(x_0,\ldots,x_z; v_0,\ldots,v_z)(x) = g^{LI(x_0,\ldots,x_z; f(x_0),\ldots,f(x_z))} = \prod_{t=0}^{z} g^{(f(x_t)\cdot\lambda_t(x))} = \prod_{t=0}^{z} v_t^{\lambda_t(x)}$. We also remark that $EXP - LI(x_0,\ldots,x_z; v_0^r,\ldots,v_z^r)(x) = [EXP - LI(x_0,\ldots,x_z; v_0,\ldots,v_z)(x)]^r$. In what follows, we will refer to a function of the form $g^{f(x)}$, where $f(x)$ is polynomial, as an EXP- polynomial.

A Malleable Oblivious Polynomial Evaluation. An OPE protocol involves two parties, the sender S who possesses a secret polynomial $P \in Z_q[x]$, and the receiver R who possesses a secret value $\alpha \in Z_q$. An OPE protocol allows R to compute $P(\alpha)$ in such a way that:

– S cannot extract any non-trivial information about α.
– R cannot extract any information about the polynomial P, other than what can trivially be extracted from $P(\alpha)$.

We assume a two communication flow protocol (such as [5, 20]) where $\{\text{OPE}\}(\alpha)$ denotes the data transmitted by the receiver R to the sender S in the first flow, and $\{\text{OPE}\}(\text{P}(\alpha))$ denotes the data transmitted by the sender to the receiver in the second communication flow. If an OPE protocol has the following two additional properties, we call it a malleable OPE [14]:

– Given $\{\text{OPE}\}(\alpha)$ the sender can easily compute $\{\text{OPE}\}(\alpha + \alpha')$, for a given (e.g., random) α' and $+$ an operation in the underlying finite field.
– It is performed over a publicly committed value, namely α can be thought of as a private key whose public key is publicly known.

3 Proposed Scheme

In this section, we describe our proposed scheme. We assume that there are three entities, a system manager, content providers, and users. The system manager generates system parameters, the public key, and the master secret key. The system manager traces the traitors after the confiscation of a pirate decoder. Given the public key, content providers send encrypted data to users. A content provider can provide several channels. Let t be the maximum number of colluded users and z be the revocation threshold. We set $z \ge 2t - 1$.

System Initialization: The system manager selects a random generator $g \in G$, where G is a group of order q in which q is a large prime such that $2q = p - 1$, and p is a large prime. It selects a random z degree polynomial $f(x) = a_0 + a_1x + \ldots + a_zx^z$ over Z_q. Then it publishes the public key $PK{=}(g, g^{a_0}, g^{f(1)}, \ldots, g^{f(z)})$. The master secret key is $f(x)$.

User Registration: The system manager generates a new index $i > z$ and a random value $\alpha_i \in Z_q$. The user's secret key is (i, α_i). Note that the user does not need to store i in a secure memory, only α_i is the secret value for the user.

Channel Registration: To register a new channel, the content provider selects a random $b_j \in Z_q$. Then the content provider securely sends b_j to the system manager. We note that b_j must be kept secret.

Subscription: To subscribe to the channel j, a user i has to receive a channel key, K_i^j, from the system manager. The system manager generates $Q_j(x, y) = f(x) + b_j y$ and computes $Q_j(i, \alpha_i) = f(i) + b_j \alpha_i$. Then the channel key for a user i to channel j is $K_i^j = Q_j(i, \alpha_i)$. The system manger sends K_i^j to the user. Note that the user does not need to store the channel key, K_i^j, in a secure memory.

We note that when a user first subscribes to a channel, user registration and subscription processes are performed sequentially. Afterwards, when a user subscribes to another channel, only subscription process is required. That is, user registration process occurs with subscription process when a user first subscribes to a channel.

Unsubscription: Let C_j be the set of the found traitors or the revoked users for the channel j. Suppose that a user i wants to un-subscribe from the channel j, the content provider inserts the index i to the revoke set C_j. Then the content provider does *encryption with revocation*(See below for the detail). Note that our scheme does not require redistribution of a new channel key to other users who subscribe to channel j.

Encryption Without Revocation: Suppose that the content provider wishes to send a message M_j of the channel j to users. Given the public key PK=(g, h_0, h_1, …, h_z)=(g, g^{a_0}, $g^{f(1)}$, …, $g^{f(z)}$), the content provider randomly selects a session key s_j and encrypts M_j with s_j (generally this process is done by symmetric encryption). Then it makes an enabling block T which contains s_j as follows:

The content provider selects a random $r \in Z_q$ and $j_1, \ldots, j_z \in Z_q$ and computes

$$T =< g^r, s_j h_0^r, g^{-b_j r}, (j_1, g^{rf(j_1)}), (j_2, g^{rf(j_2)}), \ldots, (j_z, g^{rf(j_z)}) > .$$

Where $j_1, \ldots, j_z$ are z unused shares which are not assigned to any user. The content provider can compute $g^{rf(j_t)} = EXP{-}LI(0, \ldots, z, h_0^r, h_1^r, \ldots, h_z^r)(j_t)$ using the Lagrange interpolation in the exponent.

Encryption with Revocation: Suppose that $C_j = \{c_1, \ldots, c_m\}$, $m \leq z$, is the set of the found traitors or the revoked users for the channel j. To send a message M_j to the remaining users, instead of randomly choosing unused shares, the content provider fixes the first m shares as

$$(c_1, g^{rf(c_1)}), (c_2, g^{rf(c_2)}), \ldots, (c_m, g^{rf(c_m)})$$

and randomly chooses the rest $z-m$ unused shares

$$(j_1, g^{rf(j_1)}), (j_2, g^{rf(j_2)}), \ldots, (j_{z-m}, g^{rf(j_{z-m})}).$$

The remaining process is the same as *encryption without revocation.*

Decryption: Given the enabling block $T=< H, H_0, H', (j_1, H_{j_1}), (j_2, H_{j_2}), \ldots, (j_z, H_{j_z}) > =< g^r, s_j h_0^r, g^{-b_j r}, (j_1, g^{rf(j_1)}), (j_2, g^{rf(j_2)}), \ldots, (j_z, g^{rf(j_z)}) >$ and the user's key $< i, \alpha_i, K_i^j >$, the user i can compute s_j with the equation

$$s_j \leftarrow \frac{H_0}{EXP-LI(j_1,\ldots,j_z,i;H_{j_1},\ldots,H_{j_z},H^{K_i^j}H'^{\alpha_i})(0)}$$

Before proceeding any further, we briefly show that the output of the above decryption algorithm is identical to the session key s_j as follows:

$$\begin{aligned}
&\frac{H_0}{EXP-LI(j_1,\ldots,j_z,i;H_{j_1},\ldots,H_{j_z},H^{K_i^j}H'^{\alpha_i})(0)}\\
&= \frac{s_j h_0^r}{EXP-LI(j_1,\ldots,j_z,i;H_{j_1},\ldots,H_{j_z},(g^r)^{K_i^j}(g^{-b_j r})^{\alpha_i})(0)}\\
&= \frac{s_j h_0^r}{EXP-LI(j_1,\ldots,j_z,i;H_{j_1},\ldots,H_{j_z},(g^r)^{f(i)+b_j\alpha_i}(g^{-b_j r})^{\alpha_i})(0)}\\
&= \frac{s_j h_0^r}{EXP-LI(j_1,\ldots,j_z,i;H_{j_1},\ldots,H_{j_z},(g^r)^{f(i)})(0)}\\
&= \frac{s_j h_0^r}{EXP-LI(j_1,\ldots,j_z,i;H_{j_1},\ldots,H_{j_z},H_i)(0)}\\
&= \frac{s_j h_0^r}{h_0^r}\\
&= s_j
\end{aligned}$$

If the user i is included in C_j, then she cannot extract the session key s_j. That is, since $EXP-LI(j_1, \ldots, j_z, i; H_{j_1}, \ldots, H_{j_z}, H_i)(0)$ does not have independent $z+1$ shares, the user i cannot compute h_0^r.

Tracing: We present a black box traitor tracing algorithm. This algorithm is based on "black-box confirmation" [2, 25].

For every possible m-coalition $\{c_1, c_2, \ldots, c_m\}$ of the users, $m \le t$,

1. Randomly selects $z-m$ unused shares, say, $\{j_1, \ldots, j_{z-m}\}$.
2. Construct a testing enabling block: $T=<g^r, s_j h_0^r, g^{-b_j r}, (c_1, g^{rf(c_1)}), (c_2, g^{rf(c_2)}), \ldots, (c_m, g^{rf(c_m)}), (j_1, g^{rf(j_1)}), (j_2, g^{rf(j_2)}), \ldots, (j_{z-m}, g^{rf(j_{z-m})})>$.
3. Feed $< T, Enc(M_j, s_j) >$ to the decoder. Where, $Enc(M_j, s_j)$ is the symmetric encryption of M_j with s_j.
4. If the decoder does not output the correct data M_j, we set $\{c_1, c_2, \ldots, c_m\}$ as a possible set of traitors.

Output the smallest of all possible sets of traitors found in Step 4.

Re-subscription: Suppose that the user i unsubscribes from the channel j. Then the content provider use *encryption with revocation* algorithm to send encrypted data to the users and C_j contains i. If user i wants to re-subscribe to the channel j after some period, we can do this very simply. The content provider subtracts i from C_j. That is, it can be done by not using i to form the enabling block.

Revocation Beyond the Threshold: If we have to revoke more than z users, we can use the idea of [1]. The idea is that if a pirate decoder can decrypt at most c percent of the transmitted data, say 95%, the partial part of the data is useless. For example, if the content provider sends a movie and a pirate decoder decrypt 95% of the movie, nobody uses this pirate decoder.

Assume that $C_j = \{c_1, \dots, c_m\}$, $m > z$, is the set of the found traitors or the revoked users for the channel j. To broadcast message M_j, we partition M_j as $M_j^1||M_j^2||\dots||M_j^l$. For each M_j^k, $1 \leq k \leq l$, the content provider constructs an enabling block T_j^k with shares:

$$(c_{i_1}, g^{r_k f(c_{i_1})}), (c_{i_2}, g^{r_k f(c_{i_2})}), \dots, (c_{i_z}, g^{r_k f(c_{i_z})}),$$

where c_{i_1}, …, c_{i_z} are randomly chosen from C_j.

Asymmetric Traitor Tracing Scheme: The above protocol does not provide the asymmetric property. Since the system manger knows all the user's keys, a malicious system manager can implicate an innocent user in the construction of a pirate decoder, and users can always deny their implication in the construction of a pirate decoder and claim that it was the work of a malicious system manager.

Recently, Kiayias and Yung proposed a technique to construct an asymmetric traitor tracing scheme [14] using the malleable oblivious polynomial evaluation (OPE). However, we cannot directly apply this technique to our scheme. Suppose that we use the malleable OPE between the user and the system manager as in [14]. Then the user i can compute $Q_j(i, \alpha_i){=}f(i) + b_j\alpha_i$ so that: i is randomly selected by the system manager and $\alpha_i{=}\alpha_i^C + \alpha_i^R$, where α_i^C is a value selected by the user and the value α_i^R is randomly selected by the system manager. The commitment of the user to the value α_i^C will be of the form $< g^{\alpha_i^C}, \mathbf{sign}_i(g^{\alpha_i^C}) >$. The system manager should not know the value $Q_j(i, \alpha_i)$ after the OPE protocol and this value must keep secret in the user's memory. However, in our setting $Q_j(i, \alpha_i)$ is *not* a secret value - notice that the user's secret value is only α_i -, so the system manager may know $Q_j(i, \alpha_i)$. Then the system manager can obtain the value α_i because the system manager already knows $f(i)$ and b_j. This breaks the asymmetric property.

In our setting, the system manger constructs $Q_j(x, y, z){=}f(x){+}b_jy{+}z$ and let $K_i^j{=}Q_j(i, \alpha_i^1, \alpha_i^2)$. The malleable OPE between the user and the system manager is done and finally the user i can compute $Q_j(i, \alpha_i^1, \alpha_i^2)$. The system manager cannot extract the user's secret key (α_i^1, α_i^2) which is generated by the user. Although the system manager knows $f(i)$, b_j, and $Q_j(i, \alpha_i^1, \alpha_i^2)$, from these values he cannot solve the equation $Q_j(i, \alpha_i^1, \alpha_i^2){=}f(i) + b_j\alpha_i^1 + \alpha_i^2$. We note that

$Q_j(i, \alpha_i^1, \alpha_i^2)$ is not known to the system manager during the OPE, but it can be extracted from the subscription process or else since the user does not need to store $Q_j(i, \alpha_i^1, \alpha_i^2)$ in a secure memory.

4 Security Analysis

In this section, we analyze the security of our scheme. First, we show that our scheme is secure against z-coalition assuming that the discrete logarithm problem is hard. To decrypt the message of the channel j, both the channel key for the user i, K_i^j, and the secret key, α_i, are required. Therefore, we show that the attacker cannot compute another key pair (both channel key and the secret key) for the channel j from the given z key pairs $((x_1,\ \alpha_1,\ Q_j(x_1, \alpha_1)),\ \ldots,\ (x_z, \alpha_z, Q_j(x_z, \alpha_z))$ for the channel j.

Theorem 1 *No coalition of z or fewer users who subscribe the channel j can compute the private key of another user with a non-negligible probability, assuming that computing the discrete logarithm over G_q is hard.*

Proof. Suppose that the probabilistic polynomial-time algorithm $\mathcal{A}$ can compute a new secret key $(x_k,\ \alpha_k,\ Q_j(x_k, \alpha_k))$ from the given public key $PK{=}(g,\ g^{a_0},\ g^{f(1)},\ \ldots,\ g^{f(z)})$ and z shares $(x_1,\ \alpha_1,\ Q_j(x_1, \alpha_1)),\ \ldots,\ (x_z,\ \alpha_z,\ Q_j(x_z, \alpha_z))$ with a non-negligible probability ϵ. Then we can construct another probabilistic polynomial-time algorithm $\mathcal{B}$ to compute the discrete logarithm over G_q with an overwhelming probability.

Let (p, q, y) be the input of the discrete logarithm problem. The following $\mathcal{B}'$ computes $\log_g y \pmod p$ with a non-negligible probability. Let $y = g^{a_0}$ and $f(x)$ be the z-degree polynomial passing $(0, a_0)$ and $(x_i, f(x_i))$, $1 \le i \le z$. We feed the public key $PK{=}(g,\ g^{a_0},\ g^{f(1)},\ \ldots,\ g^{f(z)})$ and z shares $(x_1,\ \alpha_1,\ Q_j(x_1, \alpha_1)),\ \ldots,\ (x_z,\ \alpha_z,\ Q_j(x_z, \alpha_z))$ to $\mathcal{A}$ and shall obtain a new share $(x_k,\ \alpha_k,\ Q_j(x_k, \alpha_k))$ with a non-negligible probability. From $Q_j(x, y){=}f(x) + b_j y$, we can compute $f(x_k)$ (here we again suppose that b_j is given to the attacker). With the given z shares and $(x_k,\ f(x_k))$, we can compute $f(0){=}a_0$.

By applying the randomized technique to $\mathcal{B}'$ for a polynomial number of tests, we obtain $\mathcal{B}$. □

The next theorem assures us that z or fewer users cannot compute the secret information b_j of the channel j.

Theorem 2 *No coalition of z or fewer users who subscribe the channel j can compute the private information b_j of the channel j with a non-negligible probability.*

Proof. Suppose that z users who subscribe the channel j put their secret keys and want to compute the private information b_j of the channel j. Then we can compute the following equation:

$$\begin{pmatrix} Q_j(x_1,\alpha_1) \\ Q_j(x_2,\alpha_2) \\ \vdots \\ Q_j(x_z,\alpha_z) \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^z & \alpha_1 \\ 1 & x_2 & x_2^2 & \cdots & x_2^z & \alpha_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_z & x_z^2 & \cdots & x_z^z & \alpha_2 \end{pmatrix}}_{M} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_z \\ b_j \end{pmatrix}$$

To solve the above equation, we require $z+2$ shares, but the matrix M has the rank z. Therefore, z coalition of users cannot compute the secret information b_j of the channel j. □

4.1 Security Against the Adaptive Chosen Ciphertext Attack

By the technique of [7, 13], we modify our scheme so that it becomes secure against the adaptive chosen ciphertext attack under the decision Diffie-Hellman assumption and the collision-resistant hash function assumption.

System Initialization and Channel Registration: The system manager selects two random generators $g_1, g_2 \in G$, where G is a group of order q in which q is a large prime such that $2q = p - 1$, p is a large prime. It selects x_1, x_2, y_1, $y_2 \in Z_q$ and z-degree polynomials $X_1(\xi)$, $X_2(\xi)$, $Y_1(\xi)$, $Y_2(\xi)$ over Z_q such that $X_1(0) = x_1$, $X_2(0) = x_2$, $Y_1(0) = y_1$, $Y_2(0) = y_2$. It also selects z-degree polynomials $Z_1(\xi)$, $Z_2(\xi)$ over Z_q and computes $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$. Then it computes $h_t{=}g_1^{Z_1(t)} g_2^{Z_2(t)}$, $0 \le t \le z$ and $x_{1,t}{=}g_1^{X_1(t)}$, $x_{2,t}{=}g_2^{X_2(t)}$, $y_{1,t}{=}g_1^{Y_1(t)}$, $y_{2,t}{=}g_2^{Y_2(t)}$. The content provider selects a random $b_j{\in}\, Z_q$ and give it to the system manager. The system manager generates $Q_j^1(x){=}Z_1(x)$ and $Q_j^2(x,y){=}Z_2(x) + b_j y$.

Finally the system manager chooses a hash function H from a family of F of collision resistant hash functions, and outputs (PK, SK_{BE}), where $PK{=}(p$, q, g_1, g_2, c, d, $x_{1,0}$, …, $x_{1,z}$, $x_{2,0}$, …, $x_{2,z}$, $y_{1,0}$, …, $y_{1,z}$, $y_{2,0}$, …, $y_{2,z}$, h_0, …, h_z, $H)$ and $SK_{BE}{=}(X_1$, X_2, Y_1, Y_2, Z_1, $Z_2)$.

User Registration and Subscription: Each time a new user $i > z$ decides to subscribe to the channel j, the system manager provides him with a decoder box containing the secret key $SK_i{=}(i$, α_i, $X_1(i)$, $X_2(i)$, $Y_1(i)$, $Y_2(i)$, $Q_j^1(i)$, $Q_j^2(i,\alpha_i))$. If a user already subscribes to other channel, only $Q_j^1(i)$ and $Q_j^2(i,\alpha_i)$ is sent to him.

The Encryption and Decryption: The encryption and decryption algorithm can be summarized as shown in Table 2.

We here verify that the output of the decryption algorithm is identical to session key s if the user i is a legitimate user. We can rewrite F_i computed from the Step D_4 as follows (let $g_2{=}g_1^w$):

Table 2. Encryption and decryption scheme secure against CCA2.

Encryption algorithm $\mathbf{Enc}(PK, s, T)$	Decryption algorithm $\mathbf{Dec}(i, T)$
$E_1.\ r_1 \leftarrow_r Z_q$	$D_1.\ \alpha \leftarrow H(S, u_1, u_2, u_3)$
$E_2.\ u_1 \leftarrow g_1^{r_1}$	$D_2.\ C_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$
$E_3.\ u_2 \leftarrow g_2^{r_1}$	$D_3.\ H_i \leftarrow u_1^{Q_j^1(i)} \cdot u_2^{Q_j^2(i,\alpha_i)} \cdot u_3^{\alpha_i}$
$E_4.\ u_3 \leftarrow g_2^{-b_j r_1}$	$D_4.\ F_i \leftarrow H_i \frac{C}{C_i}$
$E_5.\ H_t \leftarrow h_t^{r_1}, (t = 0,.., z)$	$D_5.\ s \leftarrow \frac{S}{EXP\text{-}LI(j_1,..,j_z,i;F_{j_1},..,F_{j_z},F_i)(0)}$
$E_6.\ H_{j_t} \leftarrow EXP\text{-}LI(0,.., z; H_0,.., H_z)(j_t)$ $(t = 1,.., z)$	
$E_7.\ S \leftarrow s_j \cdot H_0$	
$E_8.\ \alpha \leftarrow H(S, u_1, u_2, u_3)$	
$E_9.\ C_t \leftarrow (x_{1,t}x_{2,t})^{r_1}(y_{1,t}y_{2,t})^{r_1\alpha}$ $(t = 0,.., z)$	
$E_{10}.\ C_{j_t} \leftarrow EXP\text{-}LI(0,.., z; C_0,.., C_z)(j_t),$ $(t = 1,.., z)$	
$E_{11}.C \leftarrow c^{r_1} d^{r_1\alpha}$	
$E_{12}.F_{j_t} = H_{j_t} \frac{C}{C_{j_t}}, (t = 1,.., z)$	
$E_{13}.T \leftarrow< S, u_1, u_2, u_3, c^{r_1} d^{r_1\alpha},$ $(j_1, F_{j_1}),.., (j_z, F_{j_z}) >$	

$$
\begin{aligned}
&F_i = H_i \cdot (\tfrac{C}{C_i}) \\
&= (u_1^{Q_j^1(i)} u_2^{Q_j^2(i,\alpha_i)} u_3^{\alpha_i})(c^{r_1} d^{r_1\alpha})(u_1^{-X_1(i)-Y_1(i)\alpha} \cdot u_2^{-X_2(i)-Y_2(i)\alpha}) \\
&= g_1^{r_1 Z_1(i)+wr_1 Z_2(i)+wr_1 b_j \alpha_i - wr_1 b_j \alpha_i - r_1 X_1(i) - r_1 Y_1(i)\alpha - wr_1 X_2(i) - wr_1 Y_2(i)\alpha} c^{r_1} d^{r_1\alpha} \\
&= g_1^{r_1 Z_1(i)+wr_1 Z_2(i)-r_1 X_1(i)-r_1 Y_1(i)\alpha - wr_1 X_2(i) - wr_1 Y_2(i)\alpha + (r_1 x_1 + wr_1 x_2 + r_1 y_1 \alpha + wr_1 y_2 \alpha)} \\
&= g_1^{Q(i)}
\end{aligned}
$$

Consequently, $F_i{=}g_1^{Q(i)}$ where, $Q(\xi)$ is z-degree polynomial in Z_q. If we compute F_0 using the *Lagrange interpolation in the exponent* like Step D_5, finally we can get following value:

$$
\begin{aligned}
F_0 =& EXP - LI(j_1, \cdots, j_z, i; F_{j_1}, \ldots, F_{j_z}, F_i)(0) \\
=& g_1^{(r_1 z_1 + wr_1 z_2) - r_1 x_1 - r_1 y_1 \alpha - wr_1 x_2 - wr_1 y_2 \alpha + (r_1 x_1 + wr_1 x_2 + r_1 y_1 \alpha + wr_1 y_2 \alpha)} \\
=& H_0 \frac{c^{r_1} d^{r_1\alpha}}{C} \\
=& H_0
\end{aligned}
$$

Therefore, $\frac{S}{F_0} = \frac{(s \cdot H_0)}{H_0} = s_j$.

Theorem 3 *If the DDH problem is hard in G_q and H is chosen from a collision resistant hash function family F, then our scheme is z-resilient against the adaptive chosen ciphertext attack.*

The proof of Theorem 3 is shown in Appendix A.

5 Conclusions

We proposed a new traitor tracing scheme for multiple channels and its application to a pay-TV scheme. Our scheme requires only one secret key for each user and it cannot be changed. The *revocation* property and *holding* property of our scheme is very useful to construct a practical pay-TV scheme. The *revocation* property enables the content providers to revoke some users without redistribution of a new secret key to other un-revoked users. The *holding* property enables the content provider to revoke a user and un-revoke her without redistribution of a new key to her after some period. Moreover, our pay-TV scheme is based on a *public key* traitor tracing scheme. Therefore, any contents provider can send encrypted data to users with the public key the system manager provides. Our scheme also can be extended to provide *asymmetric* property and be secure against the adaptive chosen ciphertext attack.

References

1. M. Abdalla, Y. Shavitt, and A. Wool, Towards making broadcast encryption practice, *FC'99, LNCS V.1648*, pp.140-157, 1999.
2. D. Boneh and M. Franklin, An efficient public key traitor tracing scheme, *CRYTO'99, LNCS V.1666*, pp.338-353, 1999.
3. D. Boneh and J. Shaw, Collusion-secure fingerprinting for digital data, *IEEE Transaction on Information Theory 44(5)*, pp.1897-1905, 1998.
4. B. Chor, A. Fiat, andd M. Naor, Tracing traitor, *CRYPTO'94, LNCS V.839*, pp.257-270, 1994.
5. Y.C. Chang and C.J. Lu, Oblivious polynomial evaluation and oblivious neural learning, *ASIACRYPT'99, LNCS V.2248*, pp.369-384, 2001.
6. R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *CRYPTO'98, LNCS V.1462*, pp.13-25, 1998.
7. R. Cramer and V. Shoup, Design and analysis of practical public key encryption scheme secure against adaptive chosen ciphertext attack, *Manuscript*, 2001.
8. Y. Dodis and N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, *PKC'03*, pp.100-115, 2003.
9. Y. Dodis and N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, Full version of [8], Availabe at http://eprint.iacr.org/, 2002.
10. A. Fiat and M. Naor, Broadcast encryption, *CRYPTO'93, LNCS V.773* pp.480-491, 1993.
11. E. Gafni, J. Staddon, and Y.L. Yin, Efficient methods for integrating traceability and broadcast encryption, *CRYPTO'99, LNCS V.1666*, pp.372-287, 1999.
12. K. Kurosawa and Y. Desmedt, Optimum traitor tracing and asymmetric schemes, *EUROCRYPT'98, LNCS V.1403*, pp.145-157, 1998.
13. C.H. Kim, Y.H. Hwang, and P.J. Lee, An efficient public key trace and revoke scheme secure against adaptive chosen ciphertext attack, *ASIACRYPT'03, LNCS V.2893*, pp.359-373, 2003.
14. A. Kiayias and M. Yung, Breaking and repairing asymmetric public-key traitor tracing, *ACM workshop on digital rights management, LNCS V.2696*, 2002. available at http://www.cse.uconn.edu/ akiayias/pubs/asymvpp-f.pdf

15. A. Kiayias and M. Yung, Traitor tracing with constant transmission rate, *EUROCRYPT'02, LNCS V. 2332*, pp.450-465, 2002.
16. H. Komaki, Y. Watanabe, G. Hanaoka, and H. Imai, Efficient asymmetric self-enforcement scheme with public traceability, *PKC'01, LNCS V.1992*, pp.225-239, 2001.
17. R. Canetti and S. Goldwasser, An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack, *EUROCRYPT'99, LNCS V.1592*, pp.90-106, 1999.
18. Y. Mu and V. Varadharajan, Robust and secure broadcast, *Indocrypt'01, LNCS V.2247*, pp.223-231, 2001
19. M. Naor and B. Pinkas, Threshold traitor tracing, *CRYPTO'98, LNCS V.1462*, pp.502-517, 1998.
20. M. Naor and B. Pinkas, Oblivious transfer and polynomial evaluation, *STOC'99*, pp.245-254, 1999.
21. A. Narayanan, C.P. Rangan, and K. Kim, Practical pay TV schemes, *ACISP'03, LNCS V.2727*, pp.192-203, 2003.
22. B. Pfitzmann, Trials of traced traitors, *Workshop on Information Hiding, LNCS V.1174*, pp.49-64, 1996.
23. B. Pfitzmann and M. Waidner, Asymmetric fingerprinting for large collusions, *ACM conference on Computer and Communication Security*, pp.151-160, 1997.
24. D.R. Stinson and R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Math 11(1)*, pp.41-53, 1998.
25. W.G. Tzeng and Z.J. Tzeng, A public-key tracing scheme with revocation using dynamic shares, *PKC'01, LNCS 1992*, pp.207-224, 2001.
26. A. Wool, Key management for encrypted broadcast, *5th ACM conference on Computer and Communications Security*, pp.7-16, 1998.
27. Y. Watanabe, G. Hanaoka, and H. Imai, Efficient asymmetric public-key traitor tracing without trusted agents, *CT-RSA'01, LNCS V.2020*, pp.392-407, 2001.

Appendix A. Proof of Theorem 3

Our overall strategy for the proof follows the structural approach in [7]. We shall define a sequence $\mathbf{G_0}, \mathbf{G_1}, \ldots, \mathbf{G_l}$ of modified attack games. Each of the games $\mathbf{G_0}, \mathbf{G_1}, \ldots, \mathbf{G_l}$ operates on the same underlying probability space. In particular, the public key cryptosystem, the coin tosses **Coins** of $\mathcal{A}$, and the hidden bit σ take on identical values across all games. Only some of the rules defining how the environment responds to oracle queries differ from game to game. For any $1 \le i \le l$, we let T_i be the event that $\sigma = \sigma^*$ in the game $\mathbf{G_i}$. Our strategy is to show that for $1 \le i \le l$, the quantity $|Pr[T_{i-1}] - Pr[T_i]|$ is negligible. Also, it will be evident from the definition of game $\mathbf{G_l}$ that $Pr[T_l] = \frac{1}{2}$, which will imply that $|Pr[T_0] - \frac{1}{2}|$ is negligible.

Before continuing, we state the following simple but useful lemma [7].

Lemma 1 *Let U_1, U_2, and F be the events defined on some probability space. Suppose that the event $U_1 \wedge \neg F$ occurs if and only if $U_2 \wedge \neg F$ occurs. Then $|Pr[U_1] - Pr[U_2]| \le Pr[F]$.*

Game $\mathbf{G_0}$: Let $\mathbf{G_0}$ be the original attack game, let $\sigma^* \in \{0,1\}$ denote the output of $\mathcal{A}$, and let T_0 be the event that $\sigma = \sigma^*$ in $\mathbf{G_0}$, so that $Adv^{CCA2}_{Ourscheme,\mathcal{A}}(\lambda) = |Pr[T_0] - \frac{1}{2}|$.

Game $\mathbf{G_1}$: $\mathbf{G_1}$ is identical to $\mathbf{G_0}$, except that in $\mathbf{G_1}$, steps E_5 and E_9 are replaced with the following:

$$E_5'.\ H_t \leftarrow u_1^{Q_j^1(t)} \cdot u_2^{Q_j^2(t,\alpha_t)} \cdot u_3^{\alpha_t},\ t = 0, \ldots, z$$
$$E_9'.\ C_t \leftarrow u_1^{X_1(t)+Y_1(t)\alpha} \cdot u_2^{X_2(t)+Y_2(t)\alpha},\ t = 0, \ldots, z$$

The change we have made is purely conceptual, it merely makes explicit any functional dependency of the above quantities on u_1, u_2, and u_3. Cleary, it holds that $Pr[T_0] = Pr[T_1]$.

Game $\mathbf{G_2}$: We again modify the encryption oracle, replacing steps E_1 and E_3 by

$$E_1'.\ r_1 \leftarrow_r Z_q, r_2 \leftarrow_r Z_q \backslash \{r_1\}$$
$$E_3'.\ u_2 \leftarrow g_2^{r_2}$$

Notice that while in $\mathbf{G_1}$ the values u_1 and u_2 are obtained using the same value r_1, in $\mathbf{G_2}$ they are independent subject to $r_1 \neq r_2$. Therefore, any difference in behavior between $\mathbf{G_1}$ and $\mathbf{G_2}$ immediately yields a PPT algorithm $\mathcal{A}_1$ that is able to distinguish DH tuples from totally random tuples with a non negligible advantage. That is, $|Pr[T_2] - Pr[T_1]| \leq \epsilon_1$ for some negligible ϵ_1.

Game $\mathbf{G_3}$: In this game, we modify the decryption oracle in $\mathbf{G_2}$ to obtain $\mathbf{G_3}$ as follows:

```
$D_1$.      $\alpha \leftarrow H(S, u_1, u_2, u_3)$
$D_2'$.      $C_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha+(X_2(i)+Y_2(i)\alpha)w}$
$D_{2-1}$.  if $(u_2 = u_1^w)$
$D_3'$.      then $H_i \leftarrow u_1^{Q_j^1(i)+Q_j^2(i,\alpha_i)w-b_j r_1 \alpha_i w}$
$D_4'$.              $F_i \leftarrow H_i \frac{C}{C_i}$
$D_5'$.              $s \leftarrow \frac{S}{EXP\text{-}LI(j_1,\ldots,j_z,i,F_{j_1},\ldots,F_{j_z},F_i)(0)}$
$D_6'$.      else return $\perp$
```

Now, let R_3 be the event that the adversary $\mathcal{A}$ submits some decryption queries that are rejected in Step D_{2-1} in $\mathbf{G_3}$, but has passed in $\mathbf{G_2}$. Note that if a query passes in D_{2-1} in $\mathbf{G_3}$, it would have also passed in $\mathbf{G_2}$. It is clear that $\mathbf{G_2}$ and $\mathbf{G_3}$ proceed identically until the event R_3 occurs. In particular, the event $T_2 \wedge \neg R_3$ and $T_3 \wedge \neg R_3$ are identical. Therefore, by Lemma 1, we have

$$|Pr[T_3] - Pr[T_2]| \leq Pr[R_3]$$

and so it suffices to bound $Pr[R_3]$. To do this we consider two more games, $\mathbf{G_4}$ and $\mathbf{G_5}$

Game $\mathbf{G_4}$: This game is identical to $\mathbf{G_3}$, except for a change in Step E_7 as follows:

$$E_7'.e \leftarrow_r Z_q, S \leftarrow g_1^e$$

It is clear by construction that $Pr[T_4] = \frac{1}{2}$, since in $\mathbf{G_4}$, the variable σ is never used at all, and so the adversary's output is independent of σ.

Let R_4 be the event that some decryption queries that would have passed in $\mathbf{G_2}$, but fail to pass in Step D_{2-1} in $\mathbf{G_4}$. Then we have the following facts.

Lemma 2 *$Pr[T_4] = Pr[T_3]$ and $Pr[R_4] = Pr[R_3]$.*

Game $\mathbf{G_5}$: This game is identical to $\mathbf{G_4}$, except for the following modification. In the decryption algorithm, we add the following *special rejection rule*, to prevent $\mathcal{A}$ from submitting an illegal enabling block to the decryption oracle once she has received her challenge T^*.

Special rejection rule: After the adversary $\mathcal{A}$ receives the challenge $T^* = (S^*, u_1^*, u_2^*, u_3^*, (c^r d^{r\alpha})^*, (j_1^*, F_{j_1}^*), \ldots, (j_z^*, F_{j_z}^*))$, the decryption oracle rejects any query $< i, T >$, with $T = (S, u_1, u_2, u_3, (c^r d^{r\alpha}), (j_1, F_{j_1}), \ldots, (j_z, F_{j_z}))$, such that $(S^*, u_1^*, u_2^*) \neq (S, u_1, u_2)$, but $\alpha = \alpha^*$, and it does so before executing the test in Step D_{2-1}.

To analyze this game, we define two events. Let C_5 be the event that the adversary $\mathcal{A}$ submits a decryption query that is rejected using the above special rejection rule, and R_5 the event that the adversary $\mathcal{A}$ submits some decryption query that would have passed in $\mathbf{G_2}$, but fails to pass in Step D_{2-1} in $\mathbf{G_5}$. It is clear that $\mathbf{G_4}$ and $\mathbf{G_5}$ proceed identically until event C_5 occurs. In particular, the event $R_4 \wedge \neg C_5$ and $R_5 \wedge \neg C_5$ are identical. So by Lemma 1, we have

$$|Pr[R_5] - Pr[R_4]| \leq Pr[C_5]$$

At this point, if event C_5 occurs with non-negligible probability, we can construct a PPT algorithm $\mathcal{A}_2$ that breaks the collision resistance assumption with non-negligible probability. So, $|Pr[C_5]| \leq \epsilon_2$ for some negligible ϵ_2.

Finally, we show that event R_5 occurs with negligible probability.

Lemma 3 *$Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$.*

Where, $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary $\mathcal{A}$. The proof of Lemma 2 and Lemma 3 are omitted due to space limitations, they can be easily extracted from [13].

Finally, combining the intermediate results, we conclude that the adversary $\mathcal{A}$'s advantage is negligible:

$$Adv_{Ourscheme,\mathcal{A}}^{CCA2}(\lambda) \leq \epsilon_1 + \epsilon_2 + \frac{Q_{\mathcal{A}}(\lambda)}{q}$$

□

Vulnerability of a Mobile Payment System Proposed at WISA 2002

Sang Cheol Hwang[1], Dong Hoon Lee[2], Daewan Han[2], and Jae-Cheol Ryou[1,⋆]

[1] Choongnam National University
220 Gung-dong, Yuseong-gu, Daejeon 305-764, Korea
{schwang,jcryou}@cnu.ac.kr
[2] National Security Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
{dlee,dwh}@etri.re.kr

Abstract. In this paper, we analyze the one-way mobile payment system proposed by Ham *et al.* at WISA 2002. They claimed that the electronic cash of the system satisfies unforgeability and double spending prevention. However, we show that the forgery of valid payment scripts is possible since the purchase is not authenticated, thus the proposed system is not secure.

1 Introduction

Electronic commerce means all aspects of business and market processes (including buying and selling products and services) enabled by the Internet. The electronic cash (also called digital cash or electronic money) means the digital information equivalent to the paper (real) cash in the electronic commerce. Since digital cash is merely an electronic representation of real money, it is possible to easily duplicate and spend a certain amount of money more than once. Therefore, electronic cash systems must have a structure to prevent the user from double spending or overspending. Besides, there are several requirements for the electronic cash system: double-spending prevention, anonymity, unforgeability, divisibility, transferability etc.

The theoretical foundations of electronic cash were established by Chaum. Chaum presented the first off-line and untraceable electronic cash [2, 3]. After that much researches have been performed in the area of electronic cash [6, 4, 1, 7]. Most electronic cash protocols assume that the participants (customers, vendors, and banks, etc.) are connected via wired connection with desktop environment, thus have enough computational power and bandwidth.

However the advent of mobile commerce has added other problems such as performance degradation and new threats from the use of constrained devices and wireless network. Hence we should simplify the protocol in terms of both computational power needed and amount of data transferred by sacrificing some

⋆ The fourth author of this research was supported by University IT Research Center Project of Korea MIC (Ministry of Information and Communication).

requirements of the electronic cash system. Recently, a mobile payment system was proposed by Ham and others (From now on, we call it Ham's system in brief) [5]. They claimed that the system fulfills unforgeability and double-spending prevention, and it needs only low computation load without any expensive modular exponentiation.

In this paper, we analyze Ham's system and point out that the system is not secure as they claimed. We show that the purchase phase of the system cannot give any authentication, thus eavesdroppers or malicious vendors can forge valid payment transaction scripts. More precisely, because the equations of the purchasing phase consist of almost linear terms, we can easily derive a part of the private key of the customer. Then we present a few practical attack scenarios for the system that may happen when this part of the private key is revealed.

The rest of the paper is organized as follows: We describe briefly Ham's system in section 2. We analyze the system and present a few attack scenarios in section 3, and conclude this paper in section 4.

2 Ham's Mobile Payment System

2.1 Model of Electronic Payment System

We describe the electronic commerce model specified by Ham *et al.* There are three parties involved: the customers(C), the vendors(V), and the banks(B) (see Figure 1).

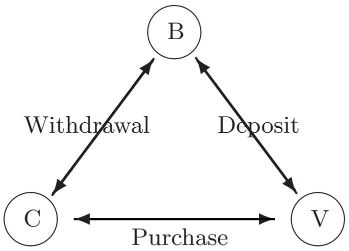


Fig. 1. Model of electronic payment system.

The electronic payment system consists of three protocols: Withdrawal, Purchase, and Deposit. Withdrawal and Deposit protocols are performed over a secure wired channel and only Purchase protocol is done through the wireless (open) networks. Hence it is possible to be eavesdropped during the purchase phase.

Each protocol has the following main functions:

Withdrawal (C $\leftrightarrow$ B): C requests setting-up for the mobile payment to **B**. Then **B** decides and publishes initial value observing the predetermined rule and issues the confirmation receipt on the request.

Purchase (C $\leftrightarrow$ V): **C** constructs the payment script to match the negotiated price of a product and pays it to **V**. Then **B** verifies the correctness of the payment script.

Deposit (V $\leftrightarrow$ B): **V** asks deposit of the payment script received from **C**. Then **B** reimburses the appropriate amount of money to **V** after verification.

The security requirements posed in [5] for a mobile payment system are as follows:

– Unforgeability: Only authorized entity (e.g. Bank) can issue valid money.
– Double-spending prevention: Customers cannot reuse issued money more than once. For *off-line* digital cash, double-spending prevention might be impossible. We can only detect it after double-spending.
– Efficiency: The payment system must be efficient with respect to storage, communication, and computation.

2.2 Notations

– p, q: large primes such that $q|p-1$.
– $\mathbb{Z}_p^*$: a multiplicative group of integers modulo p.
– G_q: a subgroup of $\mathbb{Z}_p^*$ of order q where the discrete logarithm problem is hard.
– g : a generator of G_q.
– $[2, q-1]$: the set of integers greater than 1 and less than q.
– $r \in_R [2, q-1]$: an integer r is chosen an random from $[2, q-1]$.
– a^{-1}: the multiplicative inverse of $a \in \mathbb{Z}_p^*$.
– $\|$: the concatenation of two strings.
– $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l (l \geq 160)$: a collision resistant one-way hash function.
– SK_{ID}, PK_{ID}: the private and public key of the entity ID respectively.
– SK_{ID}[messages]: the signature for 'messages' of the entity ID.

2.3 Description of Ham's System

Assume that the electronic payment is done *off-line*, that is, the communication with a bank is unnecessary during the purchase. The system consists of three protocols: Withdrawal, Purchase, and Deposit. These protocols are depicted in Figure 2, Figure 3, and Figure 4, respectively. Since Purchase protocol is done through the wireless (open) networks, malicious attackers may eavesdrop on the transmitted data during the purchase phase.

Each customer has two private and public key pairs, $(x_1; y_1(= g^{x_1}))$ and $(x_2; y_2(= g^{x_2}))$ and public keys are opened to the public. Generation and distribution of these key pairs follow the usual public key cryptosystems except that one has two public key pairs. In addition, they make use of a hash chain technique, *Keyed Hash Chain*, to ascertain uniqueness of the current transaction in the scheme.

Generation: $KH_0 = k$, where k is a secret information.
$KH_i = \mathcal{H}(k \| KH_{i-1})$ where $i = 1, \cdots, m$.
Usage: Increasing order from 1 to m.

Withdrawal. A customer **C** requests to the bank **B** a withdrawal of amount σ_0 with his identity ID_C, account information AI, and additional information ϵ. If the request is valid, **B** generates the random secret value k which is used in the keyed hash chain, and issues the money with a confirmation receipt $\texttt{Conf}_C$. **B** publishes the identity with y_k, a validity time period T, and the signature DS_B in the public domain (PD). **C** stores his(her) own secret values, the transmitted $k(= KH_0)$ and $\texttt{Conf}_C$ in his mobile device (MD). See Figure 2.

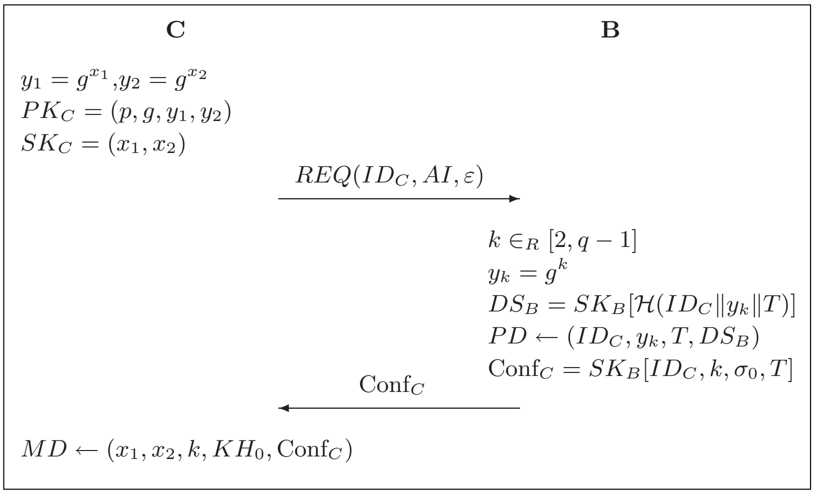

C
B
$y_1 = g^{x_1}, y_2 = g^{x_2}$
$PK_C = (p, g, y_1, y_2)$
$SK_C = (x_1, x_2)$
$REQ(ID_C, AI, \varepsilon)$
$k \in_R [2, q-1]$
$y_k = g^k$
$DS_B = SK_B[\mathcal{H}(ID_C \| y_k \| T)]$
$PD \leftarrow (ID_C, y_k, T, DS_B)$
$\text{Conf}_C = SK_B[ID_C, k, \sigma_0, T]$
Conf_C
$MD \leftarrow (x_1, x_2, k, KH_0, \text{Conf}_C)$

Fig. 2. Withdrawal protocol.

Purchase. Purchase protocol is carried out between **C** and **V** over the wireless channel, and is unilateral from **C** to **V**. Assume that σ is the negotiated price of i-th purchase of **C**. **C** calculates (α, β, ω) and sends them with his identity ID_C and **B**'s identity ID_B (See Figure 3).

$$\begin{aligned}
&\alpha = x_1 + r \text{ for randomly chosen } r \\
&\beta = k + KH_i \\
&\omega = r^{-1}(\sigma x_1 + x_2 + KH_i + \mathcal{H}(ID_C \| ID_V \| ID_B))
\end{aligned}$$

We note that in the above almost all operations are linear.

V gets public information (y_1, y_2, y_k) of **C** from public domain and verifies the validity of the payment script as follows:

$$\left(\frac{g^\alpha}{y_1}\right)^\omega \stackrel{?}{=} y_1^\sigma y_2 \left(\frac{g^\beta}{y_k}\right) g^{\mathcal{H}(ID)}$$

where $\mathcal{H}(ID) = \mathcal{H}(ID_C \| ID_V \| ID_B)$.

Deposit. **V** sends the stored payment script $(ID_V, ID_C, \omega, y_r, g^{KH_i}, REQ(\sigma))$ to the bank **B** for deposit as depicted in Figure 4.

C **V**

$\mathcal{H}(ID) = \mathcal{H}(ID_C \| ID_V \| ID_B)$
$KH_i = \mathcal{H}(k \| KH_{i-1})$
$r \in_R [2, q-1]$
$\alpha = x_1 + r$
$\beta = k + KH_i$
$\omega = r^{-1}(\sigma x_1 + x_2 + KH_i + \mathcal{H}(ID))$

$$\xrightarrow{ID_C, ID_B, (\alpha, \beta, \omega)}$$

Get y_1, y_2, y_k
P1) $\mathcal{H}(ID) = \mathcal{H}(ID_C \| ID_V \| ID_B)$
P2) $g^{\alpha}/y_1 = g^r (= y_r)$
P3) $g^{\beta}/y_k = g^{KH_i}$
P4) Check $y_r^{\omega} = y_1^{\sigma} y_2 g^{KH_i} g^{\mathcal{H}(ID)}$

Fig. 3. Purchase protocol.

B compares the payment script with the stored list of ID_C. If **B** finds the same g^{KH_i} in the list, **B** concludes that a double spending has happened and closes the protocol. Otherwise, **B** verifies the validity of the script as follows:

$$g^{KH_i} \stackrel{?}{=} g^{\mathcal{H}(k \| KH_{i-1})}$$
$$y_r^{\omega} \stackrel{?}{=} y_1^{\sigma} y_2 g^{KH_i} g^{\mathcal{H}(ID)}$$

Then **B** deposits the money of amount σ and keeps ID_C and g^{KH_i} for a period T, the lifespan of k.

3 Analysis of Ham's Scheme

In this section, we show that the payment scheme proposed by Ham and others is not secure as the authors claimed. We show that the attacker who has two purchase scripts of a certain customer can extract a part of the private key of the customer. The attacker can also forge new valid scripts, thus he can purchase any product from vendors by impersonating the victim.

3.1 Extraction of C's Private Key

We assume the following for our attack.

- A1: The attacker **E** can obtain two payment scripts that have used the same k value.
- A2: **E** can obtain the hash value $\mathcal{H}(ID)$ used in the scripts.
- A3: **E** can obtain the price σ of the product in the scripts.

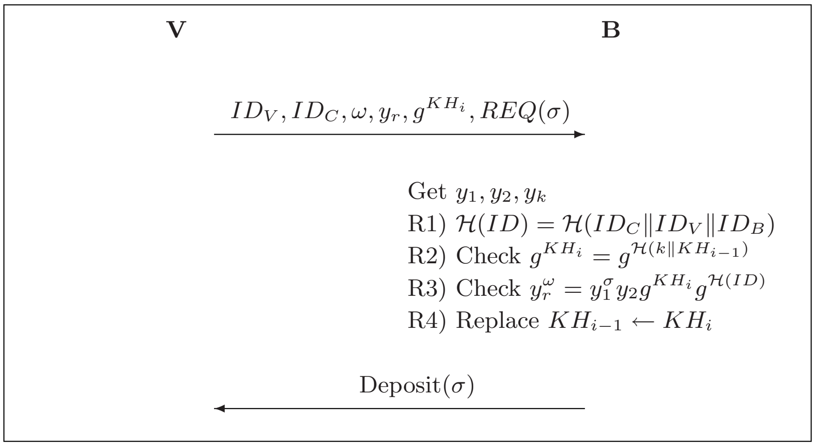

V
B
$ID_V, ID_C, \omega, y_r, g^{KH_i}, REQ(\sigma)$
Get y_1, y_2, y_k
R1) $\mathcal{H}(ID) = \mathcal{H}(ID_C \| ID_V \| ID_B)$
R2) Check $g^{KH_i} = g^{\mathcal{H}(k \| KH_{i-1})}$
R3) Check $y_r^{\omega} = y_1^{\sigma} y_2 g^{KH_i} g^{\mathcal{H}(ID)}$
R4) Replace $KH_{i-1} \leftarrow KH_i$
Deposit(σ)

Fig. 4. Deposit protocol.

The first two assumptions are very practical especially in mobile public key infrastructures, and the third one also can be assumed naturally in general payment systems. If they can't be assumed in some situation, we may restrict our attack to the case in which the attacker is a vendor **V** who participates in the protocol. Then, the above assumptions need not be made, because **V** can obtain all the variables in A1, A2, and A3 naturally during the process of the protocol. Although our attack is simpler in such a case, for generality we describe our attack assuming the attacker is a third party with the assumptions A1, A2, and A3.

Suppose **E** knows two payment scripts, (α, β, ω) which is sent from **C** to $\mathbf{V}_1$ and $(\alpha', \beta', \omega')$ from **C** to $\mathbf{V}_2$. Then **E** knows the following relations:

$$\alpha = x_1 + r, \tag{1}$$
$$\beta = k + KH_i, \tag{2}$$
$$\omega = r^{-1}(\sigma x_1 + x_2 + KH_i + \mathcal{H}(ID)), \tag{3}$$
$$\alpha' = x_1 + r', \tag{4}$$
$$\beta' = k + KH_j, \tag{5}$$
$$\omega' = r'^{-1}(\sigma' x_1 + x_2 + KH_j + \mathcal{H}(ID')). \tag{6}$$

Equation (3) can be written as

$$r\omega = \sigma x_1 + x_2 + KH_i + \mathcal{H}(ID),$$

and substituting r with $\alpha - x_1$ from (1), we obtain the following equation.

$$\alpha\omega = (\sigma + \omega)x_1 + x_2 + KH_i + \mathcal{H}(ID). \tag{7}$$

Similarly, from equations (4) and (6), we obtain

$$\alpha'\omega' = (\sigma' + \omega')x_1 + x_2 + KH_j + \mathcal{H}(ID'). \tag{8}$$

Subtracting (8) from (7), we obtain

$$\alpha\omega - \alpha'\omega' = (\sigma + \omega - \sigma' - \omega')x_1 + (KH_i - KH_j) + \mathcal{H}(ID) - \mathcal{H}(ID').$$

From (2) and (5) we see that $(KH_i - KH_j)$ can be substituted by $(\beta - \beta')$, thus we finally obtain the equation

$$(\sigma + \omega - \sigma' - \omega')x_1 = (\alpha\omega - \alpha'\omega') - (\beta - \beta') - (\mathcal{H}(ID) - \mathcal{H}(ID')).$$

All variables except x_1 are known and the probability that $(\sigma + \omega - \sigma' - \omega') = 0$ is negligible. Therefore, the attacker can recover the part of the private key x_1 of **C** by solving the above linear equation.

3.2 Forgery of Purchase Scripts and Collusion of Vendors

The attacker **E** who knows the private key x_1 of the customer **C** can obtain r in equation (1), and $x_2 + KH_i$ in equation (3) from r. Then he can construct another script $(\alpha'', \beta'', \omega'')$ for the price σ'' from (α, β, ω) as follows:

$$\begin{aligned}\mathcal{H}(ID'') &= \mathcal{H}(ID_C \| ID_{V''} \| ID_B)\\ \alpha'' &= \alpha\\ \beta'' &= \beta\\ \omega'' &= r^{-1}[\sigma'' x_1 + (x_2 + KH_i) + \mathcal{H}(ID'')]\end{aligned}$$

It is easily checked that $(ID_C, ID_B, (\alpha'', \beta'', \omega''))$ is a *valid* purchase script. Therefore the attacker can forge valid purchase scripts, and buy other products from a vendor $\mathbf{V}''$, impersonating **C**, using the forged scripts.

3.3 Attack Scenarios

Overspending. The electronic cash of the proposed payment system has a property of implicit divisibility. That is, a customer who has withdrawn electronic money of σ_0-value from his account can purchase any product whose price is $\sigma \leq \sigma_0$. However, since vendors do not verify the amount of money remaing, the customers can intentionally overspend.

Forgery of Money. Since the payment scripts are transmitted via wireless channel, malicious parties may easily eavesdrop on transmission. Hence the attacker may obtain two payment scripts of a victim, (α, β, ω) and $(\alpha', \beta', \omega')$, for the prices of σ and σ' respectively.

According to Section 3.1 and Section 3.2, the attacker can produce a forged payment script $(\alpha'', \beta'', \omega'')$ for a product of price σ'' and impersonate the victim to any vendors. In this case, the attacker sets σ'' to any value since vendors do not verify the amount of money remaining.

If a vendor executes Deposit protocol with the forged script $(\alpha'', \beta'', \omega'')$ and Deposit protocol with (α, β, ω) was executed already (by another vendor), then g^{KH_i} is presented twice. Then bank would conclude that the victim has doubly spent the issued money illegally.

Collusion of Two Vendors. If at least two vendors collude, they can extract the private key x_1 of the victim as described in Section 3.1 and Section 3.2, and forge a payment script $(\alpha'', \beta'', \omega'')$ for a product of price σ'' such that $\sigma'' > \sigma$. If the vendor sends the forged script at the Deposit protocol instead of the original script (α, β, ω), then the bank shall accept the forged payment script as a valid one. In this case, the bank may conclude that the victim has illegally spent the issued money over the limit σ_0.

4 Conclusions

We have analyzed the mobile payment system proposed at WISA 2002. Because the purchase script is not authenticated, we could forge new valid scripts by extracting a part of the private key of the customer.

We suggest that irreversible operations such as modular exponentiation should be involved for the payment system to be secure. Moreover, the signature of customers and vendors might be involved to authenticate the money remaining.

References

1. S. Brands, Untraceable off-line cash in wallets with observer, *Advances in Cryptology - CRYPTO'93*, LNCS 773, Springer-Verlag, pp.302–318, 1994.
2. D. Chaum, Blind signature for untraceable payment, *Advances in Cryptology - CRYPTO'82*, LNCS , Springer-Verlag, pp.199–203, 1983.
3. D. Chaum, A. Fiat, and M. Naor, Untraceable electronic cash, *Advances in Cryptology - CRYPTO'88*, LNCS 403, Springer-Verlag, pp.319–327, 1990.
4. N. Ferguson, Single term off-line coins, *Advances in Cryptology - Eurocrypt'93*, LNCS 765, Springer-Verlag, pp.318–328, 1994.
5. W. Ham, H. Choi, Y. Xie, M. Lee, and K. Kim, Secure one-way mobile payment system keeping low computation in mobile devices, *The third Workshop on Information Security Applications (WISA 2002)*, pp.287-301, 2002.
6. T. Okamoto and K. Ohta, Universal electronic cash, *Advances in Cryptology CRYPTO'91*, LNCS 576, Springer-Verlag, pp.324–337, 1992.
7. T. Okamoto, An efficient divisible electronic cash scheme, *Advances in Cryptology - CRYPTO'95*, LNCS 963, Springer-Verlag, pp.438–451, 1995.

Fair Offline Payment Using Verifiable Encryption*

Sangjin Kim[1] and Heekuck Oh[2]

[1] Korea Univ. of Technology and Education, School of Internet Media Eng., Byeoncheonmyeon, Cheonan, Chungnam, Republic of Korea
sangjin@kut.ac.kr
http://infosec.kut.ac.kr/sangjin/

[2] Hanyang University, Department of Computer Science and Engineering, Sa-1-dong 1271, Ansan, Kyunggi, Republic of Korea
hkoh@cse.hanyang.ac.kr
http://infosec.hanyang.ac.kr/∼hkoh/

Abstract. Verifiable encryption allows a receiver, who cannot decrypt the ciphertext, to verify what has been encrypted. This technique is widely used in fair exchange to convince a receiver that he/she can later obtain the requested item by presenting the item in an encrypted form to a TTP (Trusted Third Party). In this paper, we apply verifiable encryption to offline payment systems based on the representation problem to provide the payment atomicity. Our verifiable encryption uses the Naccache-Stern cryptosystem and a proof of equality of discrete logarithms from different groups. Although additional cost is required during payments, we show that the cost is reasonable. Furthermore, we have improved the efficiency of dispute settlement significantly. In our method, the TTP does not have to interact with any other party other than the one who filed a complaint to resolve disputes.

1 Introduction

Generally, a payment protocol using e-cash involves exchanging e-cash with digital goods between a client and a shop over the network. Normally, clients and shops do not trust each other and protocol executions can be pre-terminated accidentally or deliberately. Therefore, payment atomicity must be preserved to prevent a party from gaining an illegal profit or suffering a loss. Tygar [1] was the first to address this issue. But his method requires online participation of the TTP, thus cannot be considered as a solution to fair offline payment. Boyd and Foo [2] used an optimistic approach, meaning that the TTP participates only when disputes arise. Since their method is based on convertible signatures, their method inherently does not consider anonymous participation of clients. Xu et al. [3] proposed an another optimistic method that considers client's anonymity.

* This work was supported in part by Korea Science and Engineering Foundation under Contract R05-2004-000-11084-0.

Researches on providing fairness in payment using offline e-cash have only received a small attention, compared to vast amount of researches on e-cash. On the other hand, fair exchange [4, 5] has been studied for some time. People generally think that fairness of payment using offline e-cash can be easily provided by applying research results from fair exchange. But to date, researches on fair exchange have not considered anonymous participation of players involved and their main focus was on exchanging participant's digital signatures fairly [5, 6]. Compared to this, anonymity is a very important property in offline e-cash systems. Moreover, in these systems, instead of exchanging participant's signatures, digital goods are exchanged with e-cash, which is normally issued by a bank.

Recently, verifiable encryption has emerged as a key tool in building efficient fair exchange protocols [5, 6]. Verifiable encryption allows a sender to encrypt a message with a designated public-key and subsequently convinces a receiver, who cannot decrypt the ciphertext, that the ciphertext actually contains such message. In fair exchange, before exchanging actual items, they are exchanged in an encrypted form using verifiable encryption. Actually, only one of the items being exchanged is required to be encrypted. The encryption is done using the public key of the TTP. Therefore, if a receiver does not receive the actual item, he/she can present the item in an encrypted form to the TTP and obtain it.

Most of the current offline e-cash systems provide anonymity of clients. What follows are the basic requirements that must be met by methods used to preserve payment atomicity in such systems.

– *Clients participate anonymously.* Additional steps taken to provide payment atomicity should not affect clients' anonymity. Moreover, clients should not have to reveal their identity to file a complaint. If they are required to do so, they may give up their rights for the sake of preserving their anonymity.
– *TTP should only participate when disputes arise.*
– *Except the participating client and the shop, others should not be able to obtain the items being exchanged.* This means only the shop should be able to deposit the used coins and only the client should be able to receive the goods. The former is provided by including the shop's identity in the challenge of the payment. The latter is provided by exchanging the goods encrypted.

In this paper, we propose a new method of providing the payment atomicity for offline e-cash system based on the representation problem [7]. For this purpose, we have devised a new method for verifiably encrypting DL(Discrete Logarithm)s. This verifiable encryption uses the Naccache-Stern cryptosystem [8] and a proof of equality of DLs from different groups [9]. Although such proof normally requires an interval proof of DL [9], we show that we can do without it. As a result, additional cost required during payments is reasonable. Furthermore, we have improved the efficiency of dispute settlement significantly. In our method, the TTP does not have to interact with any other party other than the one who filed a complaint to resolve disputes.

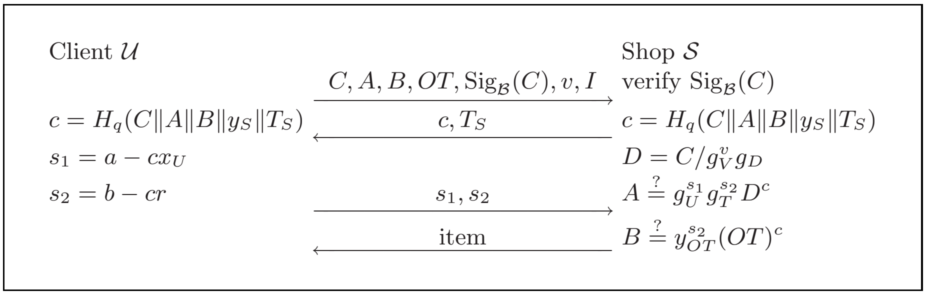

Client $\mathcal{U}$
Shop $\mathcal{S}$
$C, A, B, OT, \mathrm{Sig}_{\mathcal{B}}(C), v, I$
verify $\mathrm{Sig}_{\mathcal{B}}(C)$
$c = H_q(C\|A\|B\|y_S\|T_S)$
c, T_S
$c = H_q(C\|A\|B\|y_S\|T_S)$
$s_1 = a - cx_U$
$D = C/g_V^v g_D$
$s_2 = b - cr$
s_1, s_2
$A \overset{?}{=} g_U^{s_1} g_T^{s_2} D^c$
item
$B \overset{?}{=} y_{OT}^{s_2}(OT)^c$

Fig. 1. The Basic Payment Protocol.

2 Related Work

In this paper, unless otherwise stated, all operations are performed in a subgroup G_q of order q in $\mathbb{Z}_p^*$, where p and q are large prime numbers such that $q|(p-1)$.

2.1 Electronic Payment System

In this paper, we use an e-cash system similar to the one proposed by Solages and Traore [10] to demonstrate our new method. In this system, a coin is represented as $C = g_U^{x_U} g_V^v g_T^r g_D$, where g_U, g_V, g_T, and g_D are generators of G_q, x_U represents the client's private identifier, v is the face value of the coin, and r is a blind factor. Clients perform a restrictive blind signature with the bank to withdraw a coin. The resulting coin is represented by the following tuple:

$$A = g_U^a g_T^b,\ B = y_{OT}^b,\ OT = y_{OT}^r,\ C,\ \mathrm{Sig}_{\mathcal{B}}(A||B||OT||C),$$

where '$||$' denotes bitwise concatenation, $y_{OT} = g_T^{x_{OT}^{-1}}$ represents the TTP's owner tracing public key, and $\mathrm{Sig}_{\mathcal{B}}(A||B||OT||C)$ represents the bank's signature on $A||B||OT||C$. From now on, we will use $\mathrm{Sig}_{\mathcal{B}}(C)$ instead of $\mathrm{Sig}_{\mathcal{B}}(A||B||OT||C)$ to simplify our discussion. A, B, and OT are values committed by the client during the blind signature and are used in the payment protocol.

In a coin-based system, since the system does not provide all possible denominations, clients normally pay using several coins. But to simplify our discussion, we assume that clients always use a single coin during payments. The payment protocol of this system is depicted in Fig. 1. In this protocol, I denotes the item identifier, y_S denotes the shop's identifier, T_S denotes the purchase date and time, and $H_q : \{0,1\}^* \to \mathbb{Z}_q$ denotes a collision-resistant hash function. To buy an item I, the client sends a coin C of value v and proves that he/she knows the representation of C with respect to the generator tuple (g_U, g_V, g_T, g_D) [7]. The client also proves that the owner of C can be traced using OT. The shop can only deposit the coin if it has a valid challenge and responses. Since the shop's identifier y_S is included in the calculation of the challenge c, only y_S can claim the money. For more detail on this system, refer to [10].

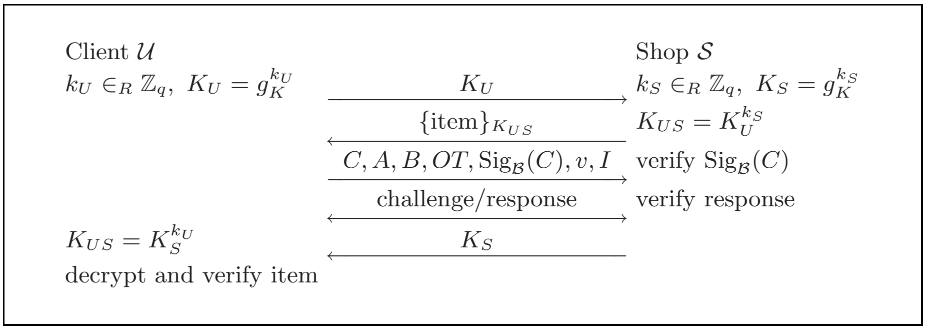

Client $\mathcal{U}$
Shop $\mathcal{S}$
$k_U \in_R \mathbb{Z}_q,\ K_U = g_K^{k_U}$
K_U
$k_S \in_R \mathbb{Z}_q,\ K_S = g_K^{k_S}$
$\{\text{item}\}_{K_{US}}$
$K_{US} = K_U^{k_S}$
$C, A, B, OT, \mathrm{Sig}_{\mathcal{B}}(C), v, I$
verify $\mathrm{Sig}_{\mathcal{B}}(C)$
challenge/response
verify response
$K_{US} = K_S^{k_U}$
K_S
decrypt and verify item

Fig. 2. Payment Protocol of Xu et al.

2.2 Xu et al.'s Method of Providing Payment Atomicity

Xu et al. [3] proposed a method for providing fair offline payment. In their method, TTP participates only when disputes arise. Their payment protocol is illustrated in Fig. 2. In this protocol, g_K is a generator of G_q used in DH (Diffie-Hellman) key [11]. The differences between this and the basic protocol are i) the item is first sent to the client encrypted using a DH key, and ii) shops must conform to a deposit deadline.

The shop sends its share of DH key only when it has received valid responses from the client. Therefore, it is not possible for a shop to suffer a loss whereas clients can suffer a loss. The reason behind this is that clients participate anonymously which makes it difficult for shops to file a complaint. As a result, only clients can initiate the recovery protocol when the protocol is pre-terminated. The followings are the circumstances which require a client to file a complaint.

- **Case 1:** The shop refuses to send its share of DH key after receiving valid responses.
- **Case 2:** The shop deliberately or accidentally sent a wrong item.

Filing a complaint can occur only when the deposit deadline is over. Although using a deadline reduces the number of possible states that the TTP has to consider, it is neither flexible nor convenient for clients and shops. We can remove this deadline, but it complicates dispute settlement.

In case 1, the client files a complaint to the TTP by sending the coin involved in the payment to the TTP. The client also proves the ownership of the coin by proving that he/she knows the representation of the coin. The TTP contacts the bank and asks whether the coin has been deposited or not. If the coin has been deposited, the TTP demands the shop to give its share of the decryption key. If the shop does not comply, it faces a legal action. If the coin has not been deposited, the TTP cancels the payment and the coin is refunded.

In case 2, in Xu et al.'s method, it is not easy to prove that the shop has sent a wrong item. Although the item identifier is included in the challenge of the payment, this is insufficient for proving what has been exchanged. Furthermore, since the bank gets to know what has been purchased, it affects unobservability of the payment. Invoice such as $\text{Sig}_S(\{\text{item}\}_K||I||\text{price}||C||y_S||T_S)$ can be used to solve this problem [12]. This invoice is sent to the client along with the encrypted item. This invoice can later be used as a proof of what the shop has sent and that the shop actually participated in the payment.

3 Mathematical Background

3.1 Naccache-Stern Public Key Cryptosystem

Naccache and Stern [8] proposed a new public key cryptosystem based on the hardness of computing higher residues modulo a composite RSA integer. We will refer to this cryptosystem as NS cryptosystem from now on. They proposed two versions: one deterministic and the other probabilistic. In this paper, we use the deterministic one. To setup this system, we need to choose a RSA modulus $n = PQ$, σ, and $g \in \mathbb{Z}_n^*$. As usual P and Q are large primes and n must be at least 768 bits long. σ is a squarefree odd B-smooth integer that must be at least 160 bits long, where B is a small integer about 10 bits long. This σ must divide $\phi(n)$ and be prime to $\phi(n)/\sigma$. The order of g must be a large multiple of σ. The generation of modulus is done as follows. First, k small odd distinct primes p_i are picked, where k is even. Then set $u = \prod_{i=1}^{k/2} p_i$, $v = \prod_{k/2+1}^{k} p_i$, and $\sigma = uv = \prod_{i=1}^{k} p_i$. Finally, pick two large primes a and b such that both $P = 2au + 1$ and $Q = 2bv + 1$ are primes. g is selected by randomly choosing it and testing whether or not it has order $\phi(n)/4$.

Encryption of message $m < \sigma$ is done by $c = g^m \mod n$. Decryption of c is based on the Chinese remainder theorem. First, the following is computed for all prime factors p_i of σ, where $y_i = (m - m_i)/p_i$.

$$c_i = c^{\frac{\phi(n)}{p_i}} \equiv g^{m\frac{\phi(n)}{p_i}} \equiv g^{\frac{(m_i+y_ip_i)\phi(n)}{p_i}} \equiv g^{\frac{m_i\phi(n)}{p_i}} g^{y_i\phi(n)} \equiv g^{\frac{m_i\phi(n)}{p_i}} \pmod n$$

Then, we compare c_i with all possible powers $g^{\frac{j\phi(n)}{p_i}}, j = 1, \ldots, p_i$ to compute m_i, which is congruent to m modulo p_i, for each c_i. Finally, we compute the simultaneous congruences $m \equiv m_i \pmod{p_i}$ using the Chinese remainder theorem to obtain m. For more detail on this system, refer to [8].

Assumption 1. NS cryptosystem is secure if computing higher residues modulo n and factoring n are computationally infeasible, where n is the RSA modulus of the system.

3.2 Proofs of Equality of Discrete Logarithms from Different Groups

In our proposed payment method, instead of sending s_1, we send $E = g_U^{s_1}$ along with $V = g^{s_1} \mod n$, which is an encryption of s_1 using the NS cryptosystem, and prove $\log_{g_U} E = \log_g V$. Since the order of G_q and $\mathbb{Z}_n^*$ are different, we need a proof of equality of DLs from different groups.

$$\begin{array}{lcl}
\text{Prover } \mathcal{P} & & \text{Verifier } \mathcal{V} \\
w \in_R \mathbb{Z}_q & & \\
W_1 = g_1^w,\ W_2 = g_2^w & \xrightarrow{\quad W_1, W_2 \quad} & c \in_R \{0,1\}^k \\
s = w - cx & \xleftarrow{\quad c \quad} & \\
 & \xrightarrow{\quad s \quad} & W_1 \overset{?}{\equiv} g_1^s y_1^c,\ W_2 \overset{?}{\equiv} g_2^s y_2^c
\end{array}$$

Fig. 3. A Proof of Equality of Discrete Logarithms from the Same Group.

Proofs of Equality of Discrete Logarithms from the Same Group. We can prove the equality of DLs of $y_1 \in G_q$ and $y_2 \in G_q$ to the bases g_1 and g_2, respectively, using the proof given in Fig. 3 [13]. One can think that the equality of DLs from different groups can also be proven using the proof depicted in Fig. 3 by computing s in $\mathbb{Z}$ without applying the modulus. However, due to the following attack, this is not possible.

Attack 1. Let $G_1 = \langle g_1 \rangle$ and $G_2 = \langle g_2 \rangle$ be two distinct groups of order q_1 and q_2, respectively. We assume $q_1 < q_2$ and $\gcd(q_1, q_2) = 1$. Let $y_1 = g_1^x$, $y_2 = g_2^{x'}$, $W_1 = g_1^w$, and $W_2 = g_2^{w'}$, where $x \neq x'$. In this case, if a prover knows the orders of G_1 and G_2, the prover can cheat by computing s satisfying $s \equiv w - cx \pmod{q_1}$ and $s \equiv w' - cx' \pmod{q_2}$ using the Chinese remainder theorem. This s is unique in $\mathbb{Z}_{q_1 q_2}$.

We must note that if s is below q_1 the $w - cx$ must be equal to $w' - cx'$. However, since a prover can only guess c with the probability of $1/2^k$, we can disregard this possibility. Therefore, if we can somehow limit s's range, then we can prevent Attack 1. To limit s's range, we have to also limit x's and w's range.

Interval Proofs of Discrete Logarithm. A prover can use an interval proof of DL to prove his/her knowledge of DL, while simultaneously proving that DL lies in a specified interval in zero-knowledge style. An interval proof of DL using binary challenge is depicted in Fig. 4 [9]. In this proof, q must be larger than 2^{l+1}, and the prover is proving that x lies in the interval $(-2^l, 2^l)$. However, the prover can only carry out this protocol successfully, if x lies in the interval $(-2^{l-2}, 2^{l-2})$. Thus, only membership to a much larger interval can be proven. To see why the verifier is convinced that x lies in $(-2^l, 2^l)$, we must consider the knowledge extractor. Using the standard rewinding technique, the knowledge extractor can acquire two accepting transcripts with the same W, but different c's and s's. Therefore, from $g^s y^c = g^{s'} y^{c'}$, the extractor can compute $x = \frac{s'-s}{c-c'}$. Since the verifier checks that s's lie in the interval $(-2^{l-1}, 2^{l-1})$, $s' - s$ lies in the interval $(-2^l, 2^l)$. Moreover, since c's are binary, $x = \pm(s' - s)$. Therefore, x lies in the interval $(-2^l, 2^l)$.

Camenisch and Michels proposed a more efficient version based on the following assumption [9].

$$\begin{array}{lcl}
\text{Prover } \mathcal{P} & & \text{Verifier } \mathcal{V} \\
w \in_R \{-2^{l-2}, \ldots, 2^{l-2}\} & & \\
W = g^w & \xrightarrow{\qquad W \qquad} & c \in_R \{0,1\} \\
 & \xleftarrow{\qquad c \qquad} & \\
s = w - cx \quad (\text{in } \mathbb{Z}) & \xrightarrow{\qquad s \qquad} & -2^{l-1} \overset{?}{<} s \overset{?}{<} 2^{l-1},\ W \overset{?}{\equiv} g^s y^c
\end{array}$$

Fig. 4. An Interval Proof of Discrete Logarithm using Binary Challenge.

$$\begin{array}{lcl}
\text{Prover } \mathcal{P} & & \text{Verifier } \mathcal{V} \\
w \in_R \mathbb{Z}_q & & \\
W_1 = g^w \mod n,\ W_2 = g_U^w & \xrightarrow{\qquad W_1, W_2 \qquad} & c \in_R \{0,1\}^k \\
 & \xleftarrow{\qquad c \qquad} & \\
s = w - cx\ (\text{in } \mathbb{Z}) & \xrightarrow{\qquad s \qquad} & W_1 \overset{?}{\equiv} g^s V^c \pmod{n},\ W_2 \overset{?}{=} g_U^s E^c
\end{array}$$

Fig. 5. ZKProof($\log_{g_U} E = \log_g V$).

Assumption 2 (Strong RSA). Given only a sufficiently large RSA modulus n and $z \in \mathbb{Z}_n^*$, it is computationally infeasible to find a pair $(u, e) \in \mathbb{Z}_n^* \times \mathbb{Z}$ such that $e > 1$ and $z \equiv u^e \pmod{n}$.

Under the Assumption 2, we can use non-binary challenges when performing interval proofs. This is due to the fact that $c-c'$ always divides $s'-s$, which allows a verifier to draw conclusions about the size of the prover's secret. We now show why $c-c'|s'-s$ always holds under the Assumption 2. Let $d = \gcd(s'-s, c-c')$, then we can use the extended Euclidean algorithm to find u and v, satisfying $u\frac{c-c'}{d} + v\frac{s'-s}{d} = 1$. From this, we can see that the following holds.

$$g = g^{u\frac{c-c'}{d} + v\frac{s'-s}{d}} = g^{u\frac{c-c'}{d}}(g^{s'-s})^{\frac{v}{d}} = g^{u\frac{c-c'}{d}}(y^{c-c'})^{\frac{v}{d}} = (g^u y^v)^{\frac{c-c'}{d}}$$

But if $d < c - c'$ then we can compute a non trivial root of g, which contradicts the Assumption 2. Therefore, $d = c - c'$, which means $(c - c')|(s' - s)$. To use this assumption, we have to add a proof to the Fig. 4 that proves the knowledge of DL in RSA modulus.

Now let's consider the knowledge extractor in this case. Let's assume k-bit challenge is used. Since $x = \frac{s'-s}{c-c'}$, $-2^l < s' - s < 2^l$, and $1 \le |c - c'| < 2^k$, x lies in the interval $(-2^l, 2^l)$. Although we can efficiently prove the interval using this method, the required range of x is further decreased to $(-2^{l-2-k}, 2^{l-2-k})$. This does not cause any problem if x is chosen from this range, but direct use of this proof is difficult when x is computed and may have random values.

4 Fair Offline Payment Using Verifiable Encryption

4.1 Verifiable Encryption of Payment

In our method, we verifiably encrypt the payment before sending it to the shop to provide the payment atomicity. This can be achieved by either encrypting $\mathrm{Sig}_{\mathcal{B}}(C)$ or encrypting the responses that clients sends to the shop. The former is not flexible in that we need to apply different verifiable encryption techniques for each different signature schemes used to issue e-cash. Therefore, we choose the latter. In the protocol given in Fig. 1, there are two responses which are sent to the shop. Only one of these responses needs to be verifiably encrypted. We choose to verifiably encrypt s_1. We send $E = g_U^{s_1}$, s_2, and $V = g^{s_1} \mod n$ to the shop, where n and g are public key of the NS cryptosystem of the TTP, and proves that $\log_{g_U} E = \log_g V$ in zero knowledge style using the proof depicted in Fig. 5. In proofs explained in subsection 3.2, it is assumed that the prover knows the order of both groups. In our case, the client does not know the order of $\mathbb{Z}_n^*$, while he/she knows the order of G_q. Therefore, Attack 1 is not possible in our setting. However, there is an another attack that must be considered.

Attack 2. Let $G_1 = \langle g_1 \rangle$ and $G_2 = \langle g_2 \rangle$ be two distinct groups of order q_1 and q_2, respectively. We assume $q_1 < q_2$ and $\gcd(q_1, q_2) = 1$. Let $y_1 = g_1^x$ and $y_2 = g_2^{x'}$, where $x \neq x'$ but $x \equiv x' \pmod{q1}$. In this case, the prover can succeed in falsely proving the equality using the proof given in Fig. 5. Moreover, when y_2 is decrypted, the resulting value will not be x' but $x' \mod q_2$.

In our situation, the TTP can detect this kind of attack. Moreover, the TTP can use the tracing mechanism of e-cash to find the person responsible for it. Therefore, we do not use an additional interval proof to prevent Attack 2.

Claim 1. In our setting, it is safe to remove the interval proof from proofs of equality of DLs from different groups.

Proof Sketch. In our setting, since the prover does not know the order of one of the group, Attack 1 is not possible. However, Attack 2 is still possible. To prevent this attack, an interval proof of discrete logarithm is required. However, we do not have to resort to interval proofs due to the following reasons. First, the TTP can detect this attack and can use the tracing mechanism of the e-cash to find the person responsible for it. Second, since only one of the response is verifiable encrypted and the false value used in the proof must be congruent to the original value modulo the order of the smaller group, only the person who knows the representation of the e-cash can attempt this kind of attack. Other ways of cheating, such as guessing the correct s satisfying the Attack 1, or guessing the correct challenge c, can be made computationally infeasible by setting the system parameters appropriately.

We have to consider whether it is safe for the shop to verify that A is congruent to $Eg_T^{s_2}D^c$ modulo p in Fig. 1 instead of receiving s_1 and verifying as before.

Claim 2. It is secure for the shop to verify that A is congruent to $Eg^{s_2}D^c$ modulo p in Fig. 1, instead of receiving s_1 and verifying that A is congruent to $g_U^{s_1}g_T^{s_2}D^c$ modulo p, if the client also has to prove his/her knowledge of $\log_{g_U} E$.

Proof Sketch. An attacker, who does not know the representation of C, can pass this verification by randomly selecting s_2' and computing $E' = Ag_T^{-s_2'}D^{-c}$. However, by DL assumption, it is computationally infeasible for the attacker to compute $\log_{g_U} E'$. Therefore, if we require the client to additionally prove his/her knowledge of $\log_{g_U} E$, it is secure to verify in this way.

In our proposed payment protocol, a client must perform ZKProof($\log_{g_U} E = \log_g V$), which cannot be performed successfully if the client does not know $\log_{g_U} E$. Therefore, we can justifiably claim the following.

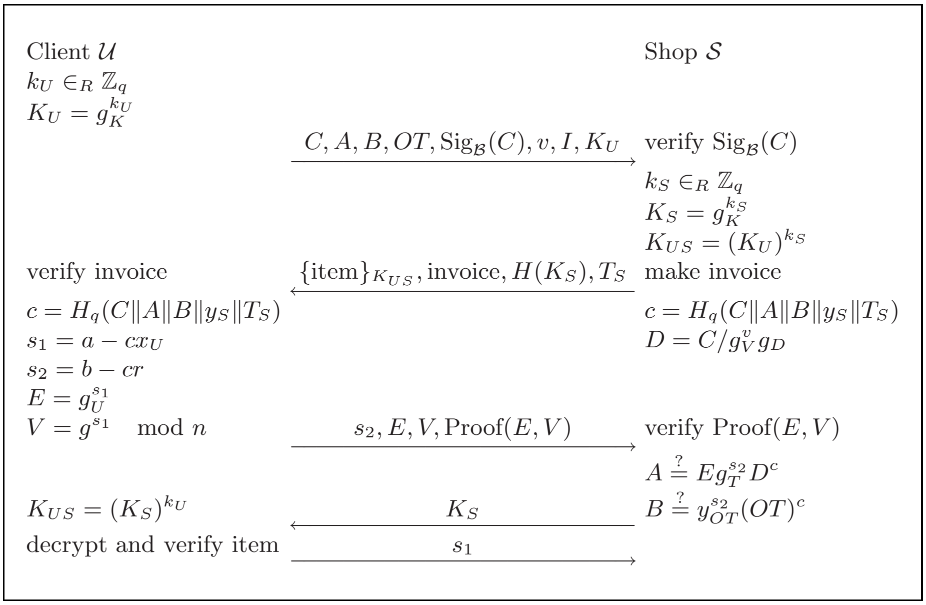
Claim 3. Our way of verifiably encrypting the response is secure and does not effect the security of the rest of the protocol execution.

4.2 Payment Protocol Preserving Payment Atomicity

We assume that $(n = PQ, g)$ are TTP's public key of the NS cryptosystem such that $\sigma > q$, where σ is one of the private key of the TTP. Our proposed payment protocol is depicted in Fig. 6. In this protocol, $H : \{0,1\}^* \to \{0,1\}^l$ denotes a collision-resistant hash function different to H_q. The shop uses $x_S = \log_{g_S} y_S$ as the signing key when making invoices, where g_S is a generator of G_q. We assume Schnorr signature scheme [14] is used for this purpose. The differences between our method and Xu et al.'s method can be summarized as follows.

Difference 1. *We use invoices which include among others the digest of the shop's share of DH key, the client's share of DH key, and the digest of the encrypted item.*
An invoice is generated by the shop by signing necessary information with its private key. This invoice serves as a proof that the payment has actually occurred. This invoice is used only to settle disputes. If we removed this invoice, there is nothing in the payment that a client cannot do by him/herself. As a result, without it, the TTP cannot know whether the payment transcript is genuine or a faked one. In other words, the TTP always have to interact with the bank to see if the given transcript has been deposited in the bank. By using invoices, we can remove the need for the TTP to contact the bank and clarify dispute settlement. Since the shares of DH key are included in the invoice, each party cannot present a different share other than the one included in it. Since the digest of the encrypted item is included in the invoice, disputes arising from a shop accidentally or deliberately sending a wrong item can be resolved without any problem. Xu et al. did not cover this kind of dispute.

Difference 2. *The response s_1 is verifiably encrypted.*
Since the response is sent verifiably encrypted, the shop is convinced that it can present V to the TTP and get s_1 needed for it to deposit the received coin. As a result, it is safe for the shop to terminate before the client does.



$$\begin{array}{lcl}
\text{Client } \mathcal{U} & & \text{Shop } \mathcal{S} \\
k_U \in_R \mathbb{Z}_q & & \\
K_U = g_K^{k_U} & & \\
 & \xrightarrow{C, A, B, OT, \mathrm{Sig}_{\mathcal{B}}(C), v, I, K_U} & \text{verify } \mathrm{Sig}_{\mathcal{B}}(C) \\
 & & k_S \in_R \mathbb{Z}_q \\
 & & K_S = g_K^{k_S} \\
 & & K_{US} = (K_U)^{k_S} \\
\text{verify invoice} & \xleftarrow{\{\text{item}\}_{K_{US}}, \text{invoice}, H(K_S), T_S} & \text{make invoice} \\
c = H_q(C\|A\|B\|y_S\|T_S) & & c = H_q(C\|A\|B\|y_S\|T_S) \\
s_1 = a - cx_U & & D = C/g_V^v g_D \\
s_2 = b - cr & & \\
E = g_U^{s_1} & & \\
V = g^{s_1} \mod n & \xrightarrow{s_2, E, V, \mathrm{Proof}(E,V)} & \text{verify } \mathrm{Proof}(E,V) \\
 & & A \overset{?}{=} E g_T^{s_2} D^c \\
K_{US} = (K_S)^{k_U} & \xleftarrow{K_S} & B \overset{?}{=} y_{OT}^{s_2}(OT)^c \\
\text{decrypt and verify item} & \xrightarrow{s_1} &
\end{array}$$

invoice $= \mathrm{Sig}_S(H(\{\text{item}\}_{K_{US}})||I||\text{price}||C||y_S||T_S||H(K_S)||K_U)$
Proof$(E, V) = $ ZKProof$(\log_{g_U} E = \log_g V)$

Fig. 6. The Proposed Payment Protocol.

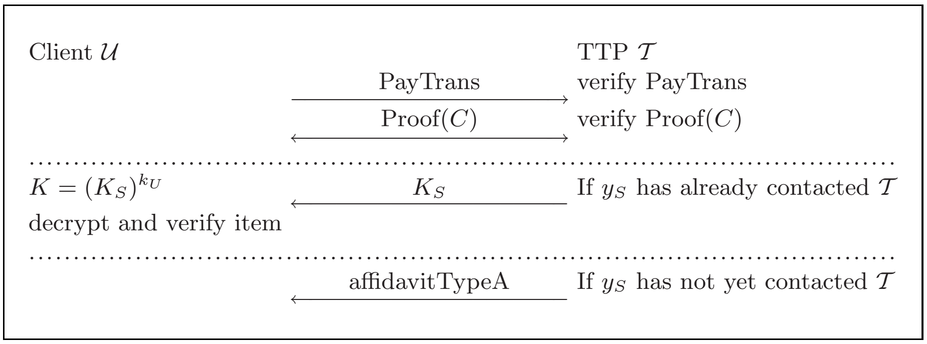
Difference 3. *The shop terminates the protocol first instead of the client.*
This difference results from the Difference 2.

As with Xu et al.'s method, we impose a deposit deadline on shops. Imposing a deadline may be inconvenient for clients, since they must wait until the deadline is over to file a complaint. The shops must also deposit received coins before the deadline. However, it simplifies dispute settlement. Especially, the TTP does not have to interact with anyone other than the one who filed a complaint to resolve disputes. Using a deadline or not may be selected as an option.

4.3 Dispute Settlement Procedure

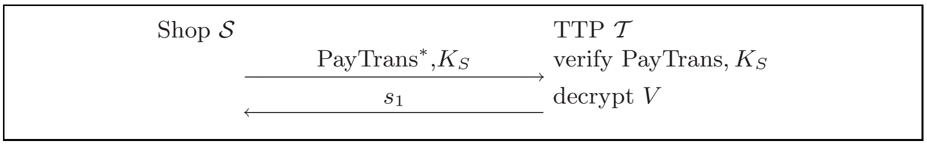
Disputes can occur only after the client has given s_2, E, V, and Proof(E, V). Therefore, we do not think there are other possible disputes other than the following three examples. In all the examples, the TTP stores the received information to log requests.

Dispute Example 1. *The client has given valid E, V, s_2, and* Proof(E, V) *but the shop has refused to give its share of DH key.*
In this case, after the deposit deadline is over, the client can execute the resolve protocol given in Fig. 7 to request the shop's share of DH key. The result of executing the protocol can be divided into four cases shown in Table 1. If the



$$\text{PayTrans} = \{C||A||B||OT||\text{Sig}_{\mathcal{B}}(C)||v||K_U||H(K_S)||y_S||T_S||I||H(\{\text{item}\}_{K_{US}})|| \\ \text{invoice}||s_2||E||V||\text{Proof}(E,V)\}$$
$$\text{Proof}(C) = \text{ZKProof}(C = g_U^{x_U} g_V^{v} g_T^{r} g_D)$$

Fig. 7. The Client's Resolve Protocol A.



* PayTrans is same as PayTrans used in Fig. 7

Fig. 8. The Shop's Resolve Protocol.

client is honest, it means that the client has not given s_1 to the shop. Since the shop cannot decrypt V by itself, the shop must ask the TTP to decrypt V for it to deposit C in the bank. If the client is dishonest, he/she may try to illegally acquire an affidavit even though he/she has received the shop's share of DH key. However, as you can see in the Table 1, the client's refund request would either be rejected or carry out in a normal way. This is to prevent a client from refunding then spending it or vice versa, which would result in double spending.

The resolve protocol given in Fig. 7 is performed as follows. The client sends the payment transcript PayTrans and proves the ownership of C by showing that he/she knows the representation of C. This proof is denoted as ZKProof($C = g_U^{x_U} g_V^{v} g_T^{r} g_D$). The TTP verifies PayTrans and the proof. If everything is confirmed, the TTP sends the shop's share of DH key or an affidavit of type A to the client depending on whether the shop has already contacted the TTP or not. The affidavit of type A is a digital signature issued by the TTP which states that the payment has been cancelled because the client did not receive the shop's share of DH key. The client can refund C at the bank using this affidavit.

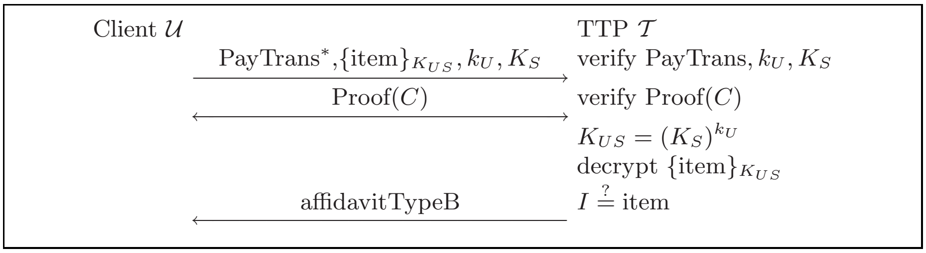
Dispute Example 2. *The shop has given its correct share of DH key, but the client has refused to give s_1.*

Table 1. The Possible Results of Executing the Client's Resolve Protocol A.

	*the client is honest**	
	the shop has contacted the TTP	*the shop has not contacted the TTP*
client obtainings	the shop's share of DH key	affidavit of type A
refund method		normal***
case	case 1-A	case 1-B

	*the client is dishonest***	
	the shop has deposited the coin	*the shop has not deposited the coin*
client obtainings	affidavit of type A	affidavit of type A
refund method	the bank rejects refund request	normal
case	case 1-C	case 1-D

* The client actually did not receive the shop's share of DH key.
** The client actually received the shop's share of DH key.
*** The normal way to refund a coin is to pay the coin to the bank using the normal payment protocol.



* PayTrans is same as PayTrans used in Fig. 7

Fig. 9. The Client's Resolve Protocol B.

In this case, the shop executes the resolve protocol given in Fig. 8. This must be done before the deadline is over. The shop gives the payment transcript PayTrans and its share of DH key. The TTP first verifies the PayTrans and the key. If everything is confirmed, it decrypts V and sends s_1 to the shop. However, if the decrypted value s_1 does not satisfy $A \overset{?}{=} g_U^{s_1} g_T^{s_2} D^c$, the TTP initiates the tracing mechanism to find the client responsible for this wrong doing. We must note that the TTP, that resolves disputes, must be different to the TTP, that has the tracing capabilities, in order to preserve the client's anonymity.

Dispute Example 3. *The received item is not the one requested by the client.* Unlike Example 1, since we use digitally signed invoices, the client cannot deceive the TTP in this case. When a client receives a wrong item, he/she can execute the resolve protocol given in Fig. 9 to report it. This protocol is performed as follows. The client sends the PayTrans along with the k_U, K_S, and $\{\text{item}\}_{K_{US}}$ to the TTP. Here, k_U must be protected from eavesdroppers, since it can allow them to decrypt the item. The TTP first verifies PayTrans. It then verify K_S

Table 2. The Possible Results of Presenting an Affidavit of Type B to the Bank.

	the shop has deposited the coin	*the shop has not deposited the coin*
refund method	anonymously refresh*	normal
case	case 3-A	case 3-B

* Refreshing a coin means exchanging the coin with a new coin.

using $H(K_S)$ and k_U using K_U. The TTP then computes K_{US} and decrypts the encrypted item and verifies whether or not the item is the one identified by I. This cannot be done electronically. If the shop has sent a wrong item, the TTP issues an affidavit of type B which states that the shop has sent a wrong item and the client can refund his/her money. If a client contacts the bank and provides an affidavit of type B, the bank checks whether the shop has deposited the involved payment or not. The possible results of presenting an affidavit of type B to the bank are shown in the Table 2. Unlike Example 1, if the shop has deposited it, the bank anonymously refreshes the coin for the client and takes necessary actions to get the money back from the shop. The protocol used to refresh a coin anonymously is omitted in this paper due to space limitation. The bank will also keep the payment transcript in the deposit database just in case the client re-spends C again. As with Example 1, if the shop has not deposited it, the bank will ask the client to refund it in the normal way.

5 System Analysis

In this section, we will discuss about why our proposed method preserves payment atomicity.

5.1 Atomicity

Claim 4. A client cannot obtain an illegal profit.

Proof Sketch. Once the client receives $\{\text{item}\}_{K_{US}}$ and $H(K_S)$ from the shop, the client can acquire the item if he/she can compute K_S from $H(K_S)$. Since H is a one-way hash function, this is computationally infeasible. After receiving K_S from the shop, the client can terminate the protocol without giving s_1 to the shop. However, since the shop already has V, the shop can execute the resolve protocol given in Fig. 8 to get s_1 from the TTP. Although, a client can send an invalid V to the shop during the payment, by the Claim 1, the client cannot obtain an illegal profit by sending a false V. A client can acquire an affidavit of type A or type B by performing resolve protocols with the TTP. The client can only anonymously refresh a coin with an affidavit of type B. Furthermore, this is only possible if the shop has deposited the payment. As a result, spending then refunding or vice versa will be detected as before. Therefore, a client cannot obtain an illegal profit.

Claim 5. A shop cannot obtain an illegal profit.

Proof Sketch. Until the shop receives valid responses from the client, the shop cannot deposit the received coin. By Assumption 1, it is computationally infeasible for the shop to decrypt V by itself. Moreover, by the DL assumption, the shop cannot compute s_1 from E either. Once the shop receives V, the shop can ask the TTP to decrypt it without giving its share of DH key to the client. The shop, however, has to give the correct share to the TTP when requesting this decryption. Therefore, in this case, the client can execute the resolve protocol given in Fig. 7 to obtain the shop's share of DH key. The shop may send a garbage data or a wrong item to the client instead of the requested item. However, since the digest of the encrypted item and the shares of DH key are included in the invoice, the shop cannot falsely deny its misconduct. Therefore, a shop cannot obtain an illegal profit.

Claim 6. Our proposed method provides payment atomicity.

Proof Sketch. There are only four cases to be considered.

- case 1: Everyone succeeds. This means that atomicity is not violated.
- case 2: The shop obtained an illegal profit. By Claim 5, this cannot happen.
- case 3: The client obtained an illegal profit. By Claim 4, this cannot happen.
- case 4: Both received nothing. This also means that payment atomicity is not violated.

From case 1 to 4, we can conclude that our method provides payment atomicity.

5.2 Security

Claim 7. Our proposed method preserves client's anonymity.

Proof Sketch. We will assume that the basic payment protocol preserves client's anonymity. It is easy to see that additional steps (E, V, $\mathrm{Proof}(E,V)$) taken cannot be used to identify the owner of C. In client's resolve protocol A and B, the client proves the ownership of C using a proof of knowledge of a representation. This proof does not also reveal the identity of the owner of C. A client may need to refund the spent coin as a result of a dispute. We have also provided the ways to preserve a client's anonymity in this case. Therefore, a client can remain anonymous even if disputes arise.

It is obvious that the TTP participates only when disputes arise. The shop sends an item encrypted using DH key. Therefore, a bystander cannot acquire the item without breaking the DH assumption. Moreover, even the TTP cannot acquire the item. If disputes of the kind given in Dispute Example 3 occur, the TTP receives enough information to compute the key K_{US}. However, decryption would only result in garbage data or a wrong item.

6 Conclusion

In this paper, we have proposed a new method of providing the payment atomicity in offline e-cash system based on the representation problem. For this purpose, we have devised a verifiable encryption scheme that uses the NS cryptosystem and a proof of equality of DLs from different groups. Although using this verifiable encryption requires additional proof and messages during payments, the extra cost is reasonable in that we have removed the interval proof from equality proof from different groups. This way of providing the payment atomicity can be applied to any offline e-cash system based on the representation problem. Moreover, it can be applied to systems that uses challenge-and-response where response is verified by raising itself to a power. Furthermore, we have improved the efficiency of dispute settlement significantly. In our method, the TTP does not have to interact with any other party other than the one who filed a complaint to resolve disputes.

Although, we think that the additional cost of our system is reasonable, a more efficient verifiable encryption scheme would further enhance the system performance. In this system, we have assumed that a coin is exchanged for a single digital item. In the future, we will consider how payment atomicity can be preserved when several coins are involved or when several items are exchanged for a single payment.

References

1. Tygar, J.D.: Atomicity in Electronic Commerce. In: Proc. of the 15th ACM Symp. on PODC. ACM Press (1996) 8–26
2. Boyd, C., Foo, E.: Off-line Fair Payment Protocols using Convertible Signatures. In: Ohta, K., Pei, D. (eds.): Advances in Cryptology, Asiacrypt 1998. Lecture Notes in Computer Science, Vol. 1514. Springer-Verlag (1998) 271–285
3. Xu, S., Yung, M., Zhang, G., Zhu, H.: Money Conservation via Atomicity in Fair Off-Line E-Cash. In: Lomas, T.A. (ed.): Proc. of the 2nd Int. Workshop on Information Security. Lecture Notes in Computer Science, Vol. 1189, Springer-Verlag (1997) 14–31
4. Asokan, N., Schunter, M., Waidner, M.: Optimistic Protocols for Fair Exchange. In: Proc. of the 4th ACM Conf. on CCS. ACM Press (1997) 6–17
5. Ateniese, G.: Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures. In: Proc. of the 6th Conf. on CCS. ACM Press (1999) 138–146
6. Asokan, N., Shoup, V., Waidner, M.: Optimistic Fair Exchange of Digital Signatures. In: IEEE J. on Selected Areas in Comm. 18(4) (2000) 593–610
7. Brands, S.: Untraceable Off-line Cash in Wallets with Observers. In: Stinson, D.R. (ed.): Advances in Cryptology, Crypto 1993. Lecture Notes in Computer Science, Vol. 773, Springer-Verlag (1994) 302–318
8. Naccache, D., Stern, J.: A New Public Key Cryptosystem Based on Higher Residues. In: Proc. of the 5th ACM Conf. on CCS. ACM Press (1998) 59–66
9. Camenish, J., Michel, M.: Separability and Efficiency for Generic Group Signature Schemes. In: Wiener, M.J. (ed.): Advances in Cryptology, Crypto 1999. Lecture Notes in Computer Science, Vol. 1666, Springer-Verlag (1999) 106–121

10. Solages, A., Traore, J.: An Efficient Fair Off-Line Electronic Cash System with Extensions to Checks and Wallet with Observer. In: Hirschfeld, R. (ed.): Proc. of the 2nd Int. Conf. on Financial Cryptography, Lecture Notes in Computer Science, Vol. 1465, Springer-Verlag (1998) 275–295
11. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Trans. on Information Theory. 22(6) (1976) 644–654
12. Kim, S., Oh, H.: A New Electronic Check System with Reusable Refunds. Int. J. of Information Security. 1(3) (2002) 175–188
13. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.): Advances in Cryptology, Crypto 1992. Lecture Notes in Computer Science, Vol. 740, Springer-Verlag (1993) 89–105
14. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. J. of Cryptology 4(3) (1991) 161–174

A Limited-Used Key Generation Scheme for Internet Transactions

Supakorn Kungpisdan[1], Phu Dung Le[2], and Bala Srinivasan[1]

[1] School of Computer Science and Software Engineering, Monash University
900 Dandenong Road, Caulfield East, Victoria 3145 Australia
{Supakorn.Kungpisdan,Bala.Srinivasan}@infotech.monash.edu.au
[2] School of Network Computing, Monash University
McMahons Road, Frankston, Victoria 3199 Australia
Phu.Dung.Le@infotech.monash.edu.au

Abstract. Traditionally, the security of symmetric-key based systems heavily relies on the security of shared keys. In this paper, we present a new session key generation technique for internet transactions that eliminates the need of storing long-term shared key which makes the system insecure against key compromise during transactions. The generation of each set of session keys is based on randomly chosen preference keys. The higher number the transactions have been performed, the less chance the system is being compromised. We show that the proposed technique is secure against various kinds of attacks. Finally, the proposed technique can be applied to any kind of internet applications that deploy shared secrets. We demonstrate the practical usefulness of our technique by applying it to credit-card payment systems. The results show that our technique enhance their security considerably.

Keywords: credit-card payment, key generation, electronic payment systems

1 Introduction

Nowadays, several internet services such as file transfer, web services, or viewing electronic articles, deploy shared secrets. In a shared-key based system, a user and the system (or another user) share a secret key that can be used for several purposes:

– **Credential or Authentication Token:** the key can be used to authenticate the user to the system e.g. access password. In some applications such as credit-card payment, a cardholder (or a client) sends her credit-card number including payment-related information to a merchant over a secure channel, such as SSL [1], in order to request for a payment to the merchant. The merchant then forwards such information to a card issuer to authorize the payment. As the credit-card number is shared between the cardholder and the card issuer, the card issuer can verify that the cardholder's request is valid. Then, the card issuer deducts the money from the cardholder's account

and transfers to the merchant's account. In the rest of the paper, the term *client* and *issuer* refer to the cardholder and the card issuer, respectively.
- **Cryptographic Operations:** the shared key can be used as the key for encrypting or hashing a message sent between users. For example, Alice sends a message, encrypted with a shared key between herself and Bob, to Bob securely over an open network. If Bob did not previously generate this message, he can infer that this message has been originated by Alice.

Among the applications of shared keys stated above, on one hand, the access password needs to be updated periodically, and more frequently for encrypting keys in order to enhance security of the system. However, the higher frequency the keys are updated, the lower performance the system will be. On the other hand, the credit-card number which is static, reusable information transferred in every transaction is possible to be eavesdropped by an attacker.

In this paper, we present a new key generation technique which is able to solve the problems stated above. We select credit-card payment as a reference scenario to illustrate our idea, yet our technique can be applied to other kinds of internet applications.

1.1 Credit-Card Payments over the Internet

In a credit-card payment system, a client who opens an account with a bank wants to make a purchase to a merchant over the Internet. The client is issued a credit card to make a purchase physically in shops or over the Internet.

Various kinds of credit-card payment systems have been implemented [1, 6]. In credit-card payment over SSL [1], after the SSL connection is established, the client supplies her semi-secret 16-digit credit-card number and relevant information, such as date of birth and billing address, to authenticate herself to an issuer. In SET (Secure Electronic Transaction) protocol [6], the credit-card information is encrypted with the public key of a payment gateway (a party who acts on behalf of the issuer) and signed with the client's private key. Once it is transferred to the issuer, the issuer can infer that this request is originated by the client and it contains the valid credit-card information.

It can be seen that the most sensitive information in credit-card payment systems is credit-card information. Several security issues related to exposing credit-card information have been reported [7, 4]. In an SSL-based credit-card payment system, although the credit-card information is not exposed to external parties, it is still revealed to the merchant who is considered as an untrusted party. In SET protocol [6], encrypted credit-card information is decrypted by the payment gateway and then forwarded to the issuer. This problem may arise if the payment gateway and the issuer are different parties.

Moreover, the credit-card number is considered as long-term, reusable, semi-secret information. It is printed on the card which is visible to everyone, and the client's information such as date of birth and billing address is not difficult to find out. Although the credit-card number is replaced by a secret known only between the client and the issuer, it still has to be transferred in every transaction. Therefore, it is vulnerable to various kinds of attacks.

Several techniques have been proposed to secure credit-card information transfer over the Internet [4,7,5,2]. Credit-card number blinding technique was proposed [4] by applying HMAC to the credit-card number and a random number. Then both the output of HMAC and the random number are sent to the issuer for verification. Recently, the concepts of disposable credit-card numbers (DCNs in short) have been proposed in online scheme [8] and offline schemes [7,5,2]. These techniques allow a client to perform transactions with fresh credit-card number in every transaction.

In this paper, we focus our consideration on offline DCN generation techniques because they have advantages over online techniques. In particular, the offline techniques do not require any connection between the client and the issuer in order to generate a DCN in every transaction, whereas the online ones do. The offline techniques therefore do not require any secure channel established between the clients and the issuer.

Several offline DCN generation techniques have been proposed. In Rubin *et al.*'s scheme [7], each DCN is generated from the encryption of payment information with a long-term key shared between the client and the issuer. In Li *et al.*'s scheme [5], a new DCN is generated from the hash of currently-used DCN and a long-term key shared between the client and the issuer. Recently, Kungpisdan *et al.* [2] proposed a disposable key generation technique based on hashing of bit-shifting of long-term shared key. The keys generated from this technique can be used either as DCNs or the keys for encryption and MAC (Message Authentication Code).

However, the above schemes require long-term secret keys shared between the client and the issuer in order to calculate DCNs. Although such keys are not transferred during transactions, they are possible to be compromised by one who intercepts DCNs and successfully deciphers it.

In this paper, we propose a technique to generate limited-use keys that does not rely on any long-term shared key. The higher number the keys have been used, the less chance the attacker can compromise the system. Our limited-use secret keys can be used as single used credentials, such as DCNs, or the keys for cryptographic operations, such as encryptions or keyed-hash functions.

This paper is organized as follows. Section 2 outlines existing offline DCN generation techniques and discusses their security issues. Section 3 introduces the proposed technique. In section 4, we apply the proposed technique to enhance security of internet payment systems. Section 5 discusses about the security of the proposed technique. Section 6 concludes our work.

2 Related Work

In this section, we outline three existing limited-use key generation techniques. Section 2.1 presents Rubin *et al.*'s approach. In section 2.2, Li *et al.*'s approach is outlined. In section 2.3, Kungpisdan *et al.*'s technique is discussed.

2.1 Rubin *et al.*'s Approach

Rubin *et al.* proposed an offline DCN generation technique [7]. This method eliminates the need of traditional long-term, reusable credit-card numbers. In this technique, a DCN (in [7], it is called a *token*) is generated from the encryption of a set of payment-related information (called *restrictions*) that contains payment amount, merchant's identity, billing address, etc., with a long-term shared key between a client and her issuer. For example, Alice wants to buy a 50-dollar book from Bob's store. She can generate the token T as follows:

$$T = \{\textit{fifty-dollars-book-Bob's-store}\}_K$$

where K is the long-term key shared between Alice and her issuer. On receiving T, the issuer decrypts it using K. This technique also deploys timestamp for replay and collision protection. Note that the collision may occur when two different payment information is encrypted either with the same key or with different keys. Although, Rubin *et al.* argued that the system is secure against various kinds of guessing attacks. To some degree, the encryption with long-term shared key is vulnerable if an attacker has enough information and attempts to decrypt DCN. The system will fail if the long-term key is compromised. Waiting until the fraud is being detected may be unacceptable to the clients whose credit-card information falls into the wrong hands. Moreover, Li *et al.* [5] argued that the encryption may be computationally expensive when there are many users and restrictions.

2.2 Li *et al.*'s Approach

Li *et al.* proposed a technique to generate DCNs based on one-way hash function for smartcard-based environment [5]. Initially, a smartcard-based credit card is issued to a client. The information stored in the credit card is composed of the semi-secret, 16-digit credit-card number CCN, the long-term key S, and initial session DCN, T_{init}. CCN also appears on the card. These secrets are known only to the client and the issuer.

In the first transaction, the client sends T_{init} to the issuer for payment authorization. In next transactions, the client generates new DCNs T_{new} as follows:

$$T_{new} = h(T_{cur}||S)$$

where T_{cur} stands for the previously used DCN. The client then sends T_{new} to the issuer. On receiving T_{new}, the issuer calculates T_{new} and compares with the one received from the client. It can be seen that the security of the system is based on the length of S and T and the security of hash function. Although it is assumed that the hash function is irreversible, the use of the long-term key S offers the opportunity for an attacker to attempt to compute S from eavesdropping DCNs. Moreover, successful guessing S will compromise the security of the system.

2.3 Kungpisdan *et al.*'s Approach

Kungpisdan *et al.* proposed a session key generation technique as a part of KSL, their mobile credit-card payment protocol. In this paper, we discuss briefly about KSL protocol and focus our consideration on the key generation technique.

There are mainly four parties involved in KSL protocol: a client **C**, a merchant **M**, an issuer **I** (the client's bank who issues a credit card to the client), and an acquirer **A** (the merchant's bank). A payment gateway **PG** acts as a medium between the client/merchant over the Internet side and the issuer/acquirer over banking private network side.

Initially, the client and the issuer share a long-term secret CCI. Also, they share another secret Y. Y needs to be updated periodically or upon requests. The main concept of this technique is to apply a hash function with one-bit cyclic shift (either left shift or right shift) of a master secret Y and CCI each time a session key is generated.

$$Y_i = h(\textit{i-bit-shift-of-}(CCI, Y)), \text{ where } i = 1, \dots, n$$

Besides the secret Y, another secret X is shared between the client and the merchant. As the client and the merchant do not share any long-term secret, the session key generation for X can be done as follows:

$$X_i = h(\textit{i-bit-shift-of-}X), \text{ where } i = 1, \dots, n$$

After the sets of X_i and Y_i are successfully generated, the client can perform a payment transaction by the following protocol:

Step 1	**C→M**:	$InitialRequest$
	M→C:	$\{InitialResponse\}_{X_i}$
Step 2	**C→M**:	$\{OI, MAC[(Price, OI), Y_i]\}_{X_i}, MAC[OI, X_{i+1}]$
Step 3	**M→PG**:	$\{\{MAC[(Price, OI), Y_i], Price\}_{K_{PG}}, h(OI)\}_{K_M^{-1}}$
Step 4	Under private network,	
4.1)	**PG→I**:	$MAC[(Price, OI), Y_i], h(OI), Price, ID_M$
4.2)	**PG→A**:	$Price, ID_M$
4.3)	**I,A→PG**:	$Yes/No, \{h(OI), Yes/No\}_{Y_i}$
Step 5	**PG→M**:	$\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$
Step 6	**M→C**:	$\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$

where OI stands for order information, $Price$ stands for the price of requested goods or services, ID_M stands for the merchant's ID, and Yes/No stands for approval result *Approved/Rejected*. In this paper, we do not present each protocol step in details, but focus only on how the above techniques are applied to the protocol. The details of KSL protocol can be found in [2].

From the above protocol, Y_i is used for two purposes: one as the key for Message Authentication Code (MAC) and the others as the key for encryption. The MAC containing Y_i and $Price$ is transferred from the client in **Step 2** to the issuer in **Step 4.1**. On receiving the message, the issuer verifies Y_i and transfers $Price$ to the merchant. Then, Y_i is used as the key to encrypt payment response transferred from the issuer in **Step 4.3** to the client in **Step 6**.

It can be seen that it is difficult to retrieve CCI. Due to the property of one-way hash function, it is computationally infeasible to run reverse operation to retrieve $\{CCI, Y\}$ from Y_i. Also, applying bit shifting to $\{CCI, Y\}$ provides diffusion to Y_i. In addition, in [2], as Y_i can be used for both encryption and MAC, if the deployed MAC algorithm requires the key with shorter length than one for encryption, the length of the key for keyed-hash function has to be shortened. Thus, it results in more difficult to extract pre-image of Y_i from the intercepted message.

However, as well as Rubin *et al.*'s and Li *et al.*'s schemes, if it happens that CCI is disclosed to an attacker, the security of the system will be compromised because generating each session key Y_i is based on the long-term key CCI.

3 The Proposed Technique

In the previous section, the security of symmetric-key based systems relies heavily on the privacy of long-term shared keys. If the keys are revealed to an attacker, the security of the entire system will be compromised. In this section, we present a limited-use key generation technique that the key used in each session does not rely on any long-term key so that the compromise of the long-term key does not affect the security of the system. Later on, we call the long-term key shared between two parties as "*master key*" and the key used in each session as "*session key*".

Consider a situation where two parties, Alice and Bob, communicate to each other using a shared key K_{AB}. This secret can be either used as an encrypting key, a key for keyed-hash function, or a credential, e.g. $\{X\}_{K_{AB}}$, $h(X, K_{AB})$, or K_{AB}, where X is a message and K_{AB} is a key shared between Alice and Bob, respectively. As the credential, K_{AB} needs to be protected by applying cryptographic operations to it. Practically, K_{AB} should be used only for short period of time before being updated. Assume that the updated key is securely distributed between parties, what we concern is that it would enhance performance of the system if the frequency of key update process can be reduced.

As discussed in section 2, generating the set of session keys from a master key shared between parties is a possible solution. However, the key generation technique must be secure in that it must be difficult for an attacker to compute the master key from capturing the session keys. To illustrate our technique, we assume that Alice and Bob wants to communicate securely. The following initial settings have been made:

3.1 Initial Settings

1. Alice and Bob share the master key K_{AB}. K_{AB} can be assumed to be never expired.
2. The "*distributed key*" DK is another shared key between Alice and Bob that needs to be updated periodically or upon their request. It is distributed by performing an authenticated key-exchange protocol between Alice and Bob.
3. Alice's and Bob's computing devices are not necessary to be secure against attacks. If required, they can be implemented using tamper-proof resistant devices such as smartcard. Moreover, they are capable of performing at least

hash operations. Referred to the analysis by Krawczyk in [4], to guarantee sufficient security, HMAC-supported devices are preferred.

4. $h(M, K)$ stands for the keyed-hash function of the message M and the key K. However, to guarantee higher security, HMAC is preferred.

3.2 Session Key Generation

The following steps show the details of the proposed technique.

1. Alice generates the key DK and distributes to Bob through a secure channel.
2. Alice and Bob generate a set of "*preference keys*" K_i, where $i = 1, \ldots, m$, based on K_{AB} as follows:

$$K_1 = h(DK, K_{AB}), K_2 = h(DK, K_1), \ldots, K_m = h(DK, K_{m-1})$$

   Later on, we name the kind of process as $K_A * K_B$, which denotes the generation of a set of keys from the output of $h(K_A, K_B)$ as described above. After generating K_i, the master key K_{AB} can be removed from the system. The set of K_i is not used in any transaction, but later used as a seed for generating session keys.

3. Alice generates a random number r and sends it to Bob.

$$\textbf{Alice} \rightarrow \textbf{Bob:} \qquad r$$

   Alice selects two preference keys; one is K_{Mid_1}, where K_{Mid_1} is the middle key among $\{K_1, \ldots, K_w\}$, $w = r\ mod\ m$. The others is K_{Mid_2}, where K_{Mid_2} is the middle key among $\{K_1, \ldots, K_{Mid_1}\}$. Then, she calculates "*session initialization key*" SIK, where

$$SIK = h(K_{Mid_1}, K_{Mid_2})$$

   Note that, in general, K_{Mid_1} and K_{Mid_2} do not have to be the middle keys among the set of $\{K_1, \ldots, K_m\}$. It can be selected from some other method which depends on the agreement between Alice and Bob. Moreover, after generating SIK, K_{Mid_1} and K_{Mid_2} are then removed from the system. On receiving r, Bob can also generate SIK.

4. Alice and Bob generate a set of session keys SK_j, where $j = 1, \ldots, n$.

$$SK_1 = h(SIK, DK), SK_2 = h(SIK, SK_1), \ldots, SK_n = h(SIK, SK_{n-1})$$

Fig. 1 depicts the proposed session key generation technique. After the set of SK_j has been generated, Alice and Bob can make use of them in several ways. Firstly, Alice can send SK_j as a credential to authenticate herself or authorize some actions to Bob over a secure channel, e.g. SSL channel or the encryption with Bob's public key. Secondly, SK_j can be used as an encrypting key in that Alice encrypts a message with SK_j and sends it to Bob. Alternatively, SK_j can be used as a key for keyed-hash function. On receiving the message, Bob can verify the message by using the set of SK_j generated by himself and compare whether or not the received SK_j matches the one he has. If they are matched, it means that Alice has sent the valid SK_j to Bob. The applications of the proposed technique to internet applications will be presented in section 4.

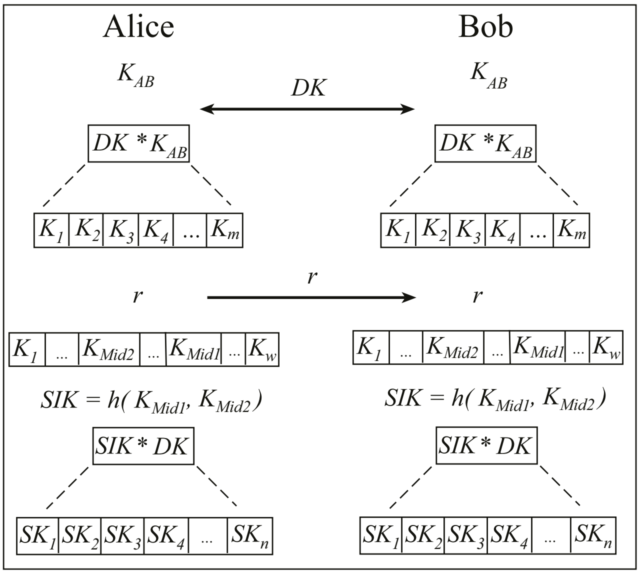

Alice
Bob
K_{AB}
DK
K_{AB}
$DK * K_{AB}$
$DK * K_{AB}$
K_1 K_2 K_3 K_4 ... K_m
K_1 K_2 K_3 K_4 ... K_m
r
r
r
K_1 ... K_{Mid2} ... K_{Mid1} ... K_w
K_1 ... K_{Mid2} ... K_{Mid1} ... K_w
$SIK = h(K_{Mid1}, K_{Mid2})$
$SIK = h(K_{Mid1}, K_{Mid2})$
$SIK * DK$
$SIK * DK$
SK_1 SK_2 SK_3 SK_4 ... SK_n
SK_1 SK_2 SK_3 SK_4 ... SK_n

Fig. 1. Session key generation.

3.3 Session Key Update

After being used for transactions for specified period, the set of SK_i needs to be updated. The session key update can be performed as follows:

1. Suppose that, both Alice and Bob have used SK_j up to SK_p, where $1 \leq p \leq n$. Either Alice or Bob selects two preference keys; one is K'_{Mid_1}, where K'_{Mid_1} is the middle key among $\{K_1, \ldots, K_q\}$, $q = p \ mod \ rm$, rm is the number of remaining members in the set of K_i. The others is K'_{Mid_2}, where K'_{Mid_2} is the middle key among $\{K_1, \ldots, K'_{Mid_1}\}$. Then, they generate a new session initialization key SIK' as follows:

$$SIK' = h(K'_{Mid_1}, K'_{Mid_2})$$

   After generating SIK', K'_{Mid_1} and K'_{Mid_2} are removed from the system.
2. Alice and Bob generate a new set of session keys SK'_j, where $j = 1, \ldots, n$.

$$SK'_1 = h(SIK', DK), SK'_2 = h(SIK', SK'_1), \ldots, SK'_n = h(SIK', SK'_{n-1})$$

   Note that the distributed key DK does not need to be updated every time a new set of session keys SK_j is generated. Updating DK also results in updating the set of K_i. Alice and Bob can repeatedly use DK until the set of K_i is depleted. Fig. 2 depicts the proposed session key update technique.
3. To update a new set of distributed keys, after the new key distributed key DK' has been distributed, the following process can be done. Assume that the currently-used session key is SK_u, Alice and Bob selects K_v among the set of $\{K_1, \ldots, K_{rm}\}$, where $v = u \ mod \ rm$. Then, they compute $K_v * DK'$

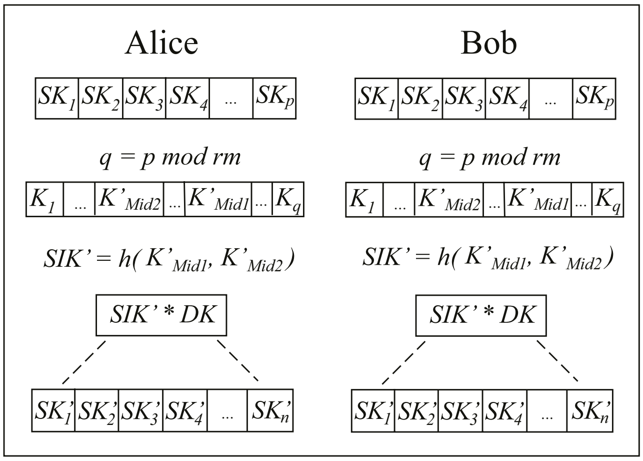

Alice
Bob
SK_1 SK_2 SK_3 SK_4 ... SK_p
$q = p \bmod rm$
K_1 ... K'_{Mid2} ... K'_{Mid1} ... K_q
$SIK' = h(K'_{Mid1}, K'_{Mid2})$
$SIK' * DK$
SK'_1 SK'_2 SK'_3 SK'_4 ... SK'_n

Fig. 2. Session key update.

to generate a new set of preference keys K'_i, where $i = 1, \ldots, n$. Alice and Bob can follow the key updating process to generate a new SI key and a new set of session keys.

It can be seen that generating each set of session keys does not rely on any long-term key, but on randomly chosen preference keys (as a result of selecting the p-th session as the last session before SK_j is being updated).

As a result, the proposed technique not only enhances the security of the system against key compromise, but also reduces the frequency of key update process that increases performance of the system.

4 Applying the Proposed Technique to Internet Payment Systems

In this section, we demonstrate the usability of the proposed technique by applying it to various kinds of internet payment systems.

4.1 KSL Protocol

In this section, we show how the proposed technique can enhance the security of KSL protocol [2]. It can be used as an example to illustrate the practical usefulness of our technique to symmetric-key based cryptographic protocol.

The proposed technique presented in section 3 can be applied directly to KSL protocol due to the similar environment settings. The following illustrates how to generate the sets of Y_i and X_i, where $i = 1, \ldots, n$.

Generating Y_i. In KSL protocol, a client needs to register herself to an issuer in order to share the distributed key Y. Applied the proposed technique, the master key CCI is also distributed during the registration. The following steps can be done in order to generate the set of session keys Y_i.

1. Both client and issuer generate a set of preference keys K_i, where $i = 1, \ldots, m$, by computing $Y * CCI$. Then they store the set of K_i on their terminals. At this stage, CCI is removed from the system.
2. The client generates a random number r and transmits it to the merchant in **Step 1**. r will then be forwarded to the issuer in **Step 4.1**. **Step 1** and **Step 4.1** of KSL protocol are now modified as follows:

   Step 1 C→M: $ID_C, r, s, TIDReq, MIDReq$
   Step 4.1 PG→I: $MAC[(Price, h(OI), ID_M), Y_i], h(OI), r, TID, Price, ID_C, ID_M$

   Note that, in **Step 1**, s stands for the random number generated by the client for generating X_i (its details will be presented in the next section). Alternatively, r can be generated by either the client or the issuer during the registration.
3. The client and the issuer select the keys K_{Mid_1} and K_{Mid_2} and computes SIK, where $SIK = h(K_{Mid_1}, K_{Mid_2})$.
4. The client and issuer generate the set of session keys Y_i, where $i = 1, \ldots, n$, by following $SIK * Y$ and use them in the transactions.

$$Y_1 = h(SIK, Y), Y_2 = h(SIK, Y_1), Y_3 = h(SIK, Y_2), \ldots, Y_n = h(SIK, Y_{n-1})$$

In KSL protocol, Y_i is first used as a MAC key (representing a DCN) to authenticate the client to the issuer. On receiving the message in **Step 4.1**, the issuer can verify the message with the session key SK_j it has. If the verification is successful, the issuer can infer the client with valid credit-card information. In **Step 6**, the client can verify that the message has been sent from the issuer by decrypting the message with the SK_j she has.

Generating X_i. The key generation for X_i presented in section 2.3 seems to be more vulnerable than the technique for generating Y_i since generating each session key X_i relies on only the distributed key X, whereas generating Y_i is based on two keys: Y and CCI and the master key CCI is never transmitted during transactions. Based on the proposed technique, X_i can be generated as follows.

1. After the client and the merchant share the distributed key X, they generate a set of preference keys $\{KK_1, \ldots, KK_m\}$ as follows.

   $$KK_1 = h(X, X_{m-bit-shift}), KK_2 = h(X, KK_1), \ldots, KK_m = h(X, KK_{m-1})$$

   where $X_{m-bit-shift}$ stands for m-bit cyclic shifting of X. The set of KK_i is stored at both client and merchant's devices.
2. The client generates a random number s and sends it to the merchant in **Step 1**. Both client and merchant then select two preference keys; one is KK_{Mid_1}, where KK_{Mid_1} is the middle key among $\{KK_1, \ldots, KK_x\}$, $x = s \ mod \ rm$, rm is the number of remaining members in the set of KK_i. The others is KK_{Mid_2}, where KK_{Mid_2} is the middle key among $\{KK_1, \ldots, KK_{Mid_1}\}$. Then, they calculate SIK', where $SIK' = h(KK_{Mid_1}, KK_{Mid_2})$.

3. The client and the merchant generate X_i, where $i = 1, \ldots, n$, by following $SIK' * X$ and use them for transactions.

$$X_1 = h(SIK', X), X_2 = h(SIK', X_1), \ldots, X_n = h(SIK', X_{n-1})$$

Updating the sets of Y_i and X_i follows the process presented in section 3.3.

4.2 Credit-Card Payment over SSL

As discussed in section 2, several techniques [7, 5] have been proposed to solve the problems of credit-card payments over an SSL that mainly come from the security of the SSL protocol itself and revealing the client's credit-card number to the merchant as it is transferred in cleartext over SSL channel. However, they are still based on long-term shared keys that is vulnerable to attacks.

Applying our technique to credit-card payment over SSL is more straightforward compared to KSL protocol. This is because, after the SSL connection between a client and a merchant is established, payment-related information is then transferred through it. What the client needs to do is transmitting a session key representing the client's credit-card information through the SSL channel.

Referred to section 3, three secrets are shared between the client and the issuer: the master key K_{CI}, the distributed key DK, and the semi-secret credit-card number CCN.

After the client selects goods or services from the merchant's site, the payment screen is displayed. The client is then required to fill in her credit-card number and password to authenticate herself to her device. When the authentication is successful, the session key SK_j is generated and transmitted to the merchant together with payment information under SSL channel as follows:

$$\mathbf{C} \rightarrow \mathbf{M}: \quad \{OD, Price, ID_I, r, h(Price, SK_j, CCN, r)\}_K$$

where K stands for the key shared between the client and the merchant generated from SSL handshake session. $Price$ and OD stand for the client's requested price and goods descriptions (including transaction ID and timestamp), respectively. ID_I stands for the issuer's ID which normally is the first four digits of CCN. Note that r needs to be sent only in the first transaction in the system.

On receiving the message, the merchant can retrieve only OD, ID_I, and $Price$. She recognizes the issuer from ID_I. The merchant then forwards $\{Price, r, h(Price, SK_j, CCN, r)\}$ to the issuer. The issuer verifies $h(Price, SK_j, CCN, r)$ and then sends approval result to the merchant and the client.

It can be seen that any party including the merchant cannot retrieve the master secret K_{CI} or even CCN. Applying the hash function to SK_j together with $Price$ and CCN, not only the merchant cannot retrieve the session key SK_j, but the issuer is also guaranteed that the merchant cannot modify the requested price $Price$.

4.3 SET Protocol

The technique presented in section 4.2 can be exploited in SET protocol [6]. In SET, a party called *payment gateway* performs transactions on behalf of a client/merchant to an issuer/acquirer. The encrypted client's *Payment Information* (PI) including credit-card information is decrypted by the payment gateway and later forwarded to the issuer under a banking private network. In fact, the credit-card information should be known only to the issuer because, in practical, the payment gateway and the issuer may be different parties.

Based on the fact that the purpose of transferring credit-card information is to authenticate the client to the issuer, it is therefore not necessary to reveal the credit-card information to any other party including the payment gateway (if the payment gateway and the issuer are different parties). Applying our technique can conceal the credit-card information from the payment gateway. In SET protocol, the original client's PI is shown as follows:

$$\{TID, h(OD, Price), Price, ID_M, CCI\}_{K_{PG}}$$

where K_{PG} stands for the payment gateway's public key, TID stands for transaction ID including the time and date of transaction, OD stands for order descriptions, and ID_M stands for the merchant's ID. Applied the proposed technique, PI can be modified as follows:

$$\{TID, h(OD, Price), Price, ID_M, SK_j, r\}_{K_{PG}}$$

Note that r has to be transmitted only in the first transaction in the system. With this method, the payment gateway only recognizes that the client has requested it to authorize the issuer for this payment, but SK_j is transparent to it.

4.4 Implementation Issues

In this section, we discuss major issues related to the implementation of the proposed technique in credit-card payment systems. The results from the discussion can also be applied to other kinds of internet applications.

Key Distribution: our technique focuses on the deployment of the master key rather than the semi-secret credit-card number. Therefore, $\{K_{CI}, DK, r\}$ needs to be distributed between the client and the issuer.

Before making the first payment, the client needs to register herself to the issuer in order to share $\{K_{CI}, DK, r\}$. In the case that the client is provided a card reader, the issuer can generate $\{K_{CI}, DK, r\}$ and store them on the card before issuing the card to the client. Therefore r and DK do not need to be transferred in any transaction which results in fully offline key generation.

Storing and Managing Keys: the proposed technique requires two sets of keys, K_i and SK_j, to be generated at each party's device. However, only the entire set of preference keys K_i is stored on the devices, whereas each member in the set of session keys SK_j is generated at the beginning of each transaction in order to reduce storage requirement. As K_i is not transmitted in any transaction, we can guarantee its security as long as the device is not compromised.

After generating the set of K_i, the master key (CCI in KSL protocol and K_{CI} in SSL-based credit-card payment scheme and SET protocol) is no longer used in the system. As well as the session key (Y_i or SK_j), after a new session key has been generated, the previously used key is then removed from the system.

5 Discussion

5.1 Security of the Proposed Technique Against Attacks

In the proposed technique, generating each session key is not based only on the master key. Thus, the compromise of the master key in next transactions will not compromise the security of the system. In this section, we discuss about the security of the proposed technique against various kinds of attacks.

Consider the situation where a session key SK_j is transmitted in cleartext over an insecure channel. First of all, due to the one-way property of hash function, reverse operation of SK_j to retrieve SIK is computationally infeasible. Referred to [4], HMAC algorithm is preferred due to its proven security. One possibility is to collect a number of session keys and try to guess the next value of SK_j. However, the fraud will be detected and limited by allowing a limited number of attempts on specific client. If the attempts exceed the specified limit, the client's account is suspended. The system then notifies the client that there were unauthorized attempts on her account and asks the client to update the SI key. After the client updates the SI key, the attacker has to repeat all the attacking processes from the beginning. Moreover, as stated that SK_j can be used as a key for encryption and keyed-hash function, its key length should be 128 bits which is compatible with AES and HMAC-MD5 algorithms. Therefore, the search space to retrieve the correct SK_j is 2^{128}.

Consider the case that the above attempts have not been detected by the system. The set of session keys SK_j is valid over the short period of time. After the session keys are updated, the keys in the attacker's hands are no longer valid.

As the initial input for the set of SK_j are SIK and DK. The attacker must be able to record all session keys from SK_1 and and then tries to compute SIK and DK. In the case that the attempt is successful, the attacker can generate the next session key. However, the fraud will be eventually detected because the session key generated by the client is found as used by the issuer. At this stage, the client does not need to update the master key. What she needs to do is requesting the issuer to update the set of session keys. After the new set of session keys SK'_j becomes valid, the current SI key SIK is no longer valid because SIK' is based on the preference keys that are not used in any transaction, not SIK. To retrieve SIK', the attacker needs to capture the transaction with SK'_1 and attempts to compute SIK'.

The only possible successful attack to the proposed technique is that the attacker must be able to do the following:

1. Capture DK distributed over the network,
2. Access to each party's device to retrieve the entire set of preference keys K_i,
3. Record all SK_j transmitted in all transactions, and
4. Detect the request to update the set of session keys so that he can know p in order to select K_{Mid_1} and K_{Mid_2} from the set of K_i.

In the worst case, if the attacker succeeds the above process, he can generate and use the valid SK_j until being detected by the system.

We can see that the proposed key generation technique is not based on any long-term secret. Generating each set of session keys is based on dynamic parameters randomly chosen from the set of preference keys K_i. As a result, the higher number the transactions have been performed, the less chance the system is compromised. Session key compromise does not affect the system considerably.

5.2 Possible Applications to Conventional Credit-Card Payments

One of the concerns about the usage of disposable credit-card numbers that has been pointed out in [7] is the compatibility to conventional electronic commerce sites which accept 16-digit credit-card number. Thus, the length of each DCN must be 128 bits and readable by the client. Alphanumeric numbers should be the proper format. In the proposed technique, the length of each session key is equal to the length of the key for AES encryption algorithm or for HMAC-MD5 algorithm so that the session key can be applied to cryptographic operations as stated in section 3.2.

However, as stated in [7], if the keys need to be alphanumeric, the search space to generate the right key is 36^{11} (the first 4-digit number is the card issuer's ID and the last digit is checksum). With the security of our technique discussed in section 5.1, the proposed technique is still secure with this key length.

6 Conclusion

In this paper, we have pointed out the problems and limitations of the deployment of shared secret keys by focusing on credit-card information transfer in credit-card payment systems. We then proposed an efficient technique to generate and update limited-use secret keys. We have shown that the proposed technique is secure against key compromise. Also, the generation of each set of session keys are based on dynamically chosen preference keys so that the system is still secure although some keys are compromised.

We have demonstrated the usability of the proposed technique by applying it to various kinds of credit-card payment systems: KSL protocol [2], SSL-based payment scheme [1], and SET protocol [6]. The results have shown that the proposed technique is advantageous not only to credit-card payment systems, but also to other kinds of internet applications that deploys shared keys. It enhances the security and performance of the system considerably. Moreover, the DCNs generated from the proposed technique are possibly implemented in conventional credit-card payment applications with higher security.

As future work, we aim to analyze performance of the system that applies the proposed technique. Moreover, we aim to apply the proposed technique to other kinds of internet applications, especially mobile commerce.

References

1. Freier, A.O., Karlton, P., Kocher, P.: The SSL Protocol Version 3.0. Internet Draft (1996) `http://wp.netscape.com/eng/ssl3/ssl-toc.html`
2. Kungpisdan, S., Srinivasan, B., Le, P.D.: Lightweight Mobile Credit-Card Payment Protocol. LNCS Vol. 2904 (2003) 295-308

3. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (1997)
4. Krawczyk, H.: Blinding of Credit Card Numbers in the SET Protocol. LNCS, Vol. 1648 (1999) 17-28
5. Li, Y., Zhang, X.: A Security-Enhanced One-Time Payment Scheme for Credit Card. Proceedings of the International Workshop on Research Issues on data Engineering: Web Services for E-Commerce and E-Government Applications (2004) 40-47
6. Mastercard and Visa. SET Protocol Specifications. (1997) `http://www.setco.org/set_specifications.html`
7. Rubin, A.D., Wright, R.N.: Off-Line Generation of Limited-Use Credit Card Numbers. LNCS, Vol. 2339 (2002) 196-209
8. Shamir, A.: SecureClick: A Web Payment System with Disposable Credit Card Numbers. LNCS, Vol.2339 (2002) 232-242

Efficient Representation and Software Implementation of Resilient Maiorana-McFarland S-boxes

Kishan Chand Gupta and Palash Sarkar

Cryptology Research Group
Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108
{kishan_t,palash}@isical.ac.in

Abstract. We consider software implementation of resilient Maiorana-McFarland S-boxes. Such S-boxes have application in the design of stream ciphers and their efficient software implementation is important for software implementation of the corresponding stream ciphers. Most papers on construction of resilient Maiorana-McFarland S-boxes provide mathematical descriptions which are not sufficient for implementation purposes. Moreover, the mathematical descriptions do not bring out the fact that in most cases such S-boxes can be efficiently implemented using a small amount of memory. Our work shows that these S-boxes can be implemented using a small amount of memory and the output of an S-box can be computed using a small number of operations.

Keywords: S-box, Resiliency, stream cipher, Maiorana-McFarland Construction.

1 Introduction

Stream ciphers are basic cryptographic primitives and have been extensively studied. A classical model for memoryless synchronous stream ciphers uses one or more linear feedback shift registers (LFSRs) and combines the outputs of these LFSRs using a Boolean function. In the nonlinear filter model, a single LFSR is used and several sequences from this LFSR are combined using a Boolean function to obtain the desired output key sequence. The nonlinear combiner model uses several LFSRs and uses one sequence from each LFSR. These sequences are combined using a Boolean function to produce the output key sequence.

The above models have been extensively studied in the literature and different attacks have been proposed on these models. These attacks have given rise to several necessary properties of the combining (or filtering) Boolean function. Some of these properties are high nonlinearity, high algebraic degree and high resiliency. There has been extensive research on the design of Boolean functions satisfying these properties and many constructions are known. The use of suitable Boolean functions is considered to be mandatory for design of such stream ciphers.

The purpose of the current paper is to consider implementation of a class of S-boxes suitable for use in the above class of stream ciphers. An S-box produces more than one bit as output and hence use of an S-box in the above model of stream ciphers ensure a higher throughput. Design of suitable S-boxes have also been studied in the literature.

In Section 3, we describe in details the various issues related to the use of S-boxes in the nonlinear combiner model of stream ciphers. We know of no other place, where such a detailed discussion of these issues are presented. While an S-box promises a higher throughput, it also has the possibility of leaking more information. Using existing results, we show that this leakage can be upper bounded and compensated by using slightly larger length LFSRs. A recent class of attacks on stream ciphers is the class of algebraic attacks. We consider the resistance of nonlinear combiner model to algebraic attacks. More importantly, a recent paper [10] considers algebraic attacks on Maiorana-McFarland functions. We show that from an application point of view this attack need not be better than a generic algebraic attack.

Most papers describing construction of Boolean functions and S-boxes suitable for use in stream ciphers are mathematical in nature. The emphasis is usually on providing precise mathematical description of the S-boxes and proving their required properties. While this is important, there is also the requirement to study suitable methods for implementing such S-boxes and Boolean functions. Implementation is normally done by engineers for whom the mathematical description alone may be too terse.

In this paper, we provide detailed description of a method for implementing Maiorana-McFarland resilient S-boxes. A series of papers [17, 9, 8, 12, 7] have studied such S-boxes and up to now, the best construction of such S-boxes is available in [7]. We consider software implementation of the class of Maiorana-McFarland S-boxes described in [7]. We show that these S-boxes can be represented quite efficiently using a small amount of memory. Further, the output of such an S-box can be computed using a small number of operations. These two features make the use of Maiorana-McFarland S-boxes attractive from an implementation point of view. As an example, we show that a 16-bit input, 4-bit output, 2-resilient S-box can be implemented using only 488 bits. The 4-bit output of such an S-box can be computed using at most two table look-ups, 4 inner product computations of 7-bit strings and two bit vector comparisons.

Our description of the software implementation of the S-boxes in [7] is illustrated using the example mentioned above. The ideas behind this example capture the general situation quite nicely. We also show that it is easy to generalize the construction to implement a large class of S-boxes.

Related Work: The usual technique for software implementation of S-boxes is to use the truth table representation. This is usually sufficient if the size of the S-box (i.e., the number of input variables) is small. On the other hand, for S-boxes with 15 to 20 input variables, the truth table method requires a relatively large amount of memory and becomes infeasible for resource constrained devices like smart cards. In such a situation, one has to look for other implementation techniques.

Implementation of Maiorana-McFarland S-boxes has been earlier considered by Nyberg [11]. In that work, perfect nonlinear (as opposed to resilient) functions were considered. An elegant technique using an LFSR was used to realise a large class of perfect nonlinear S-boxes. The implementation for an (n, m) S-box require one $\frac{n}{2}$ bit shift register and need m inner product computations and $(m-1)$ shifts.

The main difference between our work and that of Nyberg [11] is that we consider the implementation of resilient functions whereas Nyberg considers the implementation of perfect nonlinear functions. Also the functions that we consider have a more complex structure being concatenation of affine functions of different number of variables. Thus it does not seem easy to apply Nyberg's technique to realise the kind of functions that we consider. On the other hand, Nyberg's technique scales very nicely and implementation of functions on a few hundred variables is feasible. For the functions that we consider, practical considerations will usually limit the number of variables to be less than thirty. Thus, it is not meaningful to make a direct comparison between our work and that of Nyberg.

We know of no other work which considers the software implementation of large S-boxes and to the best of our knowledge, the current work provides the first practical software implementation method for resilient Maiorana-McFarland S-boxes. Further research in this direction can address the questions of more efficient and/or optimal implementation methods.

2 Preliminaries

We briefly describe some relevant preliminaries on Boolean functions and S-boxes.

2.1 Boolean Functions

Let $\mathbb{F}_2 = GF(2)$. We consider the domain of a Boolean function to be the vector space $(\mathbb{F}_2^n, \oplus)$ over $\mathbb{F}_2$, where $\oplus$ is used to denote the addition operator over both $\mathbb{F}_2$ and the vector space $\mathbb{F}_2^n$.

A Boolean function g can be written as sum of monomials over $\mathbb{F}_2$. If an ordering of the monomials are fixed, then the sum-of-monomials representation of g is unique and this representation is called the algebraic normal form (ANF) of g. The degree of the polynomial is called the algebraic degree or simply the degree of g. If the degree of a Boolean function g is at most one we call it an affine function, further if the constant term in the polynomial is zero we call it a linear function.

If we fix an enumeration of the elements of $\mathbb{F}_2^n$, then an n-variable Boolean function can be uniquely represented by a binary string of length 2^n. One standard enumeration of $\mathbb{F}_2^n$ is $\sigma(0), \ldots, \sigma(2^n - 1)$, where $\sigma(i)$ is the n-bit binary representation of i. In the rest of the paper we will assume this enumeration of $\mathbb{F}_2^n$.

A Boolean function g is said to be degenerate on the variable x_i if

$$g(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n) = g(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n).$$

The function g is said to be degenerate if it is degenerate on some variable, else it is said to be non-degenerate. A Boolean function g is said to be t-resilient if any subfunction obtained by fixing at most t variables of g to constant is balanced. Further details on resilient functions can be found in the papers mentioned in the references (for example [1, 7–9, 12, 13, 17]). We do not provide such details as they are not relevant to the present work.

2.2 S-boxes

An (n, m) S-box (or vectorial function) is a map $f : \{0, 1\}^n \to \{0, 1\}^m$. Let f be an (n, m) S-box and g be an (m, r) S-box. We define $g \circ f$ as $(g \circ f)(x) = g(f(x))$. An (n, m) S-box f is said to be t-resilient, if $g \circ f$ is t-resilient for every non-constant $(m, 1)$ linear function g. By an (n, m, t) S-box we mean a t-resilient (n, m) S-box. As in the case of Boolean functions, if we fix the enumeration σ of the set $\mathbb{F}_2^n$, then an (n, m) S-box f is uniquely defined by a $2^n \times m$ matrix M_f, where $f(\sigma(i))$ is given by the ith row of M_f. Given a sequence of S-boxes $f_1, \cdots, f_k$; where f_i is an (n_i, m) S-box we define the *concatenation* of $f_1, \cdots, f_k$ to be the matrix

$$M = \begin{bmatrix} M_{f_1} \\ M_{f_2} \\ \vdots \\ M_{f_k} \end{bmatrix}.$$

If $2^{n_1} + \cdots + 2^{n_k} = 2^n$ for some n, then the matrix M uniquely defines an (n, m) S-box f. In this case we say f is the *concatenation* of $f_1, \cdots, f_k$.

3 Suitability of S-boxes in Nonlinear Combiner Model

In the nonlinear combiner model exactly one bit sequence is extracted from each LFSR and all the bit sequences are combined using a Boolean function to generate the key sequence. In the nonlinear combiner model we can use an S-box in place of the Boolean function to increase the throughput. (Note that the criteria of resiliency is perhaps not that important for the nonlinear filter model and hence we do not consider this model in this paper.) Now we provide justification as to why we can use S-boxes as multioutput combining functions.

We start by recalling the following result. Theorem 2 of [7] states: Let f be a t-resilient S-box and g be any arbitrary Boolean function then $(g \circ f)$ is t-CI (CI stands for correlation immune). Further $(g \circ f)$ is t-resilient if and only if g is balanced.

Thus correlation immunity of an (n, m, t)-resilient S-box is preserved under composition with an *arbitrary* m-variable Boolean function. Hence to obtain correlation between an arbitrary combination of the output variables and a non trivial linear function of the input variables, we must choose at least $(t+1)$ input variables.

The next question is the amount of information leakage. The work of Zhang and Chan [16] provide an upper bound on the correlation between a linear combination of input variables and any arbitrary (i.e., not necessarily linear) combination of the output variables. This upper bound is $2^{\frac{m}{2}}$ times more than the correlation determined by the nonlinearity of f.

Theorem 4 of [16] states: Let F be a (n, m) S-box and $c(f, g)$ is the correlation coefficient of f and g. Let the maximum correlation coefficient between F and linear function l_w be defined as

$$C_F(w) = \max_{g \in \Omega_m} c(g \circ F, l_w)$$

where Ω_m is set of all m-variable Boolean functions. The Walsh transform of $l_v \circ F$ at point w is defined as

$$W_F(v, w) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle v, F(x) \rangle \oplus \langle w, x \rangle}.$$

Then

$$C_F(w) \leq \frac{2^{m/2}}{2^n} \max_{v \in \mathbb{F}_2^m} |W_F(v, w)|. \tag{1}$$

Thus, as observed in above theorem, the value of m must be small compared to that of n. Increasing the value of m will leak extra information about the input. On the other hand increasing the value of m increases throughput. *So as stated in [16] there is a trade-off against correlation attacks and throughput of the keystream sequences in the design of S-boxes for stream ciphers.*

In the example we describe in Section 5, we have $n = 16$ and $m = 4$. The correlation between any nonzero linear combination of at least $(t + 1)$ input variables and any nonzero linear combination of output variables is at most $\frac{11}{2^9}$. If we choose any arbitrary combination of the output variables, then from (1), the upper bound for the correlation is $\frac{11}{2^7}$, which is not significantly more than $\frac{11}{2^9}$. There is also a related issue, which we discuss next.

The resistance of the system to correlation attacks depends not only on maximum correlation coefficient but also on the lengths of the LFSRs. Thus to achieve a target resistance one has to correspondingly choose the lengths of the LFSRs. We compute the amount by which the length of LFSRs are to be increased to get same level of security considering the increase in value of maximum correlation coefficient. We consider this with respect to a specific correlation attack [4]. However, the same analysis also holds for other correlation attacks. Let N be the number of key bits required to successfully carry out the correlation attack in [4]. From [4] we have

$$N \simeq \frac{1}{4}.(2kj!\ln 2)^{\frac{1}{j}}.\beta^{-2}.2^{\frac{L-k}{j}},$$

where k, j are algorithm parameters, L is effective length of LFSRs and $\beta = \max_{w\in\mathbb{F}_2^n} C_F(w)$ is the bias. In the attack of [4] the complexity of the precomputation phase is approximately $N^{\lceil (j-1)/2\rceil}$ and requires $N^{\lfloor (j-1)/2\rfloor}$ memory. The number of parity check equations that need to be stored is roughly $\frac{N^j}{j!}.2^{-(L-k)}$. To successfully carry out the attack the values of $N^{\lceil (j-1)/2\rceil}$ and $\frac{N^j}{j!}.2^{-(L-k)}$ should not be very large and hence the value of j is small (≤ 6).

Now consider the effect of replacing a Boolean function by an S-box. Let $\beta' \geq \beta$ be the modified bias. To keep the required number of key bits N same for both the Boolean function and the S-box, suppose L is increased to L'. Then

$$\beta^{-2}\times 2^{\frac{L-k}{j}} = (\beta')^{-2}\times 2^{\frac{L'-k}{j}},$$

and from this $L' = L + 2j\times\log_2\frac{\beta'}{\beta}$. In our example of Section 5, we have $\frac{\beta'}{\beta} = 4$. So for same value of N we have $L' = L + 4j \leq L + 24$. Thus if we increase the effective length of LFSR by just 24 bits, we get same level of security. In our example $t = 2$ i.e., the S-box is 2-resilient. To carry out correlation attack at least 3 LFSRs have to be taken. So length of each LFSR should increase by $\frac{24}{3} = 8$ bits. As $n = 16$, total increase in length of LFSRs will be $8\times 16 = 128$. With this increase in length of LFSRs, the effect of increase in bias will be compensated while the throughput will increase $m = 4$ times. So there is a trade-off between increase in length of LFSRs and increase in throughput.

3.1 Algebraic Attacks

Algebraic attacks [5] are a new type of attacks on stream cipher. It recovers the secret key by solving an overdefined system of multivariate equations. These attacks exploit the fact that even if a function may have high degree it may have a low degree multiple. It is shown in [5] that if f be any Boolean with n inputs then there is a Boolean function $g \neq 0$ of degree at most $\lceil \frac{n}{2}\rceil$ such that $f*g$ is of degree at most $\lceil\frac{n}{2}\rceil$. If the product $f*g$ is zero, we say that f has an annihilator g of low degree.

So we can get multiple of a Boolean function which has degree at most $\lceil\frac{n}{2}\rceil$. In [10] it is shown that Maiorana-McFarland class of n-variable Boolean functions inherently has an annihilator of degree $(n-r+1)$, where r is the number of variables of affine functions which are concatenated to construct the n-variable Boolean function. If r is very close to n then $(n-r+1)$ is small and the attacker gains. If $r > \frac{n}{2}$ the annihilator has degree $< \frac{n}{2}$. On the other hand, if $r \leq \lfloor\frac{n}{2}\rfloor$, the degree of annihilator of [10] is $\geq \lceil\frac{n}{2}\rceil$. In such a situation, the multiplier of [5] is more useful, since its degree is at most $\lceil\frac{n}{2}\rceil$.

In the construction in [7], generally $r < \frac{n}{2}$ and hence degree of annihilator is $> \frac{n}{2}$. In our example $n = 16$, $r = 7$ so $n - r + 1 = 10$ which is greater than $\frac{n}{2} = 8$. So by just considering inherent annihilator of degree $(n-r+1)$ (as in [10]) one cannot conclude that Maiorana-McFarland constructions are more prone to algebraic attacks. To apply algebraic attacks on Maiorana-McFarland class of S-boxes one has to find low degree multiple as with other classes of Boolean functions.

Now consider a multiplier of degree d is used to carry out the algebraic attack. Let the length of the effective LFSR be L. Then there are L secret key bits. Suppose N bits of the keystream are available. Then one forms N nonlinear equations in the L secret key bits. In general, it is difficult to solve such equations. The usual technique used is to linearize the system, i.e., replace each nonlinear term by a new variable. Then we have a system of N linear equations in roughly the same number of variables. Solving this linear system, one then attempts to solve the original nonlinear system. Since there are L secret key bits and the multiplier has degree d, one has to consider $\sum_{i=1}^{d}\binom{L}{i}$ nonlinear terms. To solve the system of linear equations, we must have the number of equations N to be equal (or slightly more) than the number of variables. Hence we require N to be roughly equal to $\sum_{i=1}^{d}\binom{L}{i}$. If $L = 256$ and $d = 8$, then N is greater than 2^{48}. Thus to apply the attack one needs to have at least 2^{48} bits of keystream generated from a *single* secret key. From a practical point of view, this is clearly an infeasible requirement. Thus one can choose the number of LFSRs and the effective length L in a manner which makes algebraic attacks ineffective in practice.

4 A Description of Resilient Maiorana-McFarland Construction

Maiorana-McFarland construction essentially consists of obtaining a nonlinear resilient function by concatenating small affine functions and was introduced in [1]. Later work have helped in developing and sharpening the idea. Our discussion is based on [13].

We start by providing a brief description of the construction as applied to Boolean functions and then discuss the modifications required for applying to S-boxes. As discussed in Section 2.1, an n-variable Boolean function may be represented uniquely by a binary string of length 2^n. Thus to describe a Boolean function it is sufficient to describe its representation as a string of length 2^n. The construction consists of several ideas.

First Idea: Let $f_1, \cdots, f_{2^{n-r}}$ be 2^{n-r} affine functions, where each f_i is represented by a string of length 2^r and is non-degenerate on at least $t+1$ variables. Consider the concatenation of the string representation of the functions $f_1, \cdots, f_{2^{n-r}}$. The resulting string is of length 2^n and hence represent an n-variable Boolean function f.

Second Idea: Let $g(x_n, \ldots, x_{r+1})$ be a nonlinear function and let $h(x_r, \ldots, x_1)$ be a linear function which is non-degenerate on at least $t+1$ variables. Define

$$f(x_n, \ldots, x_1) = g(x_n, \ldots, x_{r+1}) \oplus h(x_r, \ldots, x_1).$$

Suppose we fix the values of $x_n, \cdots, x_{r+1}$ to some constants $c_n, \cdots, c_{r+1}$. Then $f(c_n, \cdots, c_{r+1}, x_r, \cdots, x_1) = g(c_n, \cdots, c_{r+1}) \oplus h(x_r, \cdots, x_1)$. Since $g(c_n, \cdots, c_{r+1})$

is a constant, the substring of f as $x_r, \cdots, x_1$ runs over 2^r possible values is equal to either the string representation of h or the string representation of $1 \oplus h$. Thus the entire string f is a concatenation of h and $1 \oplus h$.

Generalization: Let $R_1 + \cdots + R_k = 2^n$ where R_i is a multiple of 2^{r_i} and is of the form $R_i = M_i \times 2^{r_i} = (2^{s_{i,1}} + \cdots + 2^{s_{i,k_i}})2^{r_i} = 2^{s_{i,1}}2^{r_i} + \cdots + 2^{s_{i,k_i}}2^{r_i}$. Let $g_{i,j}$ be a nonlinear function of $s_{i,j}$ variable and l_i is a linear function of r_i variables disjoint from the previous $s_{i,j}$ variables. Let $h_{i,j} = g_{i,j} \oplus l_i$. By the second idea $h_{i,j}$ is a concatenation of l_i or $\overline{l_i}$. Consider concatenation of $h_{i,1}, \cdots, h_{i,k_i}$ and call it f_i. Each f_i may not represent a Boolean function since its string representation may not be a power of 2. Let f be concatenation of $f_1, \cdots, f_k$. This f is an n-variable Boolean function which is ultimately a concatenation of the l_is and the $(1 \oplus l_i)$s.

4.1 Maiorana-McFarland Construction for S-boxes

Our description of Maiorana-McFarland construction for S-boxes closely follows the description given in [7]. The following result which is restatement of Lemma 7 in [8] is crucial to Maiorana-McFarland construction for S-boxes. For proof see [8] and further details about the construction see [7].

Theorem 1. *Let C be a $[u, m, t+1]$ code. Then it is possible to construct $(2^m - 1) \times m$ matrix D with entries from C, such that, $\{c_1 D_{i,1} \oplus \cdots \oplus c_m D_{i,m} : 1 \le i \le 2^m - 1\} = C \setminus \{(0, \cdots, 0)\}$ for each nonzero vector $(c_1, \cdots, c_m) \in \mathbb{F}_2^m$.*

Let D be the matrix in Theorem 1. For $(1 \le i \le 2^m - 1)$ and $(1 \le j \le m)$ define a u-variable linear function $L_{i,j}(x_1, \cdots, x_u) \overset{\triangle}{=} \langle D_{i,j}, (x_1, \cdots, x_u) \rangle$. Given the code C we define a $(2^m - 1) \times m$ matrix $L(C)$ whose entries are u-variable linear functions by defining the i, j th entry of $L(C)$ to be $L_{i,j}(x_1, \cdots, x_u)$.

For positive integers k, l with $k \le l$, we define $L(C, k, l)$ to be the submatrix of $L(C)$ consisting of the rows k to l. Thus $L(C, 1, 2^m - 1) = L(C)$. Let $G(y_1, \cdots, y_p)$ be a (p, m) S-box whose component functions are $G_1, \cdots, G_m$. We define $G \oplus L(C, k, l)$ to be an $(l - k + 1) \times m$ matrix whose i, j th entry is

$$G_j(y_1, \cdots, y_p) \oplus L_{k+i-1,j}(x_1, \cdots, x_u)$$

for $1 \le i \le l - k + 1$ and $1 \le j \le m$. If $l - k + 1 = 2^r$ for some r then $G \oplus L(C, k, l)$ defines an S-box $F : \{0,1\}^{r+p+u} \rightarrow \{0,1\}^m$ in the following manner.

$$F_j(z_1, \cdots, z_r, y_1, \cdots, y_p, x_1, \cdots, x_u) = G_j(y_1, \cdots, y_p) \oplus L_{k+i-1,j}(x_1, \cdots, x_u)$$

where $1 \le j \le m$, $1 \le i \le 2^r$, $F_1, \cdots, F_m$ are the component functions of F and $z_1 \cdots z_r$ is the binary representation of $i - 1$. By $F = G \oplus L(C, k, l)$ we will mean the above representation of the S-box F.

In the matrix $M = G(y_1, \cdots, y_p) \oplus L(C, k, l)$ we say that the row $L_{i,*}$ of $L(C)$ is repeated 2^p times. Let $G(y_1, \cdots, y_p)$ and $H(y_1, \cdots, y_q)$ be (p, m) and (q, m) S-boxes respectively and $M_1 = G \oplus L(C, k, l)$, $M_2 = H \oplus L(C, k, l)$. Then we say that the row $L_{i,*}$ of $L(C)$, $(k \le i \le l)$ is repeated a total of $2^p + 2^q$ times in the matrix $[M_1\ M_2]^T$. A row of $L(C)$ can be repeated 2^{r_1} or $2^{r_1}+2^{r_2}$ or $2^{r_1}+2^{r_2}+2^{r_3}$ times as required.

A description of Maiorana-McFarland resilient S-box in [7] consists of the following information.

1. The matrix $L(C)$.
2. A list $[(n_1,R_1),(n_2,R_2),\cdots,(n_k,R_k)]$ which signifies that n_i rows of $L(C)$ are to be repeated R_i times each. For $1 \le i \le k$, either $n_i = 2^{r_i}$ or $n_i = 2^{r_{i,1}}+2^{r_{i,2}}$ or $n_i = 2^{r_{i,1}}+2^{r_{i,2}}+2^{r_{i,3}}$.
3. For $1 \le i \le k$ if $n_i = 2^{r_i}$, then G_i is a nonlinear (r_i, m) S-box otherwise $G_{i,j}$ is a nonlinear $(r_{i,j}, m)$ S-box where $2 \le j \le 3$.

The method described in [7] consists of around 15–20 different cases depending on the values of the parameters n, m and t. The cases are chosen to maximise the nonlinearity. Each case consists of a description of the above form. Hence the actual implementation method for any specific function can be described by a general method. In the next section, we illustrate this general method by considering a specific example, which is sufficiently general enough to cover all the cases mentioned in [7].

5 A Concrete Example

In this section we provide description of the construction of a $(16,4,2)$ S-box. The purpose of this example is to illustrate the methodology behind the general description provided later.

5.1 Construction of $L(C)$

We use $[7,4,3]$ linear Hamming code C. Let $c_0 = 1101000, c_1 = 1010100, c_2 = 0110010, c_3 = 1110001$ be a basis of C. Let β be a root of primitive polynomial x^4+x+1 and $(1,\beta,\beta^2,\beta^3)$ be a polynomial basis of $GF(2^4)$. Any element γ of $GF(2^4)$ can be written as $\gamma = \gamma_0+\gamma_1\beta+\gamma_2\beta^2+\gamma_3\beta^3$ where $\gamma_0,\gamma_1,\gamma_2,\gamma_3 \in \{0,1\}$. We define a bijection $\phi : GF(2^4) \mapsto C$ by (see Lemma 7 in [8])

$$\phi(a_0+a_1\beta+a_2\beta^2+a_3\beta^3) = a_0c_0 \oplus a_1c_1 \oplus a_2c_2 \oplus a_3c_3.$$

Then

$$\begin{array}{lll}
\phi(1) = 1101000, & \phi(\beta) = 1010100, & \phi(\beta^2) = 0110010, \\
\phi(\beta^3) = 1110001, & \phi(\beta^4) = 0111100, & \phi(\beta^5) = 1100110, \\
\phi(\beta^6) = 1000011, & \phi(\beta^7) = 1001101, & \phi(\beta^8) = 1011010, \\
\phi(\beta^9) = 0100101, & \phi(\beta^{10}) = 0001110, & \phi(\beta^{11}) = 0010111, \\
\phi(\beta^{12}) = 1111111, & \phi(\beta^{13}) = 0101011, & \phi(\beta^{14}) = 0011001.
\end{array}$$

We have constructed the 15×4 matrix D as given below (see Theorem 1 and Lemma 7 in [8]).

$$D = \begin{bmatrix} \phi(1) & \phi(\beta) & \phi(\beta^2) & \phi(\beta^3) \\ \phi(\beta) & \phi(\beta^2) & \phi(\beta^3) & \phi(\beta^4) \\ \vdots & \vdots & \vdots & \vdots \\ \phi(\beta^{14}) & \phi(1) & \phi(\beta) & \phi(\beta^2) \end{bmatrix}.$$

As defined after Theorem 1 we have the following 15×4 matrix $L(C)$ whose entries are 7 variable linear functions.

$$\begin{bmatrix}
x_1 \oplus x_2 \oplus x_4 & x_1 \oplus x_3 \oplus x_5 & x_2 \oplus x_3 \oplus x_6 & x_1 \oplus x_2 \oplus x_3 \oplus x_7 \\
x_1 \oplus x_3 \oplus x_5 & x_2 \oplus x_3 \oplus x_6 & x_1 \oplus x_2 \oplus x_3 \oplus x_7 & x_2 \oplus x_3 \oplus x_4 \oplus x_5 \\
x_2 \oplus x_3 \oplus x_6 & x_1 \oplus x_2 \oplus x_3 \oplus x_7 & x_2 \oplus x_3 \oplus x_4 \oplus x_5 & x_1 \oplus x_2 \oplus x_5 \oplus x_6 \\
x_1 \oplus x_2 \oplus x_3 \oplus x_7 & x_2 \oplus x_3 \oplus x_4 \oplus x_5 & x_1 \oplus x_2 \oplus x_5 \oplus x_6 & x_1 \oplus x_6 \oplus x_7 \\
x_2 \oplus x_3 \oplus x_4 \oplus x_5 & x_1 \oplus x_2 \oplus x_5 \oplus x_6 & x_1 \oplus x_6 \oplus x_7 & x_1 \oplus x_4 \oplus x_5 \oplus x_7 \\
x_1 \oplus x_2 \oplus x_5 \oplus x_6 & x_1 \oplus x_6 \oplus x_7 & x_1 \oplus x_4 \oplus x_5 \oplus x_7 & x_1 \oplus x_3 \oplus x_4 \oplus x_6 \\
x_1 \oplus x_6 \oplus x_7 & x_1 \oplus x_4 \oplus x_5 \oplus x_7 & x_1 \oplus x_3 \oplus x_4 \oplus x_6 & x_2 \oplus x_5 \oplus x_7 \\
x_1 \oplus x_4 \oplus x_5 \oplus x_7 & x_1 \oplus x_3 \oplus x_4 \oplus x_6 & x_2 \oplus x_5 \oplus x_7 & x_4 \oplus x_5 \oplus x_6 \\
x_1 \oplus x_3 \oplus x_4 \oplus x_6 & x_2 \oplus x_5 \oplus x_7 & x_4 \oplus x_5 \oplus x_6 & x_3 \oplus x_5 \oplus x_6 \oplus x_7 \\
x_2 \oplus x_5 \oplus x_7 & x_4 \oplus x_5 \oplus x_6 & x_3 \oplus x_5 \oplus x_6 \oplus x_7 & \begin{array}{c} x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \\ x_5 \oplus x_6 \oplus x_7 \end{array} \\
x_4 \oplus x_5 \oplus x_6 & x_3 \oplus x_5 \oplus x_6 \oplus x_7 & \begin{array}{c} x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \\ x_5 \oplus x_6 \oplus x_7 \end{array} & x_2 \oplus x_4 \oplus x_6 \oplus x_7 \\
x_3 \oplus x_5 \oplus x_6 \oplus x_7 & \begin{array}{c} x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \\ x_5 \oplus x_6 \oplus x_7 \end{array} & x_2 \oplus x_4 \oplus x_6 \oplus x_7 & x_3 \oplus x_4 \oplus x_7 \\
\begin{array}{c} x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \\ x_5 \oplus x_6 \oplus x_7 \end{array} & x_2 \oplus x_4 \oplus x_6 \oplus x_7 & x_3 \oplus x_4 \oplus x_7 & x_1 \oplus x_2 \oplus x_4 \\
x_2 \oplus x_4 \oplus x_6 \oplus x_7 & x_3 \oplus x_4 \oplus x_7 & x_1 \oplus x_2 \oplus x_4 & x_1 \oplus x_3 \oplus x_5 \\
x_3 \oplus x_4 \oplus x_7 & x_1 \oplus x_2 \oplus x_4 & x_1 \oplus x_3 \oplus x_5 & x_2 \oplus x_3 \oplus x_6
\end{bmatrix}$$

5.2 Construction of $(16, 4, 2)$ Resilient S-box

Following [7] the description of a $(16, 4, 2)$-resilient S-box with nonlinearity $2^{15} - 11 \times 2^6$ is as follows (see Case: 3(d)(ii) first item of Part-A and Part B in [7].

1. Matrix $L(C)$ as constructed above.
2. The list $[(2, 2^{m+1} + 2^1 + 2^0), (2^m - 3, 2^{m+1} + 2^1)] = [(2, 2^5 + 2^1 + 2^0), (2^4 - 3, 2^5 + 2^1)] = [(2, 35), (13, 34)]$. This implies that 13 rows (along with their complements) of $L(C)$ are to be repeated 34 times each and 2 rows (along with their complements) are to be repeated 35 times each.
3. A nonlinear $(5, 4)$ and an $(1, 4)$ S-Box G_1 and G_2 respectively as described below.

We construct a maximally nonlinear $(5, 4)$ S-box G_1. For this we define a bijection on $GF(2^5)$ by $x \mapsto x^3$ (see [6]). We represent this bijection as a map from $\{0, 1\}^5$ to $\{0, 1\}^4$ by representing $GF(2^5)$ using the primitive polynomial $x^5 + x + 1$. The S-box G_1 is obtained by dropping one bit of the output of this bijection. Let $G_{1,1}, G_{1,2}, G_{1,3}, G_{1,4}$ be the four component functions of G_1. Each $G_{1,j}$ is a 5-variable Boolean function as given below.

$$G_{1,1}(y_1, \cdots, y_5) = y_4 \oplus y_1y_3 \oplus y_1y_5 \oplus y_2y_3 \oplus y_2y_4 \oplus y_2y_5 \oplus y_3y_4 \oplus y_3y_5 \oplus y_4y_5$$
$$G_{1,2}(y_1, \cdots, y_5) = y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_1y_5 \oplus y_3y_4 \oplus y_3y_5$$
$$G_{1,3}(y_1, \cdots, y_5) = y_5 \oplus y_1y_2 \oplus y_1y_3 \oplus y_2y_3 \oplus y_1y_5 \oplus y_3y_5 \oplus y_4y_5$$
$$G_{1,4}(y_1, \cdots, y_5) = y_3 \oplus y_4 \oplus y_5 \oplus y_1y_2 \oplus y_1y_4 \oplus y_4y_5$$

The output of G_1 is given by a 4-bit string which can be represented as a hexadecimal digit. We can write G_1 as a 32-tuple of hexadecimal digits having the following form:

$$G_1 = (0, 0, 4, 7, 5, F, B, 2, D, C, 1, 3, 8, F, 2, A, 7, 9, B, 6, C, 8, A, D, 1, E, 5, 9, 6, 3, 8, D)$$

This representation of G_1 is useful for a table look-up implementation.

The $(1, 4)$ S-box G_2 is taken to be the constant function $G_2(y_5) = (0, 0, 0, 0)$. Thus the four component functions $G_{2,j}(y_5)$ $(1 \le j \le 4)$ are single variable constant functions. We now define

$$\begin{aligned} F_1(y_1, \ldots, y_5, x_1, \ldots, x_7) &= G_1(y_1, \ldots, y_5) \oplus L(C) \\ F_2(y_5, x_1, \ldots, x_7) &= G_2(y_5) \oplus L(C) \\ F_3(x_1, \ldots, x_7) &= L(C, 1, 2) \end{aligned}$$

Thus $F_1(y_1, \ldots, y_5, x_1, \ldots, x_7)$ is of the form $[G_{1,1} \oplus \mathbf{c_1}, G_{1,2} \oplus \mathbf{c_2}, G_{1,3} \oplus \mathbf{c_3}, G_{1,4} \oplus \mathbf{c_4}]$, where $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, \mathbf{c_4}$ are the four column vectors of $L(C)$. Further, $G_{1,j} \oplus \mathbf{c_j} = [G_{1,j} \oplus c_{1,j}, \ldots, G_{1,j} \oplus c_{15,j}]^T$ where $c_{i,j}$ is the (i, j)th entry of $L(C)$. We have $F_2(y_5, x_1, \ldots, x_7) = L(C)$, where each entry of $L(C)$ is treated as degenerate function of y_5. Lastly $F_3(x_1, \ldots, x_7)$ is defined as

$$\begin{bmatrix} x_1 \oplus x_2 \oplus x_4 & x_1 \oplus x_3 \oplus x_5 & x_2 \oplus x_3 \oplus x_6 & x_1 \oplus x_2 \oplus x_3 \oplus x_7 \\ x_1 \oplus x_3 \oplus x_5 & x_2 \oplus x_3 \oplus x_6 & x_1 \oplus x_2 \oplus x_3 \oplus x_7 & x_2 \oplus x_3 \oplus x_4 \oplus x_5 \end{bmatrix}.$$

Remarks:

1. Each entry in the 15×4 matrix F_1 is a 12-variable Boolean function. This accounts for 32 repetitions of each of the 15 rows of $L(C)$.
2. Each entry in the 15×4 matrix F_2 is an 8-variable Boolean function. This accounts for 2 repetitions of each of the 15 rows of $L(C)$.
3. Each entry in the 2×4 matrix F_3 is a 7-variable Boolean function. This accounts for 1 repetition of each of the first two rows of $L(C)$.

Thus the first two rows of $L(C)$ are repeated 35 times and the next 13 rows of $L(C)$ are repeated 34 times. The desired $(16, 4, 2)$ S-box F is the concatenation of F_1, F_2 and F_3 as we explain below. We use the notation $F_{k,i,j}$ to denote the (i, j)th entry of matrix F_k and $F_{k,i}$ to be the ith row of matrix F_k. For a binary string $a_1, \ldots, a_k$, we define $\delta(a_1, \cdots, a_k) = a_1 2^k + a_2 2^{k-1} + \cdots + a_{k-1}2 + a_k + 1$. We write $\overline{x} = (x_1, \ldots, x_7)$, $\overline{y} = (y_1, \ldots, y_5)$ and $\overline{z} = (z_1, \ldots, z_4)$.

$$\left.\begin{aligned} F(\overline{z}, \overline{y}, \overline{x}) &= F_{1,\delta(z_1,z_2,z_3,z_4)}(\overline{y}, \overline{x}) && \text{if } \overline{z} \neq (1111) \\ &= F_{2,\delta(y_1,y_2,y_3,y_4)}(y_5, \overline{x}) && \text{if } \overline{z} = (1111) \text{ and } (y_1, y_2, y_3, y_4) \neq (1111) \\ &= F_{3,y_5}(\overline{x}) && \text{if } \overline{z} = (y_1, y_2, y_3, y_4) = (1111) \end{aligned}\right\} \quad (2)$$

The above defined F is our $(16, 4, 2)$ S-box. Next we explain how to implement the above $(16, 4, 2)$ S-box efficiently and to compute the 4-bit output when 16-bit input is given.

5.3 Implementation

For implementing the S-box we need to store the following.

1. A 15×4 table D whose each entry is a 7-bit binary vector. Thus each entry of D represents a 7-variable linear function.
2. An array G of length 32 whose each entry is a 4-bit vector.

Thus the number of bits of storage required is $15 \times 4 \times 7 + 32 \times 4 = 488$ bits.

Remark: Note that the number of bits required to store the representation is only 488 bits which is much less than directly storing F as a look-up table of size $4 \times 2^{16} = 2^{18}$ bits. On the other hand, implementation by the truth-table method will require one table look-up to compute the output corresponding to an input. For our implementation, we have to use a small number of operations as described in Section 5.4.

5.4 Computing the Output

The question now is the following: Given the representation of F as defined by (2) in Section 5.3, how to compute the 4-bit output when a 16-bit input is given? Here we provide the answer to this question.
Let the input be $(z_1, z_2, z_3, z_4, y_1, y_2, y_3, y_4, y_5, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. We will denote the kth row of D by D_k and the kth entry of G by G_k. Note that D_k is a 4-tuple of 7-bit strings and G_k is a 4-bit string. By $D_{k,j}$ $(1 \le j \le 4)$ we will denote the (k, j)th entry of D. In the following, the notation $\langle \overline{x}, \overline{y} \rangle$ denotes the inner product of $\overline{x}$ and $\overline{y}$.

1. If $(z_1, z_2, z_3, z_4) = (y_1, y_2, y_3, y_4) = (1111)$ then the output is the 4-bit vector given by
$$(\langle D_{y_5,1}, \overline{x} \rangle, \langle D_{y_5,2}, \overline{x} \rangle, \langle D_{y_5,3}, \overline{x} \rangle, \langle D_{y_5,4}, \overline{x} \rangle)$$
The cost is one table lookup for D, 4 inner product computations and one comparison to check $z_1z_2z_3z_4y_1y_2y_3y_4 = 11111111$.
2. If $(z_1, z_2, z_3, z_4) = 1111$ and $(y_1, y_2, y_3, y_4) \ne (1111)$ then we compute $k = \delta(y_1, y_2, y_3, y_4)$ and the output is the 4-bit vector given by
$$(\langle D_{k,1}, \overline{x} \rangle, \langle D_{k,2}, \overline{x} \rangle, \langle D_{k,3}, \overline{x} \rangle, \langle D_{k,4}, \overline{x} \rangle).$$
The cost is one table lookup for D, 4 inner product computations and two comparisons to verify if $(z_1, z_2, z_3, z_4) = 1111$ and $(y_1, y_2, y_3, y_4) \ne (1111)$.
3. If $(z_1, z_2, z_3, z_4) \ne (1111)$ then we compute $k = \delta(z_1, z_2, z_3, z_4)$, $l = \delta(y_1, y_2, y_3, y_4, y_5)$ and the output is the 4-bit vector given by
$$G_l \oplus (\langle D_{k,1}, \overline{x} \rangle, \langle D_{k,2}, \overline{x} \rangle, \langle D_{k,3}, \overline{x} \rangle, \langle D_{k,4}, \overline{x} \rangle).$$
The cost is one table lookup each for D and G, 4 inner product computations, one XOR of 4-bit strings and one comparison to check $(z_1, z_2, z_3, z_4) \ne (1111)$.

The maximum cost occurs in the last case, which is 2 table look-ups, 4 inner product computations, one XOR and one bit string comparison. Alternatively, if F is stored directly as a look-up table of 4×2^{16} bits then the cost is one table look-up. However, this table look-up is into a large table (of size 4×2^{16} bits) whereas the two table look-ups into D and G are into much smaller sized tables.

Remarks: In this example we use a linear code, so we just need to encode 4 basis vectors along with a small memory of 4×16 bits to describe the bijection. Also D has shift symmetry which means a further reduction in bit storage by 4. But for general construction this reduction may not work. We are not claiming minimum storage. In some other technique storage may be less but then computation time will be higher.

6 General Methodology

The example in Section 5 illustrates the method for software implementation of the resilient S-boxes described in [7]. The basic points behind the example generalize quite easily. In this section, we briefly describe this generalization.

As mentioned in Section 4.1, a description of an S-box constructed using the method of [7] consists of the matrix $L(C)$, the list of pairs indicating repetition pattern of linear functions and the required nonlinear S-boxes G_i's and/or $G_{i,j}$'s.

The matrix $L(C)$ is stored as a $((2^m - 1) \times m)$ matrix D whose each entry is a u-bit vector. Thus storing $L(C)$ requires $u \times m \times (2^m - 1)$ bits. Each G_i is an (r_i, m) S-box and each $G_{i,j}$ is an $(r_{i,j}, m)$ S-box. Storing these as look-up tables require $m \times 2^{r_i}$ and $m \times 2^{r_{i,j}}$ bits respectively. Nothing else requires to be stored. Since the r_i's and the $r_{i,j}$'s are much smaller compared to n, the total amount of storage space required will be much smaller than $m \times 2^n$ bits.

The constructed S-box F can be expressed as in (2). The strategy for evaluating the output of F depends on the nature of this expression. Moreover, it is clear that such a strategy can be formulated as described in Section 5.4. The cost of evaluating the output will be m inner product computation of u-bit strings; one table look-up into matrix D; a few table look-ups into the G_i's and the $G_{i,j}$'s and some bit vector comparisons to determine the case which is applicable for the given input vector. Given a particular S-box, the actual strategy is easy to design as discussed in Section 5.4.

Maiorana-McFarland construction consists of concatenation of linear/affine S-boxes. The order of concatenation does not affect the construction. Thus in general one can consider any permutation of the functions to be concatenated. We will permute the rows of the matrix D to achieve this task. The method of changing the order of concatenation is simple. We will take an array $A[1, \ldots, 2^{m-1}]$ of length 2^{m-1}. In the array A, the desired permutation of $\{1, \ldots, 2^{m-1}\}$ is stored. Considering the matrix D as described in Section 5.4 we will take $D_{A[i],j}$ instead of $D_{i,j}$ in computing the output. Thus we see that with this simple data structure using an array A of length 2^{m-1} we can get $(2^{m-1})!$ different S-boxes.

7 Conclusion

In this paper, we have described a method for software implementation of resilient Maiorana-McFarland S-boxes. The main features of our implementation are small amount of memory and fast evaluation of the output. These features show that Maiorana-McFarland S-boxes are suitable candidates for use in software implementation of nonlinear combiner model of stream ciphers.

Acknowledgement

We would like to thank the reviewers for detailed comments which helped to improve the presentation of the paper.

References

1. P. Camion, C. Carlet, P. Charpin and N. Sendrier . On correlation immune functions. *In Advances in Cryptology - CRYPT0 1991*, pages 86–100, Lecture Notes in Computer Science, Springer-Verlag, 1992.
2. A. Canteaut and M. Videau. Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. *Advances in Cryptology – Eurocrypt 2002*, LNCS 2332, pages 518–533.
3. S. Chee, S. Lee, D. Lee and S. H. Sung . On the correlation immune functions and their nonlinearity. *In Advances in Cryptology - Asiacrypt 1996*, pages 232–243, Lecture Notes in Computer Science, Springer-Verlag, 1996.
4. V. Chepyzhov, T. Johansson and B. Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In *Fast Software Encryption – FSE 2000*, pp 181 –195, Lecture Notes in Computer Science, Springer-Verlag, 2000.
5. N. Courtois and W. Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback, *In Advances in Cryptology - EUROCRYPT 2003*, 345-359. Extended version, available at http://www.cryptosystem.net/stream/.
6. Hans Dobbertin, Almost Perfect Nonlinear Power Functions on $GF(2^n)$: The Welch Case. *IEEE Transactions on Information Theory*, Vol 45, No 4, pp. 1271-1275, 1999.
7. K. C. Gupta and P. Sarkar. Improved Construction of Nonlinear S-Boxes. *In Advances in Cryptology - Asiacrypt 2002*, pp 466–483, Lecture Notes in Computer Science, Springer-Verlag, 2002.
8. T. Johansson and E. Pasalic. A construction of resilient functions with high nonlinearity. *International Symposium on Information Theory*, 2000.
9. K. Kurosawa, T. Satoh and K. Yamamoto. Highly nonlinear t-resilient functions. Journal of Universal Computer Science, vol.3, no. 6, pp. 721-729, Springer Publishing Company, 1997.
10. W. Meier, E. Pasalic and C. Carlet. Algebraic attacks and decomposition of Boolean Functions. To be published In *Advances in Cryptology – EUROCRYPT'04*.
11. K. Nyberg. Perfect Nonlinear S-boxes. In *Advances in Cryptology – EUROCRYPT 1991*, pages 378–386, Lecture Notes in Computer Science, Springer-Verlag, 1991.
12. E. Pasalic and S. Maitra. Linear Codes in Generalized Construction of Resilient Functions with Very High Nonlinearity. *IEEE Transactions on Information Theory*, Vol 48, No 8, pp. 2182-2191, August 2002.

13. P. Sarkar and S. Maitra. Construction of Nonlinear Boolean Functions with Important Cryptographic Properties. *In Advances in Cryptology - EUROCRYPT 2000*, pages 485–506, Lecture Notes in Computer Science, Springer-Verlag, 2000.
14. J. Seberry, X.-M. Zhang and Y. Zheng . On construction and nonlinearity of correlation immune Boolean functions. *In Advances in Cryptology - EUROCRYPT 1993*, pages 181–199, Lecture Notes in Computer Science, Springer-Verlag, 1994.
15. M. Zhang. Maximum Correlation Analysis of Nonlinear Combining Functions in Stream Ciphers. *Journal of Cryptology*, 13(3): 301–314, 2000.
16. M. Zhang and A. H. Chan. Maximum Correlation Analysis of Nonlinear S-boxes in Stream Ciphers. In *Advances in Cryptology – CRYPTO 2000*, Lecture Notes in Computer Science, pages 501–514. Springer-Verlag, 2000.
17. X.-M. Zhang and Y. Zheng, On Cryptographically Resilient Functions. *IEEE Transactions on Information Theory*, Vol 43 , No 5, pp. 1740-1747 , 1997.

Signed Digit Representation with NAF and Balanced Ternary Form and Efficient Exponentiation in $GF(q^n)$ Using a Gaussian Normal Basis of Type II

Soonhak Kwon

Inst. of Basic Science and Dept. of Mathematics, Sungkyunkwan University, Suwon 440-746, Korea
shkwon@skku.edu

Abstract. We present an efficient exponentiation algorithm for a finite field $GF(q^n)$ with small characteristic determined by a Gaussian normal basis of type II using signed digit representation of the exponents. Our signed digit representation uses a nonadjacent form (NAF) for $GF(2^n)$ and the balanced ternary number system for $GF(3^n)$. It is generally believed that a signed digit representation is hard to use when a normal basis is given because the inversion of a normal element requires quite a computational delay. On the other hand, the method of a signed digit representation is easily applicable to the fields with polynomial bases. However our result shows that a special normal basis called a Gaussian normal basis of type II or an optimal normal basis (ONB) of type II has a nice property which admits an effective exponentiation using signed digit representations of the exponents.

Keywords: finite field, Gaussian normal basis, optimal normal basis, exponentiation, signed digit representation, NAF (nonadjacent form), balanced ternary number system.

1 Introduction

Arithmetic of finite fields finds various applications in many cryptographic areas these days. Especially, fast exponentiation is very important in such applications as Diffie-Hellman key exchange and pseudo random bit generators. Though exponentiation is the most time consuming and complex arithmetic operation, in some situations such as Diffie-Hellman key exchange, one can devise an efficient exponentiation algorithm since a fixed (primitive) element is raised to many different powers. Let $GF(q^n)$ be a finite field with q^n element where q is a prime and let $g \in GF(q^n)$ be a primitive element (or an element of high multiplicative order). Roughly speaking, the computation of g^s for arbitrary values of s is studied from two different directions. One is the use of precomputation with vector addition chains such as BGMW method [1] and its improvements by Lim and Lee [2] and also by Rooij [3].

The other approach is suggested by Gao et al. [4,5,6] and it uses a special primitive element called a Gauss period of type k, which generates a normal basis for $GF(q^n)$. The BGMW method and its improvements are applicable to arbitrary finite field $GF(q^n)$ and are very flexible. On the other hand, an ideal version of BGMW method requires a memory of order $O(n \log q/\log(n \log q))$ values in $GF(q^n)$ and multiplications of order $O(\log(n \log q))$ which amounts to an order of $O(n^2 \log^2 q \log(n \log q))$ bit additions. An algorithm proposed by Gao et al. is not applicable to all finite fields. However, it does not need a precomputation and the complexity of the algorithm is $O(kqn^2)$ additions. Therefore if q is small and if there is a Gauss period of high order of type k for small value of $k \geq 2$, then the method of Gao et al. outperforms the precomputation methods.

In this paper, we propose a new exponentiation algorithm in $GF(q^n)$ with a Gaussian normal basis of type II using a signed digit representation of the exponents. Our signed digit representation uses a nonadjacent form (NAF) when $q = 2$ and a balanced ternary form when $q = 3$. It is shown that our new exponentiation algorithm is approximately 33 percent faster than the existing algorithm using a Gaussian normal basis of type II. Also we show that our method of using a normal basis is significantly faster than the method of using a polynomial basis, since the Frobenius map is free in our basis while a considerable amount of $GF(q)$-additions is required for the Frobenius map with a polynomial basis. We give an explicit comparison of our method with the method using a trinomial basis with signed digit representation and show that our algorithm is at least 33 percent faster than the trinomial basis method.

2 Gaussian Normal Basis of Type k in $GF(q^n)$

We will briefly review the theory of Gauss periods and the exponentiation algorithm of Gao et al. [5]. Let n, k be positive integers such that $p = nk + 1$ is a prime different from q. Let $K = \langle \tau \rangle$ be a unique subgroup of order k in $GF(p)^\times$. Let β be a primitive pth root of unity in $GF(q^{nk})$. The following element

$$\alpha = \sum_{j=0}^{k-1} \beta^{\tau^j} \tag{1}$$

is called a Gauss period of type (n, k) over $GF(q)$. Let $ord_p q$ be the order of q (mod p) and assume $gcd(nk/ord_p q, n) = 1$. Then it is well known that α is a normal element in $GF(q^n)$. That is, $\{\alpha, \alpha^q, \alpha^{q^2}, \cdots, \alpha^{q^{n-1}}\}$ is a basis for $GF(q^n)$ over $GF(q)$. It is called a Gaussian normal basis of type (n, k) over $GF(q)$ or of type k in $GF(q^n)$.

Since $K = \langle \tau \rangle$ is a subgroup of order k in $GF(p)^\times$, a cyclic group of order nk, the quotient group $GF(p)^\times/K$ is also a cyclic group of order n and the generator of the group is qK. Therefore we have a coset decomposition of $GF(p)^\times$ as a disjoint union,

$$GF(p)^\times = K_0 \cup K_1 \cup K_2 \cdots \cup K_{n-1}, \tag{2}$$

where $K_i = q^i K, 0 \leq i \leq n-1$. Note that, from the equation (2), any element in $GF(p)^\times$ is uniquely written as $\tau^s q^t$ for some $0 \leq s \leq k-1$ and $0 \leq t \leq n-1$. For each $0 \leq i \leq n-1$, we have

$$\begin{aligned} \alpha\alpha^{q^i} &= \sum_{s=0}^{k-1} \beta^{\tau^s} \sum_{t=0}^{k-1} \beta^{\tau^t q^i} \\ &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^{t-s}q^i)} = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t q^i)}. \end{aligned} \tag{3}$$

There are unique $0 \leq u \leq k-1$ and $0 \leq v \leq n-1$ such that $1+\tau^u q^v = 0 \in GF(p)$. If $t \neq u$ or $i \neq v$, then we have $1+\tau^t q^i \in K_{\sigma(t,i)}$ for some $0 \leq \sigma(t,i) \leq n-1$ depending on t and i. Thus we may write $1+\tau^t q^i = \tau^{t'} q^{\sigma(t,i)}$ for some t'. Now when $i \neq v$,

$$\begin{aligned} \alpha\alpha^{q^i} &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t q^i)} = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(\tau^{t'} q^{\sigma(t,i)})} \\ &= \sum_{t=0}^{k-1}\sum_{s=0}^{k-1} \beta^{\tau^{s+t'} q^{\sigma(t,i)}} = \sum_{t=0}^{k-1} \alpha^{q^{\sigma(t,i)}}. \end{aligned} \tag{4}$$

Also when $i = v$,

$$\begin{aligned} \alpha\alpha^{q^v} &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t q^v)} \\ &= \sum_{t\neq u}\sum_{s=0}^{k-1} \beta^{\tau^s(\tau^{t'} q^{\sigma(t,v)})} + \sum_{s=0}^{k-1} \beta^{\tau^s(1+\tau^u q^v)} \\ &= \sum_{t\neq u}\sum_{s=0}^{k-1} \beta^{\tau^{s+t'} q^{\sigma(t,v)}} + \sum_{s=0}^{k-1} 1 = \sum_{t\neq u} \alpha^{q^{\sigma(t,v)}} + k. \end{aligned} \tag{5}$$

Therefore $\alpha\alpha^{q^i}$ is computed by the sum of at most k basis elements in $\{\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}}\}$ for $i \neq v$ and $\alpha\alpha^{q^v}$ is computed by the sum of at most $k-1$ basis elements and the constant term $k \in GF(q)$. Using these ideas, Gao et al. [5] showed the following.

Theorem 1. *Let α be a Gauss period of type k in $GF(q^n)$ with $k \geq 2$. Then for any $0 \leq r < q^n$, α^r can be computed in $(k-1)(q-1)n(n+1)$ additions in $GF(q)$.*

Sketch of Proof. Write $r = \sum_{j=0}^{n-1} r_j q^j$ with $0 \leq r_j < q$. Then the following algorithm gives an output α^r.

Assuming that the qth Frobenius map $\alpha \to \alpha^q$ is (almost) free, from the equations (4) and (5), $A\alpha^{q^i}$ is computed by at most $(k-1)(n+1)$ additions

Table 1. Exponentiation algorithm in [5].

```
Input: $r = \sum_{j=0}^{n-1} r_j q^j$ with $0 \le r_j < q$
Output: $\alpha^r$
$A \leftarrow 1$
for ($i = 0$ to $n-1$ ; $i++$)
  if $r_i \ne 0$
    for ($j = 1$ to $r_i$ ; $j++$)
      $A \leftarrow A\alpha^{q^i}$
    end for
  end if
end for
```

in $GF(q)$ in a redundant basis $\{\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}}, 1\}$. For each i, the inner for-loop $A \leftarrow A\alpha^{q^i}$ runs r_i times. Therefore the total number of multiplications $A \leftarrow A\alpha^{q^i}$ is $\sum_{i=0}^{n-1} r_i \le (q-1)n$. It follows that one can compute α^r by $(k-1)(n+1)(q-1)n = (k-1)(q-1)n(n+1)$ additions in $GF(q)$ in a redundant basis, whose complexity is $O((k-1)(q-1)n^2)$ for fixed k and q. $\square$

If the above theorem should have any application, it must be guaranteed that the Gauss period α is a primitive element in $GF(q^n)$, or at least is of high order. This is not always satisfied. For example, a Gauss period α of type I or $(n, 1)$ is never a primitive element since $\alpha^{n+1} = 1$ and $n + 1 << q^n$. However, various computational results imply that a Gauss period α of type k, $k \ge 2$, over $GF(q)$ is very often primitive, and even in the cases that α is not primitive, it usually has a very high multiplicative order. For example, it is known [5] that, among the 177 values of $n \le 1000$ for which a Gauss period α of type II or $(n, 2)$ over $GF(2)$ exists, α is a primitive element for 146 values of n. The same table in [5] implies that a Gauss period of type (n, k) over $GF(2)$ is also very often primitive for $k \ge 3$. In the table, it is shown that for approximately 1050 values of $2 \le n \le 1200$, there is a primitive Gauss period of type k for some k, and in many cases, one can choose $k < 20$. A theorem supporting this experimental evidence is obtained by Gathen and Shparlinski [9], where it is shown that a Gauss period of type II in $GF(q^n)$ has order at least $2^{\sqrt{2n}-2}$ for infinitely many n. On the other hand, Feisel et al. [8] extend the notion of Gauss periods to obtain general Gauss periods which, in many cases, have low computational cost in multiplication than the usual Gauss periods.

3 Signed Digit Representation and Exponentiation in $GF(q^n)$ Using a Type II ONB

3.1 Binary NAF and Balanced Ternary Form

Every integer $0 \le s < 2^n$ has a unique binary expansion $s = \sum_{i=0}^{n-1} s_i 2^i$ with $s_i = 0, 1$. This binary representation is particularly useful when one computes an exponentiation in $GF(2^n)$ using a normal basis because the Frobenius map is free in this case. A nonzero $s_i = 1$ contributes one multiplication $A \leftarrow A\alpha$ during the computation of α^s in Table 1. The average number of nonzero bits of arbitrary $0 \le s < 2^n$ is $\frac{n}{2}$. On the other hand, one may express $s = \sum_{i=0}^{n} s_i 2^i$ with $s_i = 0, \pm 1$ as follows. From the usual binary expansion of s, if there is a consecutive nonzero bits such as $2^{j+1} + 2^j$ (i.e. $\cdots 11 \cdots$), then replace it by $2^{j+2} - 2^j$ (i.e. $\cdots 10\bar{1} \cdots$ with $\bar{1} = -1$). The resulting expression is unique and it is called a nonadjacent form (NAF) of s. In fact, this expression is completely determined by the following condition,

Definition 2. *A nonadjacent form (NAF) of an integer s is a representation $s = \sum_{i=0} s_i 2^i$ with $s_i = 0, \pm 1$ such that $s_i s_{i+1} = 0$ for all $i \ge 0$.*

It is well known [12,14] that every positive integer has a unique NAF and the average number of nonzero bits of the integer $0 \le s < 2^n$ is $\frac{n}{3}$. This NAF representation of an integer can be generalized [13,14] to the case of radix m ($m \ge 2$) representation. It is shown [12,13,14] that the length n NAF of an integer with radix m representation has the expected number of nonzero digits $n\frac{m-1}{m+1}$. Since the expected number of nonzero digits of an integer with usual radix m representation is $n\frac{m-1}{m}$, the improvement made by using NAF is not so spectacular for the case $m \ge 3$. Therefore for the finite field $GF(3^n)$ with characteristic three, we will express the exponents using so called the balanced ternary representation [16].

Definition 3. *A balanced ternary form of an integer s is a representation $s = \sum_{i=0} s_i 3^i$ with $s_i = 0, \pm 1$.*

When $0 \le s < 3^n$, from the usual ternary representation of s, if there is a digit 2 (i.e. $2 \cdot 3^j$ for some j), then replace it by $3^{j+1} - 3^j$. The resulting expression can be written as $s = \sum_{i=0}^{n} s_i 3^i$ with $s_i = 0, \pm 1$. Every integer s has a unique balanced ternary representation and it can be shown as follows. If s has two different expressions $s = \sum_{i=0} s_i 3^i = \sum_{i=0} s_i' 3^i$, letting j be the least nonnegative integer satisfying

$$s_j \ne s_j', \quad \text{and } s_i = s_i',\ 0 \le i < j, \tag{6}$$

we have $\sum_{i=j} s_i 3^i = \sum_{i=j} s_i' 3^i$. Dividing this equality by 3^j and taking modulo 3, we get $s_j \equiv s_j' \pmod 3$. This is a contradiction because the equation (6) says $s_j \ne s_j'$ and $s_j, s_j' = 0, \pm 1$.

3.2 Efficient Exponentiation Using a Type II ONB with NAF and Balanced Ternary Form

In section 2, we showed that one has the quadratic time complexity $O((k-1)(q-1)n^2)$ for exponentiation of a Gauss period α of type k. To use a signed digit representation for the exponent of α effectively, it is necessary to compute $A\alpha^{-1}$ for arbitrary $A \in GF(q^n)$ and the cost of the computation should be less or equal to the cost of computing $A\alpha$. For arbitrary Gauss periods of type k, this is not an easy problem since one needs to find the inverse of the multiplication matrix determined by the Gauss period of type k and the usual method of Gaussian elimination requires too much time. On the other hand, when $k = 2$, one has a nice basis called the palindromic representation [6,7] which is a permutation of a normal basis. Recall that there exists a Gaussian normal basis $\{\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}}\}$ of type II in $GF(q^n)$ if and only if $gcd(2n/ord_p q, n) = 1$, i.e. $ord_p q = 2n$ or $ord_p q = n$ with $n = odd$. This is so called a type II optimal normal basis (ONB) and it is easy to see that our definition is equivalent to the following definition used in some other literature [6,7].

Definition 4. *Let $GF(q^n)$ be a finite field with q^n elements where $2n + 1 = p$ is a prime. Suppose that either ($\star$) q is a primitive root $\pmod p$ or ($\star\star$) -1 is a quadratic nonresidue $\pmod p$ and q generates all the quadratic residues $\pmod p$. Then the basis of the form $\{\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}}\}$ is called a type II optimal normal basis (ONB) in $GF(q^n)$, where $\alpha = \beta + \beta^{-1}$ and β is a primitive pth root of unity in $GF(q^{2n})$.*

Using the assumptions in the previous definition, one finds easily

$$\alpha^{q^s} = (\beta + \beta^{-1})^{q^s} = \beta^{q^s} + \beta^{-q^s} = \beta^t + \beta^{-t}, \tag{7}$$

where $0 < t < p = 2n + 1$ with $q^s \equiv t \pmod p$. Moreover, replacing t by $p - t$ if $n+1 \le t \le 2n$, we find that $\{\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}}\}$ and $\{\beta+\beta^{-1}, \beta^2+\beta^{-2}, \cdots, \beta^n + \beta^{-n}\}$ are the same sets. That is, $\alpha^{q^s}, 0 \le s \le n-1$ is just a permutation of $\beta^s + \beta^{-s}, 1 \le s \le n$. This observation leads to the following definition.

Definition 5. *Let β be a primitive pth ($p = 2n + 1$) root of unity in $GF(q^{2n})$. For each integer s, define α_s as*

$$\alpha_s = \beta^s + \beta^{-s}. \tag{8}$$

Then for each integer s and t, we easily find

$$\alpha_s \alpha_t = (\beta^s + \beta^{-s})(\beta^t + \beta^{-t}) = \alpha_{s-t} + \alpha_{s+t}. \tag{9}$$

In other words, a multiplication of two basis elements has a simple expression as a sum of two basis elements.

Lemma 6. *We have $\alpha_0 = 2$ and $\alpha_s = \alpha_t$ if and only if $s \pm t \equiv 0 \pmod{2n+1}$.*

Proof. It is clear that $\alpha_0 = \alpha^0 + \alpha^{-0} = 2$ and $\alpha_s = \beta^s + \beta^{-s}$ depends only on the residue classes of $s \pmod{2n+1}$ because $\beta^{2n+1} = 1$. Moreover $\alpha_{2n+1-s} = \beta^{2n+1-s} + \beta^{-(2n+1-s)} = \beta^{-s} + \beta^s = \alpha_s$. $\square$

From now on, we are mainly interested in the field arithmetic using the basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ or $\{\alpha_1, \alpha_2, \cdots, \alpha_n, 1\}$ depending on whether $q = even$ or odd. To deduce a low complexity field arithmetic, the relation (9) with arbitrary indices s and t will be frequently used. Though our method is applicable to all finite fields with small characteristic, we will focus on the cases with characteristic two and three since these are the most widely used small characteristic finite fields.

Let us consider the finite field $GF(q^n)$ with $q = 2$ or 3 and suppose that we have a basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ for $GF(q^n)$ where $\alpha = \alpha_1 = \beta + \beta^{-1}$ is a type II optimal normal element. Let $A = \sum_{i=1}^n a_i\alpha_i$ and $B = \sum_{i=1}^n b_i\alpha_i$ be in $GF(q^n)$ such that

$$B = A\alpha^{-1}. \tag{10}$$

Then using the equation (9) and Lemma 6, we get

$$\begin{aligned} A = B\alpha = \sum_{i=1}^n b_i\alpha_i\alpha_1 &= \sum_{i=1}^n b_i(\alpha_{i-1} + \alpha_{i+1}) \\ &= b_1(2 + \alpha_2) + b_2(\alpha_1 + \alpha_3) + \cdots + b_{n-1}(\alpha_{n-2} + \alpha_n) + b_n(\alpha_{n-1} + \alpha_n) \\ &= 2b_1 + b_2\alpha_1 + (b_1 + b_3)\alpha_2 + (b_2 + b_4)\alpha_3 + \cdots \\ &\qquad + (b_{n-2} + b_n)\alpha_{n-1} + (b_{n-1} + b_n)\alpha_n \\ &= 2b_1 + b_2\alpha_1 + (b_{n-1} + b_n)\alpha_n + \sum_{i=2}^{n-1}(b_{i-1} + b_{i+1})\alpha_i. \end{aligned} \tag{11}$$

If we use a redundant basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n, 1\}$ with $A = \sum_{i=1}^n a_i\alpha_i + a_0$ and $B = \sum_{i=1}^n b_i\alpha_i + b_0$, then the same computation shows that

$$\begin{aligned} A = B\alpha = \sum_{i=1}^n b_i\alpha_i\alpha_1 + b_0\alpha_1 &= \sum_{i=1}^n b_i(\alpha_{i-1} + \alpha_{i+1}) + b_0\alpha_1 \\ &= 2b_1 + (b_{n-1} + b_n)\alpha_n + \sum_{i=1}^{n-1}(b_{i-1} + b_{i+1})\alpha_i. \end{aligned} \tag{12}$$

Theorem 7. *Using the basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, one needs $n-1$ additions in $GF(2)$ to compute $B = A\alpha^{-1}$ in $GF(q^n)$ with $q = 2$, and using the redundant basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n, 1\}$, one needs n additions in $GF(3)$ to compute $B = A\alpha^{-1}$ in $GF(q^n)$ with $q = 3$.*

Proof. First, let us consider the case $q = 2$. From (11), we get $2b_1 = 0$ and

$$a_1 = b_2, a_n = b_{n-1} + b_n, \quad \text{and } a_i = b_{i-1} + b_{i+1} \quad \text{for } 2 \le i \le n-1. \tag{13}$$

Therefore using the above relations, we have the values b_i recursively for even indices i as

$$b_2 = a_1, b_4 = b_2 + a_3, b_6 = b_4 + a_5, \cdots, b_{2s} = b_{2s-2} + a_{2s-1}, \tag{14}$$

where $s = \lfloor \frac{n}{2} \rfloor$, i.e. $n = 2s$ or $2s+1$. If $n = 2s$, then using (13) again, we get the values of b_i recursively for odd indices i as

$$b_{n-1} = b_n + a_n, b_{n-3} = b_{n-1} + a_{n-2}, b_{n-5} = b_{n-3} + a_{n-4}, \cdots, b_1 = b_3 + a_2, \tag{15}$$

and if $n = 2s+1$, i.e. if $2s = n-1$, then we have the values of b_i for odd i as

$$b_n = b_{n-1} + a_n, b_{n-2} = b_n + a_{n-1}, b_{n-4} = b_{n-2} + a_{n-3}, \cdots, b_1 = b_3 + a_2. \tag{16}$$

Thus the computation of b_i $(i \neq 2,\ 1 \leq i \leq n)$ needs one addition with $b_2 = a_1$. Consequently the total number of necessary additions in $GF(2)$ to compute $A\alpha^{-1}$ is $n-1$.

Now let us consider the case $q = 3$. We will use the redundant basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n, 1\}$ in this case. Writing $A, B \in GF(3^n)$ as $A = \sum_{i=1}^n a_i\alpha_i + a_0$ and $B = \sum_{i=1}^n b_i\alpha_i + b_0$, the equation (12) says

$$a_0 = -b_1, a_n = b_{n-1} + b_n, \quad \text{and } \ a_i = b_{i-1} + b_{i+1} \quad \text{for } \ 1 \leq i \leq n-1. \tag{17}$$

Thus using the above relations, we may compute b_i for odd indices i as

$$b_1 = -a_0, b_3 = -b_1 + a_2, b_5 = -b_3 + a_4, \cdots, b_{2s+1} = -b_{2s-1} + a_{2s}, \tag{18}$$

where $s = \lfloor \frac{n-1}{2} \rfloor$, i.e. $n = 2s+1$ or $2s+2$. If $n = 2s+1$, then using (17) again,

$$b_{n-1} = -b_n + a_n, b_{n-3} = -b_{n-1} + a_{n-2}, b_{n-5} = -b_{n-3} + a_{n-4}, \cdots, b_0 = -b_2 + a_1, \tag{19}$$

and if $n = 2s+2$, i.e. if $2s+1 = n-1$, we get

$$b_n = -b_{n-1} + a_n, b_{n-2} = -b_n + a_{n-1}, b_{n-4} = -b_{n-2} + a_{n-3}, \cdots, b_0 = -b_2 + a_1. \tag{20}$$

Therefore the computation of b_i $(i \neq 1,\ 0 \leq i \leq n)$ needs one addition with $b_1 = -a_0$. Consequently the total number of necessary additions in $GF(3)$ to compute $A\alpha^{-1}$ is n using the redundant basis $\{\alpha_1, \alpha_2, \cdots, \alpha_n, 1\}$ in $GF(3^n)$. □

Based on Theorem 7, we propose a new exponentiation algorithm with signed digit representation of the exponents as follows.

Table 2. New exponentiation algorithm using signed digit representation with a type II ONB in $GF(q^n)$.

```
Input: $r = \sum_{j=0}^n r_j q^j$ with $r_j = 0, \pm 1$
Output: $\alpha^r$
$A \leftarrow 1$
for ($i = n$ to $0$ ; $i - -$)
   $A \leftarrow A^q \alpha^{r_i}$
end for
```

The above algorithm is just a simple form of q-ary window method which computes

$$\alpha^r = \alpha^{\sum_{i=0}^n r_i q^i} = (\cdots(((\alpha^{r_n})^q \alpha^{r_{n-1}})^q \alpha^{r_{n-2}})^q \cdots)^q \alpha^{r_0}. \tag{21}$$

Compared with the algorithm in Table 1, our algorithm has only one for-loop and the computational cost is significantly reduced from the original algorithm of Gao et al. [5].

4 Comparison of Our Algorithm with the Method of Gao et al. and the Polynomial Basis Method

The complexity of one exponentiation using a type k Gaussian normal basis in the algorithm of Table 1 is bounded by $(k-1)(q-1)n(n+1)$ additions in $GF(q)$. For the case of a type II ONB (i.e. $k=2$), it is $(q-1)n(n+1)$ additions in $GF(q)$. Therefore in $GF(2^n)$, since the expected number of nonzero bits of the exponent r is $\frac{n}{2}$ with the usual binary representation, the average number of $GF(2)$-additions to compute $\alpha^r \in GF(2^n)$ using the method of Gao et al. is $\frac{1}{2}n(n+1) \approx \frac{1}{2}n^2$. For $GF(3^n)$, since the expected number of nonzero digits of the exponent r is $\frac{2}{3}n$ with equal probability of appearing 1 and 2 in the exponent, the expected value of the sum of all digits of the exponent r is n, which is the number of the iterations of the inner for-loop in Table 1. Therefore the average number of $GF(3)$-additions to compute $\alpha^r \in GF(3^n)$ using the method in [5] is $n(n+1) \approx n^2$.

Now let us evaluate the complexity of our proposed method. It is trivial, from the equations (11) and (12), to show that the number of necessary $GF(q)$-additions to compute $A\alpha$ for any $A \in GF(q^n)$ is also $n-1$ if $q=2$ and is n if $q=3$. For $GF(2^n)$, we use a nonadjacent form (NAF) of the exponent r and the expected number of nonzero bits of r with NAF is $\frac{n}{3}$. Thus, in view of Theorem 7, the average number of $GF(2)$-additions to compute $\alpha^r \in GF(2^n)$ using our algorithm is $\frac{1}{3}n(n-1) \approx \frac{1}{3}n^2$. For $GF(3^n)$, the expected number of nonzero digits (either 1 or -1) of the exponent r with balance ternary representation is $\frac{2}{3}n$ and the number of necessary $GF(3)$-additions to compute $\alpha^r \in GF(3^n)$ using our method with Theorem 7 is $\frac{2}{3}n \cdot n = \frac{2}{3}n^2$. Therefore for both cases of $GF(2^n)$ and $GF(3^n)$, our algorithm improves the method Gao et al. [5] by the factor of 33 percent in terms of the necessary additions in $GF(q)$ for one exponentiation in $GF(q^n)$.

We also claim that our method is superior to the method of using a polynomial basis with signed digit representation. Intuitively, the reason is that the qth Frobenius map $A \to A^q \in GF(q^n)$ is not free in a polynomial basis representation and one should repeatedly take the qth power n times to get the result of one exponentiation, while all these operations are free in a normal basis. The complexity of the Frobenius map is heavily dependent on the weight (the number of nonzero coefficients of the polynomial) of the given irreducible polynomial $f(X) \in GF(q)[X]$. It is not difficult to see that one has the lowest complexity exponentiation algorithm when one use a trinomial $f(X) = X^n + aX^k + b$.

For a binary field $GF(2^n)$, Wu [10] gave an exact estimation of the number of necessary additions in $GF(2)$ for one squaring (i.e. $q=2$) using a trinomial $f(X) = X^n + X^k + 1 \in GF(2)[X]$ with $1 \leq k < \frac{n}{2}$. That is, when $nk = odd$, the number of necessary $GF(2)$-additions is $\frac{1}{2}n$, and when $nk = even$, it is $\frac{3}{4}n$. On the other hand, it can be derived easily [11] that the operation $A \leftarrow A\alpha^{\pm 1}$ needs just one addition, where α is a zero of the trinomial $f(X)$. Thus using a trinomial basis with signed binary method in [11], one gets the complexity of one exponentiation $\alpha^r \in GF(2^n)$ as $\frac{1}{2}n \cdot n + 1 \cdot \frac{n}{3} \approx \frac{1}{2}n^2$ additions if $nk = odd$, and $\frac{3}{4}n \cdot n + 1 \cdot \frac{n}{3} \approx \frac{3}{4}n^2$ additions if $nk = even$.

Table 3. Comparison of the number of necessary additions in $GF(q)$ for one exponentiation.

Basis	[5] type II ONB	[10,11] trinomial	This paper type II ONB
In $GF(2^n)$	$\frac{1}{2}n^2$	$\frac{1}{2}n^2$ or $\frac{3}{4}n^2$	$\frac{1}{3}n^2$
In $GF(3^n)$	n^2	——	$\frac{2}{3}n^2$

Table 3 shows that our proposed exponentiation algorithm is superior to other algorithms using either a normal or a polynomial basis. The number of necessary $GF(q)$-additions of our method is 33 percent reduced from the method with a type II ONB of Gao et al. [5], and also at least 33 percent reduced from the method in [10,11] using a signed bit representation with a trinomial basis $f(X) = X^n + X^k + 1$ for $GF(2^n)$.

We conclude this section by mentioning that the base change between $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ and $\{\alpha_1, \alpha_2, \cdots, \alpha_n, 1\}$ contributes a negligible cost. This can be verified as follows. From Definition 5, using $\beta^{2n+1} = 1$, we have

$$\begin{aligned}\alpha_1 + \alpha_2 + \cdots + \alpha_n &= \beta + \beta^{-1} + \beta^2 + \beta^{-2} + \cdots + \beta^n + \beta^{-n} \\ &= \beta + \beta^2 + \cdots + \beta^n + \beta^{n+1} + \beta^{n+2} + \cdots + \beta^{2n} \\ &= \frac{\beta(\beta^{2n} - 1)}{\beta - 1} = \frac{1 - \beta}{\beta - 1} = -1.\end{aligned} \tag{22}$$

Therefore using the above relation, the redundant expression of A can be transformed into the expression with respect to the standard basis as follows,

$$A = \sum_{i=1}^{n} a_i\alpha_i + a_0 = \sum_{i=1}^{n} a_i\alpha_i - a_0\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n}(a_i - a_0)\alpha_i. \tag{23}$$

Therefore one needs just n additions in $GF(q)$ for the base change, which is negligible compared with the quadratic time complexity $O(n^2)$ of one exponentiation in $GF(q^n)$.

5 Distribution of Type II Optimal Normal Bases

In previous section, we showed that our method of using a Gaussian normal basis of type II outperforms the method with a trinomial basis for an exponentiation in $GF(2^n)$, because the Frobenius map contributes a significant portion of the computation delay in a polynomial basis. It is expected that the same result holds for other low characteristic finite fields $GF(q^n)$ when we use similar techniques in [10,11]. It should be mentioned that one advantage of using a polynomial basis is that there are more irreducible (and primitive) trinomials than type II Gaussian normal bases [7,15]. However, a type II ONB appears frequently enough for cryptographic purposes (like secure pseudo random number generators). For example, a table in [7] shows that the number of $n \leq 2000$ for which a type II ONB exists in $GF(2^n)$ is 324. It is shown [7] that a type II ONB exists in $GF(2^n)$ when $n = 2, 3, 5, 6, 9, 11, 14, 18, 23, 26, 29, 30, 33, 35, 39, 41, 50, 51, 53, 65, 69, \cdots$. On the other hand, the same table shows that the number of $n \leq 2000$ for which a type I ONB (i.e. a Gaussian normal basis of type I) exists is 118. We have a type I ONB when $n = 2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82, 100, 106, \cdots$. Since one has the ratio $324/118 \approx 2.75$, it can be said that, for binary fields, a type II ONB appears 2.75 times more frequently than a type I ONB, though this is not a really correct statement from a mathematical point of view. However, we may give a satisfactory argument on the the density of n for which a type I ONB exists in $GF(q^n)$ using the widely believed and one of the most important conjectures in mathematics, the generalized 'Riemann Hypothesis' [20], and from which we may have a rough estimation on the distribution of n for which a type II ONB exists in $GF(q^n)$.

Let $a \neq 0, \pm 1$ be an integer which is not an rth power for any $r > 1$. Define $N_a(x)$ be the number of primes $p \leq x$ for which a is a primitive root $\pmod p$. In 1927, E. Artin conjectured that $N_a(x)$ is related to the following asymptotic formula,

$$\lim_{x\to\infty} \frac{N_a(x)}{x/\ln x} = C(a), \tag{24}$$

where $C(a) = C_a C$ is a constant depending on a. That is, writing $a = a'b^2$ with a' square free, we have the constant C_a depending on a,

$$C_a = 1 \text{ if } a' \not\equiv 1 \pmod 4, \quad C_a = 1 - \mu(a') \prod_{q|a'} \frac{1}{q^2 - q - 1} \text{ if } a' \equiv 1 \pmod 4, \tag{25}$$

where $\mu(a')$ is the usual Möbius function and the product runs through all primes q dividing a'. The Artin constant C is expressed as

$$C = \prod_q (1 - \frac{1}{q^2 - q}) = 0.3739558 \cdots, \tag{26}$$

where the product runs through all primes. This conjecture was proved by Hooley [17] using the generalized Riemann Hypothesis. Later, a weaker form of Artin's conjecture was proved by Gupta and Murty [18] and also by Heath-Brown [19] without using Riemann Hypothesis. However, at this moment, there is no known single example of a for which the conjecture of Artin is proved without any extra assumption or hypothesis. Based on extensive computational evidence, it is generally believed that Riemann Hypothesis and also Artin's conjecture are true. Therefore, to apply the conjecture to our case, let $a = 2$ or 3. Then, from the equation (25), we have $C_a = 1$ and thus $C(a) = C = 0.3739558 \cdots$. Consequently by using the well known 'Prime Number Theorem' [20] saying

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1, \tag{27}$$

where $\pi(x)$ is the number of primes $\leq x$, we conclude

$$\lim_{x\to\infty} \frac{N_a(x)}{\pi(x)} = \lim_{x\to\infty} \frac{N_a(x)}{x/\ln x} \cdot \frac{x/\ln x}{\pi(x)} = 0.3739558\cdots. \tag{28}$$

In other words, a $(= 2$ or $3)$ is a primitive root $\pmod p$ for approximately $37.39558\cdots \approx 37.4$ percent of all primes p. And for those primes p, $n = p - 1$ gives the values of n for which a type I ONB exists.

Recall that for the case of a binary field, the table in [7] implies that there are 2.75 times more $n \le 2000$ for which a type II ONB exists in $GF(2^n)$ than the number of $n \le 2000$ for which a type I ONB exists in $GF(2^n)$. Therefore letting $D(x)$ be the number of $n \le x$ for which a type II ONB exists in $GF(2^n)$, we get the following,

$$\lim_{x\to\infty} \frac{D(x)}{\pi(x)} = \lim_{x\to\infty} \frac{D(x)}{N_2(x)} \cdot \frac{N_2(x)}{\pi(x)} \approx 2.75 \cdot 0.374 \approx 1.03. \tag{29}$$

An obvious implication is that the field size n for which a type II ONB exists in $GF(2^n)$ appears as frequently as a prime number appears in the set of natural numbers.

6 Conclusions

We proposed an efficient exponentiation algorithm using a Gaussian normal basis of type II with signed digit representation of the exponents for small characteristic finite fields $GF(q^n)$. Our method uses a binary nonadjacent form of the exponent if $q = 2$ and a balanced ternary representation if $q = 3$. Though it is supposed that computing $A\alpha^{-1}$ is not so easy when one use a normal basis, we showed that it is as efficient as computing $A\alpha$ when we have a Gaussian normal basis of type II. We also gave a rough estimation of the distribution of type II optimal normal elements. A computational result implies that our algorithm is significantly faster than previously proposed exponentiation algorithms using either a normal basis or a polynomial basis for a finite field $GF(q^n)$ with small characteristic.

Acknowledgements

This work was supported by grant No. R05-2003-000-11325-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

1. E.F. Brickell, D.M. Gordon, K.S. McCurley, and D.B. Wilson, “Fast exponentiation with precomputation,” *Eurocrypt 92, Lecture Notes in Computer Science*, vol. 658, pp. 200–207, 1992.
2. C.H. Lim and P.J. Lee, “More flexible exponentiation with precomputation,” *Crypto 94, Lecture Notes in Computer Science*, vol. 839, pp. 95–107, 1994.

3. P. de Rooij, "Efficient exponentiation using precomputation and vector addition chains," *Eurocrypt 94, Lecture Notes in Computer Science*, vol. 950, pp. 389–399, 1994.
4. S. Gao, J. von zur Gathen, and D. Panario, "Gauss periods and fast exponentiation in finite fields," *Latin 95, Lecture Notes in Computer Science*, vol. 911, pp. 311–322, 1995.
5. S. Gao, J. von zur Gathen, and D. Panario, "Orders and cryptographical applications," *Math. Comp.*, vol. 67, pp. 343–352, 1998.
6. S. Gao and S. Vanstone, "On orders of optimal normal basis generators," *Math. Comp.*, vol. 64, pp. 1227–1233, 1995.
7. A.J. Menezes, I.F. Blake, S. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publisher, 1993.
8. S. Feisel, J. von zur Gathen, M. Shokrollahi, "Normal bases via general Gauss periods," *Math. Comp.*, vol. 68, pp. 271–290, 1999.
9. J. von zur Gathen and I. Shparlinski, "Orders of Gauss periods in finite fields," *ISAAC 95, Lecture Notes in Computer Science*, vol. 1004, pp. 208–215, 1995.
10. H. Wu, "On complexity of polynomial basis squaring in $\mathbb{F}_{2^m}$," *SAC 00, Lecture Notes in Computer Science*, vol. 2012, pp. 118–129, 2001.
11. H. Wu and M.A. Hasan, "Efficient exponentiation of a primitive root in $GF(2^m)$," *IEEE Trans. Computers*, vol. 46, pp. 162–172, 1997.
12. D.M. Gordon, "A survey of fast exponentiation methods," *J. Algorithms*, vol. 27, pp. 129–146, 1998.
13. S. Arno and F.S. Wheeler, "Signed digit representation of minimal hamming weight" *IEEE Trans. Computers*, vol. 42, pp. 1007–1010, 1993.
14. J.H. van Lint, *Introduction to Coding Theory, 3rd*, Springer-Verlag, 1999.
15. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handboook of Applied Cryptography*, CRC Press, 1996.
16. D.E. Knuth, *The Art of Computer Programming, 3rd* : Seminumerical algorithms, vol. II, Addison-Wesley, 2001.
17. C. Hooley, "On Artin's conjecture," *J. reine angew. Math.*, vol. 225, pp. 209–220, 1967.
18. R. Gupta and M. Ram Murty, "A remark on Artin's conjecture," *Inventiones Math.*, vol. 78, pp. 127–130, 1984.
19. D. Heath-Brown, "Artin's conjecture for primitive roots," *Quart. J. Math.*, vol. 37, pp. 27–38, 1986.
20. G. Tenenbaum and M.M. France, *The Prime Numbers and Their Distribution, translated by P.G. Spain*, Ameriacn Mathematical Society, 2000.

Novel Efficient Implementations of Hyperelliptic Curve Cryptosystems Using Degenerate Divisors

Masanobu Katagi[1], Izuru Kitamura[1], Toru Akishita[1], and Tsuyoshi Takagi[2]

[1] Sony Corporation, 6-7-35 Kitashinagawa Shinagawa-ku, Tokyo, 141-0001 Japan
{Masanobu.Katagi,Izuru.Kitamura}@jp.sony.com, akishita@pal.arch.sony.co.jp
[2] Technische Universität Darmstadt, Fachbereich Informatik,
Alexanderstr. 10, D-64283 Darmstadt, Germany
takagi@informatik.tu-darmstadt.de

Abstract. It has recently been reported that the performance of hyperelliptic curve cryptosystems (HECC) is competitive to that of elliptic curve cryptosystems (ECC). However, it is expected that HECC still can be improved due to their mathematically rich structure. We consider here the application of degenerate divisors of HECC to scalar multiplication. We investigate the operations of the degenerate divisors in the Harley algorithm and the Cantor algorithm of genus 2. The timings of these operations are reported. We then present a novel efficient scalar multiplication method using the degenerate divisors. This method is applicable to cryptosystems with fixed base point, e.g., ElGamal-type encryption, sender of Diffie-Hellman, and DSA. Using a Xeon processor, we found that the double-and-add-always method using the degenerate base point can achieve about a 20% increase in speed for a 160-bit HECC. However, we mounted an timing attack using the time difference to designate the degenerate divisors. The attack assumes that the secret key is fixed and the base point can be freely chosen by the attacker. Therefore, the attack is applicable to ElGamal-type decryption and single-pass Diffie-Hellman – SSL using a hyperelliptic curve could be vulnerable to the proposed attack. Our experimental results show that one bit of the secret key for a 160-bit HECC can be recovered by calling the decryption oracle 500 times.

Keywords: hyperelliptic curve cryptosystem, scalar multiplication, timing attack, degenerate divisor, efficient computation, SSL

1 Introduction

In 1989, Koblitz proposed hyperelliptic curve cryptosystems (HECC) [Kob89]. HECC has the advantage of shorter operand length than elliptic curve cryptosystems (ECC), and it has been expected that HECC can attain a faster encryption compared to ECC due to their rich algebraic structure. Recently some efficient addition formulae of HECC have been proposed (see, for example, [Lan02a-c]), and implementation in software shows the performance of HECC to be competitive to that of ECC [PWG$^+$03,Ava03b].

When we implement a cryptosystem in a real security system, we have to prepared for side channel attacks such as timing attack [Koc96] and power analysis [KJJ99]. A simple timing attack on HECC can reveal the hamming weight of the secret scalar by measuring the computation time of the scalar multiplication. The timing attack using the final subtraction of Montgomery multiplication [Sch00,Sch02,SKQ01] is also applicable to HECC. Other possible attacks use exceptional procedures of the addition formula [AT03,Ava03a,Gou03,IT03]. The addition formula of HECC involves many exceptional procedures, so that we have many possibilities to use them in a timing attack. Note that these attacks assume that the base point D can be freely chosen by the attacker and the secret scalar d is fixed: we can apply our attack to HEC ElGamal-type decryption and single-pass HEC Diffie-Hellman but not HEC DSA.

In this paper, we focus on the degenerate divisors of HECC, which has a lower weight than g, where g is the genus of the underlying curve. We investigate possible degenerate divisors appearing in the Harley algorithm [Har00a,Har00b] and Cantor algorithm [Can87,Kob89] of genus 2. The addition formulae to compute these degenerate divisors have a different running time than the standard divisors. We estimated the time required to compute the doubling and addition using the degenerate divisors. In general, the probability that these procedures will have to be used in the addition formula is $\mathcal{O}(1/q)$, where q is the definition field of HECC [Nag00]. However, in some cryptographic protocols we can use the degenerate divisor in both positive and negative ways.

As a positive application, we present an algorithm to efficiently perform scalar multiplication using a fixed base point. The degenerate divisor, which allows faster hyperelliptic curve addition in the Harley algorithm, is used for the base point. We found that the security of the underlying discrete logarithm problem is not decreased due to the random self reducibility. We also found that a randomly chosen divisor with weight 1 achieves about 20% faster scalar multiplication using the double-and-add-always method for a 160-bit HECC. Efficient scalar multiplication with a fixed base point can be applied to ElGamal-type encryption, sender of Diffie-Hellman, and DSA.

As a negative application, we propose a timing attack on HECC using the degenerate divisors. The algorithm of recovering the secret key draws on previous work [AT03,Ava03a,Gou03,IT03]. We will consider the attack on the hyperelliptic curve with genus 2, especially the Harley algorithm and Cantor algorithm. Several explicit exceptional procedures suitable for the timing attack in the setting are investigated. We examine the exceptional procedures which arise from the divisors with weight 1, whose computational time is faster than the ordinary one in the Harley algorithm. Then we show how to apply these procedures to the timing attack. The exceptional procedures appear with negligible probability in the scalar multiplication for a randomly chosen base point. Thus the exceptional procedures cause a timing difference in the scalar multiplication from ordinary operation. The proposed timing attack analyses the timing difference with many sampling numbers. Our experiment on a Xeon processor using a 160-bit HECC shows that the timing difference is about 0.02 ms for the exceptional case of the

Harley algorithm. The difference is quite small, so that we apply the sampling technique proposed by Dhem et al. [DKL$^+$98]. We consider three different types of timing: the timing with a randomly chosen divisor, timing with divisor D_b that brings about an exceptional procedure if the supposed bit is $b = 0, 1$. If b is correct, the timing difference from the random divisor becomes larger than otherwise. Our experiment successfully recovered one bit of secret key with 500 samples for a 160-bit HECC.

This paper is organized as follows: In Section 2, we describe the basic properties of HECC. In Section 3, we review the side channel attacks. In Section 4, the degenerate divisors are investigated, and we present an efficient scalar multiplication using the degenerate divisors. In Section 5, we propose the new timing attack of HECC and show the experimental results of genus 2 HECC.

2 Hyperelliptic Curve Cryptosystems

In this section we review hyperelliptic curve cryptosystems related to this paper.

2.1 Hyperelliptic Curve

Let g be a positive integer, and let $K = \mathbb{F}_q$ be a finite field of characteristic p, $q = p^n$ where n is a positive integer. Hyperelliptic curve C of genus g over $\mathbb{F}_q$ is defined by equation $y^2 + h(x)y = f(x)$, where $f(x)$ is a monic polynomial over $\mathbb{F}_q$ with degree $2g+1$ in x. $h(x)$ is a polynomial over $\mathbb{F}_q$ with $\deg h \leq g$ for even characteristic and 0 for odd characteristic. Let $P_i = (x_i, y_i)$ be a rational point on curve C and P_∞ be a point at infinity. The inverse of $P = (x, y)$ is the point $-P = (x, -y - h(x))$. We define point P as satisfying $P = -P$ as a ramification point.

A divisor $D = \sum m_i P_i$, $m_i \in \mathbb{Z}$, is defined as a formal sum of points P_i on C. The set $\mathbb{D}$ of the divisors form an Abelian group. The degree(weight) of a divisor D is defined as $\sum_i m_i$, and we denote it by $w(D)$. The set $\mathbb{D}^0$ of the degree zero divisors is a subgroup of $\mathbb{D}$. The divisor of a rational function on C, called the principal divisor, is a finite formal sum of the zeros and poles. The set $\mathbb{P}$ of principal divisors is a subgroup of $\mathbb{D}^0$. Denote by $J_c(\mathbb{F}_q)$ the Jacobian variety of C defined over $\mathbb{F}_q$. The divisor class group $\mathbb{D}^0/\mathbb{P}$ is isomorphic to the Jacobian variety $J_c(\mathbb{F}_q)$. A semi-reduced divisor of a hyperelliptic curve is represented by $D = \sum_i m_i P_i - (\sum_i m_i) P_\infty$, where $m_i \geq 0$ and $P_i \neq -P_j$ for $i \neq j$. Mumford reported that the semi-reduced divisor can be expressed by two polynomials (u, v) of $\mathbb{F}_q[x]$, which satisfy the following conditions [Mum84]:

$$u(x) = \prod_i (x - x_i)^{m_i}, \quad v(x_i) = y_i, \quad \deg v < \deg u, \quad v^2 + hv - f \equiv 0 \bmod u.$$

A semi-reduced divisor with $\deg u \leq g$ is called a reduced divisor. Any divisor class of $J_c(\mathbb{F}_q)$ is uniquely represented by a reduced divisor. Hereafter we denote $D \in J_c(\mathbb{F}_q)$ by a reduced divisor $D = (u, v)$. The unit element of additive group

$J_c(\mathbb{F}_q)$ is $(1,0)$ and the inverse of divisor $D = (u, v)$ is $-D = (u, v + h)$ where the second polynomial is reduced modulo u.

In this paper, we deal with hyperelliptic curves that are suitable for cryptographic purposes, for example, the order of Jacobian $\#J_c(\mathbb{F}_q)$ has only a small cofactor, say c. Based on Hasse-Weil range, the size of $\mathbb{F}_q$ have to satisfy at least $g \log_2 q \approx 2^{160}$. (We call 160-bit HECC.)

2.2 Scalar Multiplication

The basic operation for implementing HECC is the scalar multiplication, which computes $dD = D + \cdots + D$ (d times) for a divisor $D \in J_c(K)$ and integer d. Denote by $(d_{m-1} \cdots d_1 d_0)_2$ the m-bit binary representation of d. The standard method of computing scalar multiplication dD is the following binary method.

Algorithm 1 *Binary Method*

```
Input: $d = (d_{m-1} \cdots d_1 d_0)_2,\ D \in J_c(K),\ (d_{m-1} = 1)$
Output: $dD$
1. $D_1 \leftarrow D$
2. for $i$ from $m-2$ to $0$ do
3.   $D_1 \leftarrow HECDBL(D_1)$
4.   If $d_i = 1$ then $D_1 \leftarrow HECADD(D_1, D)$
5. Return($D_1$)
```

We denote by HECDBL and HECADD hyperelliptic doubling and addition, e.g., $HECDBL(D_1) = 2D_1$ and $HECADD(D_1, D) = D_1 + D$, respectively. The binary method requires $(m-1)$ HECDBL and $(m-1)/2$ HECADD on average for randomly chosen d.

2.3 Addition Formulae

In order to implement a group operation in $J_c(\mathbb{F}_q)$, we deploy addition formulae assembled by the operations of polynomial ring $\mathbb{F}_q[x]$. There are two basic algorithms, namely Cantor algorithm [Can87,Kob89] and Harley algorithm [Har00a,Har00b]. Cantor algorithm is a universal addition formula. It is used for all hyperelliptic curves with any genus g, so we are able to compute both HECDBL and HECADD with one formula (however, the computation times of HECDBL and HECADD are not the same). However, the Cantor algorithm is quite slow due to its versatility. The Harley algorithm aims at efficiently implementing the group operations for a small genus. The arithmetic of HECDBL and HECADD is independently optimized based on their explicit operations.

In the following, we describe the Cantor algorithm [Can87,Kob89]. Let $D_i = (u_i(x), v_i(x)) \in J_c(\mathbb{F}_q)$ be the reduced divisors, $i = 1, 2$. The reduced divisor D_3 of addition $D_1 + D_2$ is computed as follows:

Algorithm 2 *Cantor Algorithm*

```
Input: $D_1 = (u_1, v_1), D_2 = (u_2, v_2)$
Output: $D_3 = (u_3, v_3) = D_1 + D_2$
1. $d = \gcd(u_1, u_2, v_1 + v_2 + h) = s_1u_1 + s_2u_2 + s_3(v_1 + v_2 + h)$
2. $u \leftarrow u_1u_2/d^2$, $v \leftarrow (s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f))/d \bmod u)$
3. while $\deg(u) > g$
      $u' \leftarrow (f - hv - v^2)/u$, $v' \leftarrow -h - v \bmod u'$, $u \leftarrow \mathrm{MakeMonic}(u'), v \leftarrow v'$
4. $u_3 \leftarrow u$, $v_3 \leftarrow v$
5. return $(u_3, v_3)$
```

Step 1 and Step 2 are called the composition part and Step 3 is called the reduction part. The composition part computes the semi-reduced divisor $D = (u, v)$ that is equivalent to D_3. The reduction part finds the reduced divisor $D_3 = (u_3, v_3)$.

The Harley algorithm is an explicit representaition of the Cantor algorithm using the weight classification of the divisors D_1 and D_2. The Harley algorithm includes the most frequent weight classification for addtion or doubling. We denote the most frequent weight classification by *HarleyADD* and *HarleyDBL*, respectively. The *HarleyADD* and *HarleyDBL* satisfy the following conditions.

$$\begin{aligned}&\mathit{HarleyADD} : w(D_1) = g,\, w(D_2) = g,\, w(D_3) = g,\, D_1 \neq D_2,\, \gcd(u_1, u_2) = 1,\\ &\mathit{HarleyDBL} : w(D_1) = g,\, w(D_3) = g,\, D_1 = D_2,\, \gcd(h, u_1) = 1.\end{aligned}$$

3 Side Channel Attacks

In this section we review the side channel attacks which HECC is vulnerable to.

At Crypto'96 Kocher introduced the timing attack (TA), which considers the secret key in cryptographic devices [Koc96]. TA measures the computation time for various inputs and analyzes the difference between these times. For example, if we compute a scalar multiplication dD using the binary method (Algorithm 1), then TA can reveal the hamming weight of secret key d. The standard way to resist this attack is the following double-and-add-always method:

Algorithm 3 *Double-and-Add-Always Method*

```
Input: $d = (d_{m-1} \cdots d_1d_0)_2$, $D \in J_c(K)$, $(d_{m-1} = 1)$
Output: $dD$
1. $D[0] \leftarrow D$
2. for $i$ from $m - 2$ to $0$ do
3.  $D[0] \leftarrow$ HECDBL($D[0]$), $D[1] \leftarrow$ HECADD($D[0]$,$D$), $D[0] \leftarrow D[d_i]$
4. Return($D[0]$)
```

The double-and-add-always method always computes HECADD whether $d_i = 0$ or 1. Therefore, TA cannot compute the bit hamming weight of d. There is another type of timing attack. In the RSA cryptosystem, the attacker can utilize the existence of the final subtraction in Montgomery multiplication [Sch00, Sch02, SKQ01]. The same argument can be applied to HECC.

At Crypto '99 Kocher et al. introduced simple power analysis (SPA) and differential power analysis (DPA) to reveal the secret key by measuring the power consumption of cryptographic devices [KJJ99]. SPA uses only a single observation of the power to obtain information, and DPA uses many observations together with statistical tools. Algorithm 1 is vulnerable to SPA. The operation HECADD is computed only if the corresponding bit is 1, although HECDBL is always computed. HECDBL and HECADD show different power consumption curves because they are different operations, as is described in Section 2. Thus the SPA can detect the secret bits. In order to defend against SPA, we must eliminate the relation between the addition formulas and the bit information. The double-and-add-always method in the previous section can be used to defend against SPA. Even if a scheme is secure against SPA, it might be insecure against DPA. The standard method to resist DPA is randomizing the parameters of the curve [Cor99,JT01]. However, Goubin proposed an extension of DPA to an elliptic curve cryptosystem [Gou03]. He pointed out that the point $(0, y)$ is not fully randomized by the standard countermeasures against DPA. Recently, Avanzi extended his attack to a hyperelliptic curve cryptosystem [Ava03a]. He noted that the divisors with zero coefficient could be used in a Goubin-type attack, in which one of the coefficients of u or v for divisor (u, v) would be zero.

Izu and Takagi proposed an exceptional procedure attack by using the exceptional procedure in the addition formula of ECC [IT03]. The standard addition formula of ECC bring about an exceptional procedure only if either the input or output is infinity point $\mathcal{O}$. The order of elliptic curve $\#E$ is usually chosen such that $\#E$ is the product of a large prime and a very small integer. When scalar d is smaller than the order of elliptic curve $\#E$, the exceptional procedure occurs only if the order of the processing point is small. Thus, we can detect this attack by checking if the base point does not belong to small group before scalar multiplication. Avanzi also mentioned the possibility of extending this attack to hyperelliptic curve cryptosystems [Ava03a]. However, the details of the extended attack require further discussion.

4 Degenerate Divisors of HECC

We assume here that the genus of hyperelliptic curves is equal to 2 and the characteristic is even for the sake of convenience. However, all discussions are applicapble to higher genus > 2 and an odd prime characteristic.

We deal with the divisor with weight 1, and we define it as the degenerate divisor.

Definition 1. *Let C be a hyperelliptic curve over $\mathbb{F}_{2^n}$, let $J_c(\mathbb{F}_{2^n})$ be the Jacobian of curve C. We call reduced divisor $D = (u, v) \in J_c(\mathbb{F}_{2^n})$ degenerate, if the degree of D is smaller than g, namely* $\deg u < g$.

Let $D_1 = (u_1, v_1), D_2 = (u_2, v_2)$ be the reduced divisors of Jacobian $J_c(\mathbb{F}_{2^n})$. Denote by D_3 the addition of $D_1 + D_2$. There is the following possible group operation with degenerate divisors:

$\mathsf{ExHarADD}^{2+2\to 1}$: $w(D_1) = 2$, $w(D_2) = 2$, $w(D_3) = 1$, $D_1 \neq D_2, \gcd(u_1, u_2) = 1$,
$\mathsf{ExHarADD}^{1+2\to 2}$: $w(D_1) = 1$, $w(D_2) = 2$, $w(D_3) = 2$, $D_1 \neq D_2$, $\gcd(u_1, u_2) = 1$,
$\mathsf{ExHarDBL}^{1\to 2}$: $w(D_1) = 1$, $w(D_3) = 2$, $D_1 = D_2$, $\gcd(h, u_1) = 1$,
$\mathsf{ExHarDBL}^{2\to 1}$: $w(D_1) = 2$, $w(D_3) = 1$, $D_1 = D_2$, $\gcd(h, u_1) = 1$.

Similarly, there could exist other exceptional procedures using degenerate divisors, for instance, $\mathsf{ExHarADD}^{1+2\to 1}$ and $\mathsf{ExHarADD}^{1+1\to 2}$. However, these cases are not suitable for application to scalar multiplication, because the combination of divisors D_1, D_2, D_3 is not freely chosen. In this paper we do not consider these other exceptional procedures.

The computational cost of these exceptional procedures strongly depends how to implement the addition formulae. In the following we assume $g = 2$. Table 1 shows the cost of the Harley algorithm and its degenerate algorithms improved by Lange [Lan02a] and Sugizaki et al. [SMC+02]. The explicit algorithms for *HarleyDBL* and its degenerate variations are shown in the full version of this paper [KKA+04]. We evaluate the computational cost according to the time of one multiplication M and one inversion I. The exceptional cases are clearly faster than the ordinary cases.

Table 1. Number of multiplication and inversion of Harley Algorithm.

Addition Formula	Cost
HarleyADD	$1I + 25M$
HarleyDBL	$1I + 27M$
$\mathsf{ExHarADD}^{2+2\to 1}$	$1I + 14M$
$\mathsf{ExHarADD}^{1+2\to 2}$	$1I + 11M$
$\mathsf{ExHarDBL}^{2\to 1}$	$1I + 17M$
$\mathsf{ExHarDBL}^{1\to 2}$	$1I + 7M$

The total cost for computing the scalar multiplication with the double-and-add-always method is $318I + 8268M$ on average for a 160-bit HECC, if there is no exceptional procedure during the computation. For example, it is $10112.4M$ for $1I = 5.8M$.

4.1 Efficient Scalar Multiplication Using Degenerate Divisors

In this section we present efficient scalar multiplication using degenerate divisors.

The main thrust of our improvement is to choose a degenerate divisors as the base point. As we noted in the previous section, the computational cost of group operation between standard divisors and degenerate divisors are quite different. (See Table 1 and [KKA+04].) For example, $\mathsf{ExHarADD}$ is faster than *HarleyADD*. Note that the scalar multiplication is usually computed from the most significant bit because we can utilize efficient mixed coordinates for the affine base point D [CMO98]. Indeed, the binary method (Algorithm 1) in Section 2.2 is a left-to-right method, and thus the base point D is added to D_1 during the scalar multiplication dD. In our case, the base point D is chosen as the degenerate divisor (e.g., $w(D) < g$), so $\mathsf{ExHarADD}$ can be calculated more efficiently than the standard HECADD. Therefore we are able to achieve efficient scalar multiplication using the degenerate base point D.[1]

Here we have a question about the security of choosing the degenerate divisor as the base point. The following theorem can be easily proven thanks to random self reducibility. (See [KKA$^+$04] for the proof.)

Theorem 1. *Let J be the Jacobian of a hyperelliptic curve of genus g, where $\frac{\#J}{c}$ is prime. We assume that $\bar{D} = (u, v)$ is the degenerate divisor, where* $\deg u < g$. *Solving the discrete logarithm problem with base point $\bar{D}$ is as intractable as using a random divisor of J.*

From this theorem, the special base point has no influence the security of the underlying discrete logarithm problem.

The presented efficient scalar multiplication with the fixed base point can be applied only to ElGamal-type encryption, the sender of Diffie-Hellman, and DSA. The scalar of these schemes is usually an ephemeral random number, and thus we focus only on SPA, not DPA. A standard countermeasure against SPA is the double-and-add-always method (Algorithm 3) in Section 3. Note that SPA-resistant addition chains (e.g., window-based methods) are not used for the efficiency improvement using the degenerate base point, because it is difficult to generate two degenerate divisors D, aD with previously known integer a.

We assume that the scalar multiplication is computed using the double-and-add-always method (Algorithm 3). We choose the divisor with the weight 1 as the base point D, and then $\mathsf{ExHarADD}^{1+2\to 2}$ is used for HECADD. The power consumption curve of the scalar multiplication shows the fixed pattern, namely the repeats of the power consumption curve of *HarleyDBL* and $\mathsf{ExHarADD}^{1+2\to 2}$. However, HECADD $2D + D$ cannot be computed using $\mathsf{ExHarADD}^{1+2\to 2}$, so we must use different addition formula $\mathsf{ExHarADD}^{1+2\to 2}_{D+2D}$ because $D = P - P_\infty$ and $2D = P + P - 2P_\infty$ have a common factor.

We compare the computational cost of the scalar multiplication for a divisor with weight 1, with that for a divisor with weight 2. For the divisor with weight 1, the first HECDBL and HECADD costs $1I + 7M$ ($\mathsf{ExHarDBL}^{1\to 2}$) and $1I + 24M$ ($\mathsf{ExHarADD}^{1+2\to 2}_{D+2D}$) respectively. The other HECDBL and HECADD costs $1I + 27M$ (*HarleyDBL*) and $1I + 11M$ ($\mathsf{ExHarADD}^{1+2\to 2}$) respectively. Therefore the total cost of the scalar multiplication is $(1I + 7M) + (1I + 24M) + ((1I + 27M) + (1I + 11M)) \times 158 = 318I + 6035M$. On the other hand, for a divisor with weight 2, HECDBL and HECADD costs $1I + 27M$ (*HarleyDBL*) and $1I + 25M$ (*HarleyADD*), respectively. The total cost of the scalar multiplication is $((1I + 27M) + (1I + 25M)) \times 159 = 318I + 8268M$. Thus, the proposed scheme can achieve about 22% improvement under $1I = 5.8M$.

[1] D is randomly generated from a point P on C. The most efficient choice of P is $(0, y)$ [KKA$^+$04].

Table 2. Improved timing of scalar multiplication.

Base Point	Timing
random weight 2	13.36 ms
random weight 1	10.92 ms

In order to demonstrate the improvement of our algorithm, we implemented the proposed scheme (See Table 2). For our experiment we chose the following hyperelliptic curve with genus 2 from [HSS00]: Let $\mathbb{F}_{2^{83}}$ be defined as $\mathbb{F}_2[t]/(t^{83}+t^7+t^4+t^2+1)$ and $y^2+h(x)y=f(x) \quad \text{over} \quad \mathbb{F}_{2^{83}}$,

$$h(x) = x^2 + \texttt{2b770d0d26724d479105f}\ x + \texttt{540efb4e1010a0fc69f23},$$
$$f(x) = x^5 + \texttt{2cc2f2131681e8fe80246}\ x^3 + \texttt{53b00bad6fbb8f6ea5538}\ x + \texttt{54f5f3b4f4fc25898ee4}.$$

The order of the Jacobian is:

$$2 \times 4676805239458889338251790932012099166718374086785 3.$$

The experiment was implemented on an Intel Xeon Processor 2.80GHz using operation system Linux 2.4 (RedHat). We employed compiler gcc 3.3 and number theoretic library NTL5.3 with GMP4.0 [NTL]. The precise measurement of the timing difference of scalar multiplication on PC is difficult due to many other processes running on the PC, so that we use a CPU clock as the measurement of timing for the test codes. In this computational environment, the timing ratio of the inversion by the multiplication is estimated to be $I/M = 5.78$ from 10 million random samples.

5 Timing Attack on HECC

In this section, we propose a timing attack using degenerate divisors.

5.1 Target of Timing Attack

We explain the target system of the proposed timing attack.

Note that the probability, which a randomly chosen divisor in Jacobian $J_c(\mathbb{F}_{2^n})$ causes an exceptional procedure in the addition formula, is $\mathcal{O}(1/2^n)$ [Nag00]. Therefore, the exceptional procedure appears with negligible probability during the scalar multiplication for a randomly chosen base point. The attacker has to choose appropriate divisors in order to achieve the timing attack.

The proposed attack is categorized as a chosen ciphertext attack on a public-key cryptosystem. We assume that the secret key d is fixed during the attack and the base point P can be freely chosen by the attacker. This scenario has been used for several attacks, namely an exceptional procedure based attack [AT03,Ava03a,Gou03,IT03]. The protocols for which our proposed attack works are HEC ElGamal-type decryption (e.g. HECIES) and single-pass HEC Diffie-Hellman.

5.2 Recovering Secret Scalar

We describe here how to recover the secret key d by observing the whole timing of the scalar multiplication dD using the exceptional procedures, where D is a divisor of $J = J_c(\mathbb{F}_{2^n})$. The recovering technique follows the algorithm proposed by Goubin [Gou03] and Izu et al. [IT03]. However, we have to consider where the exceptional procedure occurs and how to compare this timing with that of the ordinary case.

Denote by $(d_{m-1}d_{m-2}\cdots d_1 d_0)_2$ the binary representation of d with $d_{m-1} = 1$. We assume the scalar multiplication is calculated by the double-and-add-always method (Algorithm 3). The attacker tries to bring about an exceptional procedure during the scalar multiplication using the degenerate divisor aD for the base point D and some integer a. We can easily choose the base point, e.g., divisor $D = ((a^{-1}) \bmod \frac{\#J}{c})\bar{D}$ for any degenerate divisor $\bar{D}$, where $\#J$ is the order of $J_c(\mathbb{F}_{2^n})$. We calculate the whole time of the scalar multiplication dD and compare it with that of the scalar multiplication with random base point.

We now describe how to determine the second bit d_{m-2}. First, we determine the second most significant bit d_{m-2}. If $d_{m-2} = 0$, the addition chain generates the following sequence $D, 2D, 3D$(dummy), $4D, 5D$(dummy)$, 8D$ for $d_{m-3} = 0$, and $D, 2D, 3D$(dummy)$, 4D, 5D$ for $d_{m-3} = 1$. If divisor $4D$ is degenerate, the exceptional procedures $\mathsf{ExHarDBL}^{2\to1}(2D) \to 4D$ and $\mathsf{ExHarADD}^{1+2\to2}$ $(4D) \to 5D$ appear. In this case we have additional exceptional procedure $\mathsf{ExHarDBL}^{1\to2}(4D) \to 8D$ only if $d_{m-3} = 0$.

$$\begin{array}{llllll} d_{m-2} = 0: & 2D & \xrightarrow{HarleyADD} & 3D & & \\ & & \xrightarrow{\mathsf{ExHarDBL}^{2\to1}} & 4D & \xrightarrow{\mathsf{ExHarADD}^{1+2\to2}} & 5D \\ & & & & \xrightarrow{\mathsf{ExHarDBL}^{1\to2}} & 8D\ (d_{m-3} = 0) \end{array}$$

Therefore the timing difference ΔT^0 for $d_{m-2} = 0$ is: follows:

$$\begin{aligned} \Delta T^0 = & (HarleyDBL - \mathsf{ExHarDBL}^{2\to1}) + (HarleyADD - \mathsf{ExHarADD}^{1+2\to2}) \\ & +1/2(HarleyDBL - \mathsf{ExHarDBL}^{1\to2}) = 34M. \end{aligned}$$

Similarly, $d_{m-2} = 1$, the addition chain generates the following sequence $D, 2D$, $3D$, $6D, 7D$ or $D, 2D, 3D, 6D, 7D$(dummy)$, 12D$. If divisor $6D$ is degenerate, we have the following exceptional procedures:

$$\begin{array}{llllll} d_{m-2} = 1: & 2D \xrightarrow{HarleyADD} 3D & \xrightarrow{\mathsf{ExHarDBL}^{2\to1}} & 6D & \xrightarrow{\mathsf{ExHarADD}^{1+2\to2}} & 7D \\ & & & & \xrightarrow{\mathsf{ExHarDBL}^{1\to2}} & 12D\ (d_{m-3} = 0) \end{array}$$

Therefore the timing difference ΔT^1 for $d_{m-2} = 1$ is as follows:

$$\begin{aligned} \Delta T^1 = & (HarleyDBL - \mathsf{ExHarDBL}^{2\to1}) + (HarleyADD - \mathsf{ExHarADD}^{1+2\to2}) \\ & +1/2(HarleyDBL - \mathsf{ExHarDBL}^{1\to2}) = 34M. \end{aligned}$$

The timing differences for $d_{m_{i-2}} = 0, 1$ are exactly same for this attack. For a 160-bit HECC, the timing difference is about 0.34% of the whole scalar multiplication under $1I = 5.8M$.

In the above situation, the attacker is able to compute the bit by comparing the whole computation time of the scalar multiplication for $((4^{-1}) \bmod \frac{\#J}{c})\bar{D}$ or $((6^{-1}) \bmod \frac{\#J}{c})\bar{D}$, where $\bar{D}$ is any divisor with weight 1.

The lower bits can be recursively recovered using the above method. We explain how to compute d_i after knowing the highest bits $(d_{m-1}d_{m-2}\cdots d_{i+1})$. The attacker chooses $D_0 = ((\sum_{j=i}^{m-1} d_j 2^{j-i})^{-1} \bmod \frac{\#J}{c})\bar{D}$ or $D_1 = ((\sum_{j=i+1}^{m-1} d_j 2^{j-i})^{-1} \bmod \frac{\#J}{c})\bar{D}$ as the base point, where $\bar{D}$ is any divisor with weight 1. The base point D_0, D_1 brings about the exceptional procedure if $d_i = 0, 1$, respectively.

We have measured the average timings of the scalar multiplication for the Harley algorithm (*Harley*), the Harley algorithm with one exceptional procedure (*Harley* + ExHarley), and the Harley algorithm with one exceptional procedure of the Cantor algorithm (*Harley* + ExCantor) [2]. Table 3 shows the results with 50000 random samples.

The arithmetic of HECC was programmed only using the operations of finite field $\mathbb{F}_{2^{83}}$. The common commands of NTL library were used for both the exceptional procedure and the ordinary procedure. The timing of the branch condition, which switches the ordinary case to the exceptional cases, is negligible compared to that of the operations of $\mathbb{F}_{2^{83}}$.

The timing difference using one exceptional procedure ExHarley or ExCantor is 0.15% or 1.72%, respectively. These timings are comparable to the results in Section 4 and [KKA+04]. The exceptional procedure of the Cantor algorithm causes a larger difference than that of the Harley algorithm.

Although there is a timing difference of exceptional cases from the ordinary case, the difference is quite small. In the next section we explain how to improve the success probability of determining the secret bit.

5.3 Outline of Experiment

We report an experiment of the timing attack based on the exceptional procedures in the previous section. We explain our experimental technique, which distinguishes the timing difference of the exceptional procedures from the ordinary procedure. The test codes calculate scalar multiplication dD for the base

Table 3. Timings of scalar multiplication.

Addition Formula	Timing
Harley	13.36 *ms*
Harley + ExHarley	13.34 *ms*
Harley + ExCantor	13.59 *ms*

[2] Some implementations employ the Cantor algorithm for exceptional cases because of many exceptional cases.

point D and n-bit secret scalar d. We assume that the scalar multiplication is computed by the double-and-add-always method (Algorithm 3).

This technique of measurement is similar to that used in [DKL+98]. Let T be the average time of N scalar multiplications with different divisors. Note that there are enough such divisors for the timing attack. The attacker aims at determining d_i, that is the i-th bit of the secret scalar d. The total bits of the secret key can be recovered by recursively applying this attack from the most significant bits. Section 5.2 shows how to generate a divisor that bring about the exceptional procedure with $d_i = 0$ or $d_i = 1$. We denote by D^{ex0} or D^{ex1} these divisors, respectively. We have the following three different timings:

T^{rand}: the average time with a randomly chosen divisor
T^{ex0}: the average time with divisor D^{ex0}
T^{ex1}: the average time with divisor D^{ex1}

The sample number of different divisors for obtaining T^{ex0} or T^{ex1} is N for each bit d_i. Timing T^{rand} is measured with N different divisors during the whole attack. The minimum number N for succeeding in the attack depends on the computational environment and distribution of divisors, and we show the minimum N for our setting in the next section.

Then we compute the differences from the random instance, more precisely $\Delta T^0 = |T^{rand} - T^{ex0}|$ and $\Delta T^1 = |T^{rand} - T^{ex1}|$. If $d_i = b$ holds for $b = 0, 1$, then $\Delta T^{\bar{b}}$ is nearly zero due to random distribution $T^{ex\bar{b}}$, that is $T^{ex\bar{b}} \approx T^{rand}$, where $\bar{b} = 1 - b$. Recall that the scalar multiplication bring about an exceptional procedure with negligible probability for a randomly chosen base point. Therefore, we can suppose $d_i = b$ if $\Delta T^b > \Delta T^{\bar{b}}$ holds for $b = 0, 1$. We summarize this as follows:

Algorithm 4 *Experiment for Determining d_i*

```
1. Calculate $T^{rand}, T^{ex0}, T^{ex1}$
2. Calculate $\Delta T^0 = |T^{rand} - T^{ex0}|$ and $\Delta T^1 = |T^{rand} - T^{ex1}|$
3. Return $d_i = b$ if $\Delta T^b > \Delta T^{\bar{b}}$ for $b = 0, 1$
```

5.4 Analysis of Timing Attack

We analyze the distribution of timings ΔT^b for $b = 0, 1$, and we present the experimental result of the timing attack.

The distribution of timings ΔT^b depends on the distribution of divisors D appearing in the scalar multiplication dD and the noise arising from the measurement of dD. We can average the deviation by increasing the number of base points $D_1, ..., D_N$ used for the experiment. Therefore, we can define the following distribution, which comprises k iterations of experiments for T^{rand} and T^{exi}.

Definition 2. *Let $\mathcal{T}^{rand}(N)$ and $\mathcal{T}^{exb}(N)$ be the average timing of the scalar multiplication dD_j for $j = 1, 2, ..., N$ with random base point D_j and base point D_j that brings about the exceptional procedure at target bit d_b, respectively. The (N, k)-distribution $\mathcal{T}^b_{N,k}$ is the distribution of timings $\Delta T^b = \mathcal{T}^{rand}(N) - \mathcal{T}^{exb}(N)$ for k iterations of the experiment for obtaining $\mathcal{T}^{rand}(N)$ and $\mathcal{T}^{exb}(N)$, where $b = 0, 1$.*

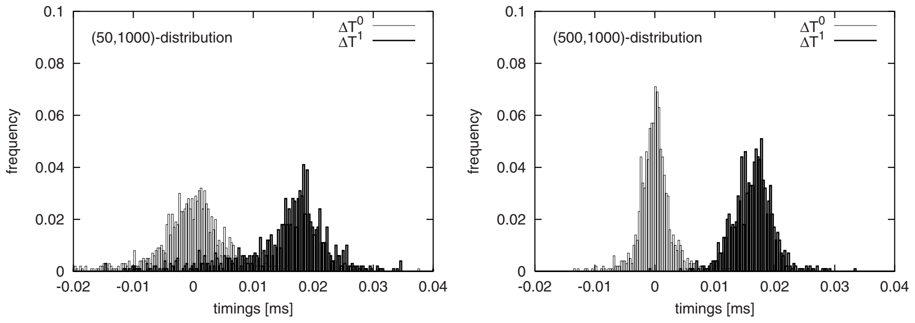


Fig. 1. (50,1000)-distribution, and (500,1000)-distribution.

The mean value of (N, k)-distribution $\Delta\mathcal{T}^b$ can be used for determining the secret bit b. We can determine it by comparing the mean value of $\Delta\mathcal{T}^b$ with that of $\Delta\mathcal{T}^{\bar{b}}$ for $b = 0, 1$ as we showed in Algorithm 4.

Figure 1 shows histograms of (N, k)-distribution $\mathcal{T}^b_{N,k}$ for different $N = 50$, $N = 500$ with fixed $k = 1000$. The horizontal axis and vertical axis are timing (ms) and frequency of timing, respectively. To compare the different types of (N, k)-distributions, we averaged the timings over N divisors and normalized the frequency by k. When N increases from 50 to 500, the overlapping of the two histograms between $\mathcal{T}^0_{N,k}$ and $\mathcal{T}^1_{N,k}$ becomes smaller. The proportion of the overlapping shows the probability of determining the secret bit. The success rate of our experiment is defined as the ratio of correct determines to the total number of experiments (i.e. k). Figure 2 shows the relation between the success rate for the increasing number N of random divisors with fixed $k = 1000$. If we choose $N > 500$, we achieve almost a 100% success rate.

If k is small, the effect of the noise on the experiment becomes large and the success rate becomes smaller. For example, we show the distribution of $k = 50$ in Fig. 2, which is irregular even for increasing N. However, the choice of $k = 1000$ is large enough for eliminating the influence of the noise on our experiment, and N

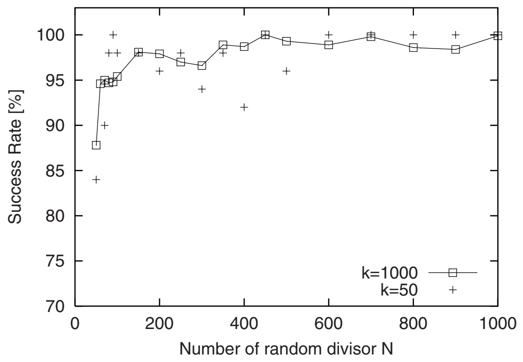


Fig. 2. Success rate of determining one bit.

is considered to be the number of measurements required for the timing attack. Consequently, we conclude one bit of the secret scalar can be recovered if the attacker can measure more than 500 samples ($N > 500$) with high probability.

In order to reveal all 160 bits of the secret scalar, we recursively performed the proposed attack from the 2nd most significant bit to the 2nd least significant bit. The least significant bit could be easily revealed. In this case, we required $500 \times 158 = 79,000$ samples. Our experiment did not provide error correction similar to that used by [DKL+98]. With error-correction implemented, the time required for recovery would decrease dramatically because fewer samples (for example, $N = 50$) would be needed for successful recovery.

6 Summary

We investigated the use of degenerate divisors of hyperelliptic curves in cryptography. The timing of computing addition formulas with degenerate divisors (the exceptional procedure) is in general different from that of the standard procedure. We considered the precise timing of the exceptional procedure required using the Harley algorithm and Cantor algorithm.

We presented two different applications of the exceptional procedures, – which can be, however, a two-edged sword. For a positive application we presented an efficient scalar multiplication using degenerate divisors as the base point. The discrete logarithm problem of the degenerate divisors is as hard as that of the random divisors due to the random self-reducibility. Our experiment shows that we can achieve about 20% improvement in speed. For a negative application, we mounted the degenerate divisors to the timing attack on the secret scalar. The attack tries to distinguish the timing of the exceptional procedure from that of the ordinary procedure. About 500 samples of the scalar multiplication enable us to break one bit of the secret key.

References

[AT03] T. Akishita and T. Takagi, "Zero-Value Point Attacks on Elliptic Curve Cryptosystem," ISC 2002, LNCS 2851, Springer-Verlag, pp.218-233, 2003.

[Ava03a] R. Avanzi, "Countermeasures against Differential Power Analysis for Hyperelliptic Curve Cryptosystems," CHES 2003, LNCS 2779, Springer-Verlag, pp.366-381, 2003.

[Ava03b] R. Avanzi, "Aspects of Hyperelliptic Curves over Large Prime Fields in Software Implementations", Cryptology ePrint Archive, 2003/253, IACR, 2003.

[Can87] D. Cantor, "Computing in the Jacobian of a Hyperelliptic Curve," Mathematics of Computation, 48, 177, pp.95-101, 1987.

[CMO98] H. Cohen, A. Miyaji, and T. Ono, "Efficient Elliptic Curve Exponentiation Using Mixed Coordinates," *ASIACRYPT '98*, LNCS1514, pp.51-65, 1998.

[Cor99] J.-S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," CHES '99, LNCS 1717, Springer-Verlag, pp.292-302, 1999.

[DKL+98] J.F. Dhem, F. Koeune, P.A. Leroux, P. Mestré, J.J Quisquater and J.L. Willems, "A Practical Implementation of the Timing Attack," UCL Crypto Group Technical Report CG–1998/1, 1998.

[GMP] GMP, GNU MP Library GMP. http://www.swox.com/gmp

[Gou03] L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystem," PKC2003, LNCS 2567, Springer-Verlag, pp.199-211, 2003.

[Har00a] R. Harley, "Adding.text," 2000. http://cristal.inria.fr/~harley/hyper/

[Har00b] R. Harley, "Doubling.c," 2000. http://cristal.inria.fr/~harley/hyper/

[HSS00] F. Hess, G. Seroussi and N. Smart, "Two Topics in Hyperelliptic Cryptography," CSTR-00-008, Depart. of Computer Science, University of Bristol, 2000.

[IT03] T. Izu and T. Takagi, "Exceptional Procedure Attack on Elliptic Curve Cryptosystems," PKC2003, LNCS 2567, Springer-Verlag, pp.224-239, 2003.

[JT01] M. Joye and C. Tymen, "Protection against Differential Analysis for Elliptic Curve Cryptography," CHES 2001, LNCS 2162, Springer-Verlag, pp.377-390, 2001.

[KKA$^+$04] M. Katagi, I. Kitamura, T. Akishita, and T. Takagi, "Novel Efficient Implementations of Hyperelliptic Curve Cryptosystems using Degenerate Divisors," Cryptology ePrint Archive, IACR, 2004. http://eprint.iacr.org/

[Kob89] N. Koblitz, "Hyperelliptic Cryptosystems," Journal of Cryptology, Vol.1, Springer-Verlag, pp.139-150, 1989.

[Koc96] C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," CRYPTO '96, LNCS 1109, pp.104-113, 1996.

[KJJ99] C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," CRYPTO '99, LNCS 1666, pp.388-397, 1999.

[KGM$^+$02] J. Kuroki, M. Gonda, K. Matsuo, J. Chao and S. Tsujii, "Fast Genus Three Hyperelliptic Curve Cryptosystems," Proc. of SCIS2002, 2002.

[Lan02a] T. Lange, "Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae," Cryptology ePrint Archive, 2002/121, IACR, 2002.

[Lan02b] T. Lange, "Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves," Cryptology ePrint Archive, 2002/147, IACR, 2002.

[Lan02c] T. Lange, "Weighed Coordinate on Genus 2 Hyperellipitc Curve," Cryptology ePrint Archive, 2002/153, IACR, 2002.

[Mum84] D. Mumford, *Tata Lectures on Theta II*, Progress in Mathematics 43, Birkhäuser, 1984.

[MCT01] K. Matsuo, J. Chao and S. Tsuji, "Fast Genus Two Hyperelliptic Curve Cryptosystems," Technical Report ISEC2001-31, IEICE Japan, pp.89-96, 2001.

[Nag00] N. Nagao, "Improving Group Law Algorithms for Jacobians of Hyperelliptic Curves," ANTS-IV, LNCS 1838, Springer-Verlag, pp.439-448, 2000.

[NTL] NTL: A Library for Doing Number Theory. http://www.shoup.net/ntl

[Pel02] J. Pelzl, "Hyperelliptic Cryptosystems on Embedded Microprocessors," Diploma Thesis, Rühr-Universität Bochum, 2002.

[PWG$^+$03] J. Pelzl, T. Wollinger, J. Guajardo and C. Paar, "Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves," CHES 2003, LNCS 2779, Springer-Verlag, pp.351-365, 2003.

[SMC$^+$02] T. Sugizaki, K. Matsuo, J. Chao, and S. Tsujii, "An Extension of Harley Addition Algorithm for Hyperelliptic Curves over Finite Fields of Characteristic Two," Technical Report ISEC2002-9, IEICE Japan, pp.49-56, 2002.

[Sch00] W. Schindler, "A Timing Attack against RSA with the Chinese Remainder Theorem," CHES 2000, LNCS 1965, pp.109-124, 2000.

[Sch02] W. Schindler, "A Combined Timing and Power Attack," PKC 2002, LNCS 2274, pp.263-279, 2002.

[SKQ01] W. Schindler, F. Koeune, J.-J. Quisquater, "Improving Divide and Conquer Attacks against Cryptosystems by Better Error Detection/Correction Strategies," Cryptography and Coding, 8th IMA Int. Conf., LNCS 2260, pp.245-267, 2001.

Hyperelliptic Curve Coprocessors on a FPGA

HoWon Kim[1], Thomas Wollinger[1], YongJe Choi[2],
KyoIl Chung[2], and Christof Paar[1]

[1] Department of Electrical Engineering and Information Sciences,
Ruhr-Universität Bochum, Germany
{khw,wollinger,cpaar}@crypto.rub.de
[2] Electronics and Telecommunications Research Institute, South Korea
{choiyj,kyoil}@etri.re.kr

Abstract. Cryptographic algorithms are used in a large variety of different applications to ensure security services. It is, thus, very interesting to investigate various implementation platforms. Hyperelliptic curve schemes are cryptographic primitives to which a lot of attention was recently given due to the short operand size compared to other algorithms. They are specifically interesting for special-purpose hardware. This paper provides a comprehensive investigation of high-efficient HEC architectures.
We propose a genus-2 hyperelliptic curve cryptographic coprocessor using affine coordinates. We implemented a special class of hyperelliptic curves, namely using the parameter $h(x) = x$ and $f = x^5 + f_1x + f_0$ and the base field GF(2^{89}). In addition, we only consider the most frequent case in our implementation and assume that the other cases are handled, e.g. by the protocol.
We provide three different implementations ranging from high speed to moderate area. Hence, we provide a solution for a variety of applications. Our high performance HECC coprocessor is 78.5% faster than the best previous implementation and our low area implementation utilizes only 22.7% of the area that the smallest published design uses. Taking into account both area and latency, our coprocessor is an order of magnitude more efficient than previous implementations. We hope that the work at hand provides a step towards introducing HEC systems in practical applications.

Keywords: Hyperelliptic curve coprocessor, high speed, low area, reconfigurable hardware, FPGA, cryptographical applications, affine coordinates.

1 Introduction

Cryptographic primitives are used in a variety of different applications. It is obvious that the condition for a practical implementation is very much application driven. Imagine a scenario, where a number of PDAs communicate with a server. On the server side high data throughput is a necessity because the server has to encrypt the data traffic of all personal devices. Area and power consumption is often not critical on the server. However, moderate speed is acceptable on the PDAs. The battery life and size of the device demand small area and low power. In short, both systems have totally different requirements in the implementation of the cryptographic primitive.

In the mid 1980s, a variant of the Diffie-Hellman key exchange was published based on the difficulty of the DL problem in the group of points of an elliptic curve (EC) over a finite field [12,18]. In 1988, Koblitz suggested for the first time the generalization of EC to curves of higher genus for cryptographic use, namely hyperelliptic curves [13]. The operand size of a hyperelliptic curve cryptosystem (HECC) is even shorter compared to elliptic curve cryptosystem (ECC). This fact is advantageous for HECC on any platform.

In recent years, there has been a major effort in improving the group operations and in implementing HECC on different processors (see [26] for a summary). Through the contribution from different industry and research groups, HECC was able to perform in the same range as ECC and even in some special cases outperform ECC. However, there are only a few implementations of this cryptographic primitive on hardware and all of them only target high encryption speed.

Our Main Contributions

Previously some contributions also implemented HECC on FPGAs [2,4,5,7,25]. However, we present for the first time an FPGA implementation considering affine explicit HECC formulae.

Given the fact that implementations of certain cryptosystems always are application dependent, we provide three different designs of the HECC coprocessor ranging from high performance to moderate area. We introduce in this work (i) the fastest HECC coprocessor up to date and (ii) the HECC implementation on FPGA with the lowest area.

The HECC coprocessor designs were evaluated by considering both the hardware requirements and the time constraints of the cryptographic application. Our HECC coprocessor is 78.5% faster than the best previous implementation and our low area implementation utilizes only 22.7% of the area of the smallest design published. Taking into account area and speed by using the area-time product our coprocessor is between a factor of 12 and 125 better than previous publications. Hence, we are able to provide designs that achieve the encryption using HECC faster utilizing less area. Through this improvement HECC is now approaching the performance range of ECC FPGA implementations.

The remainder of the paper is organized as follows. Section 2 summarizes the previous work of FPGA implementation of HECC. Section 3 gives a brief overview of the mathematical background. Section 4 presents the HECC coprocessor and in Section 5 we outline our methodology and the different design options. Our results and the analysis of them is given in Section 6. Finally, we end this contribution with the some conclusions.

2 Previous Work

This section gives a short overview of the hardware implementations targeting HECC. The first work discussing hardware architectures for the implementation of HECC appeared in [25]. The authors describe efficient architectures to implement the necessary field operations and polynomial arithmetic in hardware. All of the presented architectures are speed and area optimized. In [25], they also estimated that for a hypothetical clock frequency of 20 MHz, the scalar multiplication of HECC would take $21.4ms$ using the NAF method.

In [2] the authors presented the first complete hardware implementation of a hyperelliptic curve coprocessor. This implementation targets a genus-2 HEC over $\mathbb{F}_{2^{113}}$. The target platform is a Xilinx Virtex II FPGA. Point addition and point doubling with a clock frequency of 45MHz took 105μs and 90μs, respectively. The scalar multiplication could be computed in $20.2ms$.

In [4, 5] the authors presented extended results of [2]. They implemented a HECC coprocessor using a variety of base fields, ranging from $\mathbb{F}_{2^{83}}$ to $\mathbb{F}_{2^{163}}$, as well as two different multipliers (digit size D=1 and D=4 bits). The scalar multiplication took between $9ms$ and $40ms$ and used between 22,000 and 119,000 slices. From the implementation numbers given in the paper we consider that the design options using D=4 multiplier have unreasonable hardware requirements.

Note that publications mentioned so far adopt the Cantor algorithm to compute group operations [3]. Today, there exist more efficient algorithms to compute group addition and group doubling.

The first approach to implement a hyperelliptic curve cryptosystem in hardware using explicit formulae is presented in [7]. The authors used the inversion-free group operations for HECC introduced in [15]. Using the $\mathbb{F}_{2^{113}}$ and the NAF method for the scalar multiplication, they were able to reach a speed of $2.03ms$. Note that the paper at hand uses, in contrary to [7], the affine version of the explicit formulae. We are not aware of any hardware implementation using this kind of formulae.

3 Mathematical Background

In this section we introduce the theory of HECC, restricting attention to the relevant material and refer the interested reader to [14].

3.1 Definition of HECC

Let $\mathbb{F}$ be a finite field and let $\overline{\mathbb{F}}$ be the algebraic closure of $\mathbb{F}$. A hyperelliptic curve C of genus $g \geq 1$ over $\mathbb{F}$ is the set of the solutions $(x, y) \in \mathbb{F} \times \mathbb{F}$ to the following equation:

$$C : y^2 + h(x)y = f(x)$$

The polynomial $h(x) \in \mathbb{F}[x]$ is of degree at most g and $f(x) \in \mathbb{F}[x]$ is a monic polynomial of degree $2g+1$. For odd characteristic it suffices to let $h(x) = 0$ and to have $f(x)$ square free. Such a curve C is said to be non-singular if there does not exist any pair $(x, y) \in \overline{\mathbb{F}} \times \overline{\mathbb{F}}$ satisfying the equation of the curve C and the two partial differential equations $2y + h(x) = 0$ and $h'(x)y - f'(x) = 0$.

A divisor D is defined as $D = \sum m_i P_i$, where the P_i are points on the curve C and the integers m_i are the weights. Additionally, they have to fulfill the condition that $D^\sigma = \sum m_i P_i^\sigma$ is equal to D for all the automorphisms σ of $\overline{\mathbb{F}}$ over $\mathbb{F}$ (see [17] for details).

Divisors admit a reduced form. A reduced divisor can be represented as a pair of polynomials $u(x)$, $v(x)$ [20]. Reduced divisors can be added (group addition), e.g. $D_3 = D_1 + D_2$, or doubled (group doubling), e.g. $D_2 = 2D_1 = D_1 + D_1$, and hence the scalar multiplication $kD = D + \cdots + D$ can be performed. The scalar multiplication kD is the basic operation of HECC, that we want to implement with a coprocessor.

3.2 Group Operations

The formulae given for the group operations of HEC by Cantor [3] can be rewritten in explicit form, thus resulting in more efficient arithmetic. The explicit formulae were first presented in [9]. Starting with this finding, a considerable effort by different research groups has been put into finding more efficient operations. A treatment about the historic improvements of the HECC group operations can be found in [26].

In this work, we target our HECC coprocessor for genus-2 curves using underlying fields of characteristic two. We used the current fastest explicit formulae, as presented in [22], where the authors introduced a group doubling requiring a single field inversion, 9 field multiplications and 6 field squarings. Group addition can be computed with 1 field inversion, 21 field multiplications and 3 field squarings [16]. These explicit formulae as given in the references cover only the most frequent case that happens in almost all cases. In addition, special cases of curve parameters are used to decrease the complexity of the group operations.

3.3 Security of HECC

It is widely accepted that for most cryptographic applications based on EC or low genus HEC, the necessary group is of order at least $\approx 2^{160}$. Thus, for HECC over $\mathbb{F}_q$, we must have at least $g \cdot \log_2 q \approx 160$. In particular, we will need a field order $|\mathbb{F}_q| \approx 2^{80}$ for genus-2 curves. Even the very recent attack found by Thériault [24] shows no progress in attacks against genus-2 HEC.

In addition, one should consider the Weil decent attack methodology [8] and especially the GHS Weil decent attack [10]. Consider E to be a non-supersingular elliptic curve defined over a field $K = \mathbb{F}_{2^m}$, and m is composite. The idea of the attack is to reduce the ECDLP in $E(\mathbb{F}_{2^m})$ to the DLP in the jacobian variety of a curve of larger genus defined over a proper subfield $k = \mathbb{F}_l$ of K. The curve parameters selected ($h(x) = x$ and $f = x^5 + f_1 x + f_0$) and the chosen underlying field GF(2^{89}) have to our knowledge no security implications. In addition, the approximate group order of 2^{178} are well suited for medium term security.

4 HECC Coprocessor

The coprocessor has to compute the scalar multiplication kD for the HECC. Efficient algorithms are needed to implement the group operations as well as the field arithmetic. This work proposes three different processor architectures that are also well suited for implementation on Field Programmable Gate Arrays (FPGAs). The implementations target a special class of hyperelliptic curves ($h(x) = x$, $f = x^5 + f_1x + f_0$) and a fixed underlying field GF(2^{89}). We implemented only the most frequent case and assume for a practical application, that the protocol will handle the other case, e.g. by restarting the setup phase of the protocol.

The HEC coprocessor consists of three main components: main control unit (MC), arithmetic unit (AU), and register file or memory (RF), see Figure 1. MC is the main control unit of the coprocessor. It computes the scalar multiplication kD for HECC and interacts with the host system as well as with the other components. It generates control signals for RF, interconnection block, and AU. The interconnection block, which is mainly composed of multiplexers, takes care of the data exchange between the RF and the AU. The AU performs the field and group operations. In the following, we are going to describe the individual components in more detail.

4.1 Field Operation Units

The AU in the HECC coprocessor has field operation units for addition, squaring, multiplication and inversion over $\mathbb{F}_{2^{89}}$. In existing literature one can find a variety of ways to implement the field operations. We provide a short description of the implementation and give references pointing to more detailed information.

Field Adder: The addition of two elements requires the modulo 2 operation of the coefficients of the field elements. Hence, we use 89 exclusive OR gates to add two field elements in one clock cycle.

Field Squarer: The squaring of a field element $A = \sum_{i=0}^{m-1} a_i x^i$ is ruled by the following equation: $A^2 \equiv \sum_{i=0}^{m-1} a_i x^{2i} \bmod f(x)$. It is implemented with

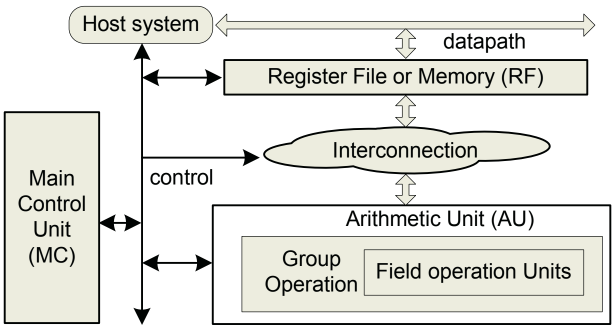


Fig. 1. Architecture of the HECC coprocessor.

a combinatorial logic and specialized with an underlying minimal polynomial $f(x)$. Squaring can be performed within one clock cycle and further details can be found in [21].

Field Inversion: The inversion was implemented using the Modified Almost Inverse Algorithm (MAIA) [11] and rewritten in Algorithm 1. In addition, we increased the performance by using the loop unfolding techniques described in [27]. Detailed algorithm is shown in Algorithm 2. These techniques allow to execute the Step 2 to 5 of Algorithm 1 in nearly one clock cycle. In order to do so, we a) unrolled the loop (Step 2) and b) merged Step 5 in Step 2. To realize the improvement a) we had to replicate and expand the conditional statement for all cases. The replication number of the loop body was decided by considering the tradeoff between the hardware complexity and performance. In our design, we have unrolled four loops in MAIA algorithm and we get about two times performance gain in inversion operations with sacrificing the hardware complexity.

The field inversion can be performed in an average time of $1.3\mu s$ (see Table 1). The input data that was used to measure the execution time had an Hamming weight of $n/2$, where n is the bit size of the input data. The reason being the performance of the inversion varies with the input data.

Algorithm 1. Modified Almost Inverse Algorithm for inversion in $\mathbb{F}_{2^m}$

```
INPUT : $a \in \mathbb{F}_{2^m}, a \neq 0$.
OUTPUT : $a^{-1} \bmod f(x)$
  1. $b \leftarrow 1$, $c \leftarrow 0$, $u \leftarrow a$, $v \leftarrow f$.
  2. While $x$ divides $u$ do:
      2.1 $u \leftarrow u/x$
      2.2 If $x$ divides $b$ then $b \leftarrow b/x$; $else\, b \leftarrow (b+f)/x$.
  3. If $u = 1$ then return($b$).
  4. If $\deg\_u < \deg\_v$ then: $u \leftrightarrow v, b \leftrightarrow c$.
  5. $u \leftarrow u + v, b \leftarrow b + c$.
  6. Goto step 2.
```

Field Multiplication: For the field multiplication we use the digit multiplier introduced in [23]. This kind of multiplier allows a trade-off between speed, area and power consumption. This can be achieved by a varying number of multiplicand coefficients that are processed in parallel, denoted as digit-size D. Given D, we denote by $d = \lceil m/D \rceil$ the total number of digits in a polynomial of degree $m-1$. Hence, $C \equiv AB \equiv A \sum_{i=0}^{d-1} B_i x^{Di} \bmod f(x)$.

We investigated the performance of the field multiplication in more depth because it is a crucial field operation in the HECC. The digit multipliers come in two flavors: Least Significant Digit (LSD) first and Most Significant Digit (MSD) first multiplier. We have implemented LSD first digit serial multiplier for this HECC design because it is known that LSD first multiplier has better performance than MSD first digit serial multiplier [23]. Table 1 shows the latency and the area requirement of the digit serial multiplier with various digit sizes.

Algorithm 2. Modified Almost Inverse Algorithm with Loop Unfolding

```
INPUT : $a \in \mathbb{F}_{2^m}, a \neq 0$.
OUTPUT : $a^{-1} \bmod f(x)$
  1. $b \leftarrow 1,\ c \leftarrow 0,\ u \leftarrow a,\ v \leftarrow f, \deg\_u \leftarrow 0, \deg\_v \leftarrow m$
  2. Find the degree of $u$ and save it to $\deg\_u$
  3. While $\deg\_u \neq 0$ do:
     3.1 If $u[0] = 0$ then let $b' = (b + b[0] * f)/x, b'' = (b' + b'[0] * f)/x, b''' = (b'' + b''[0] * f)/x)$
       3.1.1 If $u[3:0] = (0000)_2$ then $u \leftarrow u/x^4, \deg\_u \leftarrow \deg\_u - 4$,
             $b \leftarrow ((((b + b[0] * f)/x + b'[0] * f)/x + b''[0] * f)/x + b'''[0] * f)/x$
       3.1.2 If $u[2:0] = (000)_2$ then $u \leftarrow u/x^3, \deg\_u \leftarrow \deg\_u - 3$,
             $b \leftarrow (((b + b[0] * f)/x + b'[0] * f)/x + b''[0] * f)/x$
       3.1.3 If $u[1:0] = (00)_2$ then $u \leftarrow u/x^2, \deg\_u \leftarrow \deg\_u - 2,\ b \leftarrow ((b + b[0] * f)/x + b'[0] * f)/x$
       3.1.4 If $u[0] = (0)_2$ then $u \leftarrow u/x, \deg\_u \leftarrow \deg\_u - 1,\ b \leftarrow (b + b[0] * f)/x$
     3.2 else $(u[0] \neq 0$ : let $uv = u + v, bc = b + c)$
             Let $bc' = (bc + bc[0] * f)/x, bc'' = (bc' + bc'[0] * f)/x, bc''' = (bc'' + bc''[0] * f)/x$
       3.2.1 If $uv[3:0] = (0000)_2$ then $u \leftarrow uv/x^4$,
             $b \leftarrow ((((bc + bc[0] * f)/x + bc'[0] * f)/x + bc''[0] * f)/x$
             If $\deg\_u < \deg\_v$ then $v \leftarrow u$, c $\leftarrow b, \deg\_u \leftarrow \deg\_v - 4, \deg\_v \leftarrow \deg\_u$else $\deg\_u \leftarrow \deg\_u - 4$
       3.2.2 else If $uv[2:0] = (000)_2$ then $u \leftarrow uv/x^3$,
             $b \leftarrow (((bc + bc[0] * f)/x + bc'[0] * f)/x + bc''[0] * f)/x$
             If $\deg\_u < \deg\_v$ then $v \leftarrow u$, c $\leftarrow b, \deg\_u \leftarrow \deg\_v - 3, \deg\_v \leftarrow \deg\_u$ else $\deg\_u \leftarrow \deg\_u - 3$
       3.2.3 else If $uv[1:0] = (00)_2$ then $u \leftarrow uv/x^2,\ b \leftarrow ((bc + bc[0] * f)/x + bc'[0] * f)/x$
             If $\deg\_u < \deg\_v$ then $v \leftarrow u$, c $\leftarrow b, \deg\_u \leftarrow \deg\_v - 2, \deg\_v \leftarrow \deg\_u$ else $\deg\_u \leftarrow \deg\_u - 2$
       3.2.4 else If $uv[0] = (0)_2$ then $u \leftarrow uv/x,\ b \leftarrow (bc + bc[0] * f)/x$
             If $\deg\_u < \deg\_v$ then $v \leftarrow u$, c $\leftarrow b, \deg\_u \leftarrow \deg\_v - 1, \deg\_v \leftarrow \deg\_u$ else $\deg\_u \leftarrow \deg\_u - 1$
  4. return($b$).
```

Table 1. Performance of multiplication and inversion logic (FPGA Xilinx Virtex II XC2V4000 ff1517-6, $\mathbb{F}_{2^{89}}$).

Type	Digit Size [bits]	slices	Frequency [MHz]	Clock cycles	Time [μs]
Digit Serial Multiplier (LSD)	D=1	145	97.5	89	0.913
	D=4	239	106.8	23	0.215
	D=8	414	110.1	12	0.109
	D=16	645	87.4	6	0.069
	D=32	1,189	71.6	3	0.042
	D=45	1,616	63.7	2	0.031
	D=89	3,205	52.2	1	0.019
MAIA (4 loops unfolded)	-	733	74.6	97	1.300

One would expect the frequency of the multipliers to decrease with higher digit size. However, examining Table 1 one notices the lower frequency for D=1 and D=4 multipliers compared to D=8. Careful analysis shows that the critical path of the D=1 and 4 multipliers are dominated by the control logic, more specifically from the comparator logic and additional logic (such as multiplexers). For all the other multipliers the critical path is conditioned by the data path. The data path includes the parallel multiplication, the accumulation of the partial results and the reduction (for more detail see [23]).

The choice of the ideal multiplier for our HECC coprocessor largely depends on frequency of the total design and secondly, on the latency of the multiplier. The frequency of the coprocessor is limited by the interconnect network and does not exceed 63 MHz (see Table 3).

Hence, the multipliers using digit size D=32 are good choice. The reason being it can be clocked with a high enough frequency and computes the multiplication in only three cycles. Furthermore, this performance is achieved by moderate area usage.

4.2 Arithmetic Unit

Figure 2 shows the arithmetic unit (AU) that performs the group addition and group doubling operations. The AU consists of the field operation units and the group operation logic. The control logic is in charge of scheduling the right sequence of operations to perform the HEC group operations. Necessarily, the control unit has to interact with the main control unit (MC) and the register files (RF). In addition, we had to provide internal control logic for the multiplier and inversion logic.

Field addition and field squaring are the least expensive operations and, therefore, we decided to integrate them into the data path between RF and the multipliers. However, we still can compute the operation without consecutive multiplication by setting the right multiplexer options.

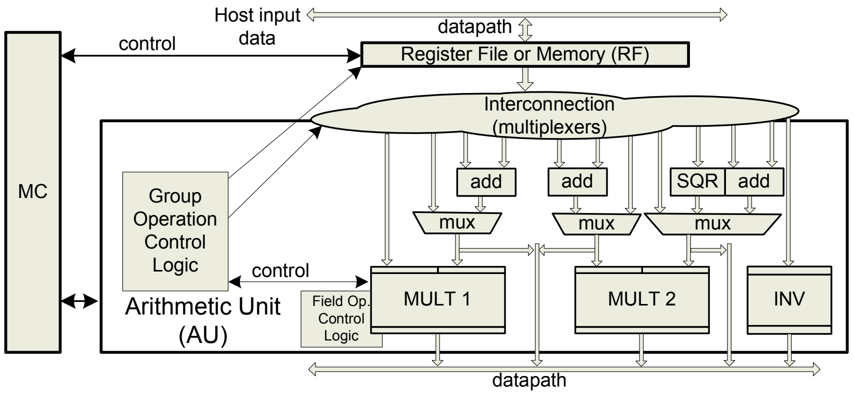


Fig. 2. Arithmetic Unit of the HECC coprocessor.

4.3 Interconnect Network

The interconnect network, consisting of a multiplexer, handles the data transfer between the RF and AU. Note that for our high performance design (see Section 5), we included the capability to load the input data in parallel. This results in the possibility to start two multipliers and the inversion logic at the same time.

5 Design Methodology for the HECC Coprocessor

In this section, we will briefly describe our design methodology that resulted in the HECC coprocessor. We tried for all presented design options to reach the best possible performance and at the same time to reduce overall hardware complexity. In order to do so we examined the parallelism within the group operations, minimized the number of registers, and reduced the complexity of the interconnect network.

Parallel Architecture for the Group Operations: We used the Data Dependence Graph (DDG) to design the architecture for computing the group operations. We found that the multiplication and the inversion operations are the dominant components and, therefore, crucial for the overall performance. Hence, we choose that the nodes of the DDG represent the multiplications and the inversions, whereas the addition and squaring are an edge of the DDG.

We can describe the DDG as $G(F) = (V, E)$, where V is the set of multiplication and inversion operations, F is the HECC explicit formulae, and E is the data flow between multiplication, inversion, and registers.

Our analysis using the DDG resulted in the choice of using two parallel multipliers for the genus-2 HECC coprocessor. This choice considered the hardware complexity as well as the performance. The utilization rate of the two multipliers for group addition and group doubling is 91% and 50%, respectively. Therefore, adding additional multipliers would not provide any advantages considering the increasing hardware complexity. Hence, we used two multipliers and speed up the performance of the HECC coprocessor by choosing a high digit size (D=32).

Previously, there has been two contributions studying the parallelism of the genus-2 HECC group operations [1, 19]. In [19], the authors develop a general methodology for obtaining parallel algorithm for the HECC. However, this work only focused on theoretical aspects on the parallelism of the HECC. Furthermore, they did not consider the register allocation and interconnect network complexity problems, which are important factors in practical implementations. In [1], the authors presented an architecture that is suited for real implementations by simulating the HECC coprocessor. However, this contribution uses a different design methodology, resulting in slightly different results. One main difference between our and their design methodology is that they did not include the addition and squaring into the data path.

Minimizing the Number of Registers: We have designed the HECC coprocessor with an effort to minimize the number of registers. We used eight 89-bit registers to store two divisor values which are updated during the computation of the group operations with the output result. In addition, we needed some extra registers to store intermediate values. We found the optimum number of extra registers by efficiently reusing these resources. This was manually done with the help of register allocation tables.

Reduction of the Complexity of the Interconnect Network: After we found the minimum number of registers, we tried to reduce the complexity of the interconnect network. We did this by looking at the inputs and outputs of the different units (field operations and RF) involved in the computation. The minimum complexity would be achieved by a) always storing a designated output into the same register and b) loading a certain input value every time from the same register. Looking at the complexity of the HECC group operations, it is obvious that there will not be a fixed register for loading inputs or storing the output of one unit. However, by trying to use this method we were able to reduce the size of the interconnection multiplexers.

5.1 Various Design Options for the HECC Coprocessor

We will now introduce the three types for HECC coprocessors implemented in this contribution. We have designed Type 1 for high performance and Type 2 and 3 for lower hardware complexity with reasonable performance. The characteristics of the various types of HECC coprocessors are summarized in Table 2.

At this point we would like to stress, that the two main factors limiting the performance of the HEC coprocessor are the AU and the interconnect network (see discussion in previous sections).

Type 1 Design: High Performance: Our Type 1 design aims for a high performance implementation and is shown in Figure 3(a). This design has two independent arithmetic units; one for group addition and one for group doubling. The independency of the two group operations results from separate field operation units, registers, interconnect networks, and control units. Thus, we can compute the group operations in parallel by using a modified version of the double-and-add scalar multiplication. In the case of group addition we used two multipliers and one inversion logic, whereas for group doubling we provided only one multiplier and one inversion logic.

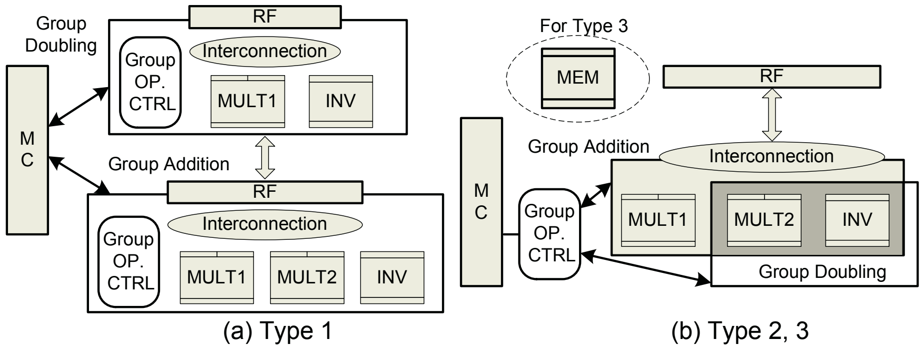


Fig. 3. Various design options for the HECC coprocessor.

Type 2 Design: Resource Sharing: Our Type 2 HECC coprocessor provides only one AU shared by the group addition and group doubling (see Figure 3(b)). Hence, the two group operations share the field arithmetic unit, the register file, the interconnect network, and the control logic. Another difference is that a more complex interconnect network is needed to handle the computation. These differences result in a slower operating frequency, however, the total hardware complexity is smaller compared to Type 1.

Type 3 Design: Low Area: Type 3 HECC coprocessor also uses, like in the case of the Type 2 design, shared resources (see Figure 3(b)). The difference between Type 2 and Type 3 design is the usage of memory for storing the intermediate data needed during the computation of the group operations and the scalar multiplication. When we use memory instead of registers, the decoding logic for reading (writing) data from (to) the RF unit is intrinsically implemented inside the memory.

We have used distributed memory with dual ports (DIST_MEM_V6), which is internally provided by the Xilinx FPGAs. We have chosen the smallest memory block available, namely of the size 1,536 bits, however we are currently using only 1,246 bits to store 14 field elements.

Trading memory versus registers, results in higher frequency and smaller area, however, there are two disadvantages: a) we use specific memory which will be costly when converting the system to an ASIC and b) the number of clock cycles for the overall computation increases, because of the expensive data movement from and to the memory. The latter disadvantage was partially removed by applying pipelining technique to the HECC coprocessor. For example, one can move data to or from the memory while performing a field multiplication.

6 Results and Analysis

In this section we present our implementation results using the stated methodology above. In the first part, we present our throughput results on the different designs. In the second part, we put our results in perspective to previous published ECC and HECC implementations.

6.1 Results

We implemented the HECC coprocessor using the formulae for the group operations [16, 22]. The coprocessor computes the most frequent case using the suggested special curve parameter and the fixed underlying field GF(2^{89}). It was modeled using VHDL language and then implemented with a Xilinx Virtex II FPGA (XC2V4000ff1517-6). The VHDL code was synthesized using Synplicity's Synplify Pro 7.3.1 and Xilinx Foundation 5.2.03i to implement the modeled

Table 2. Architectural characteristics of the different HECC coprocessor types.

	logic	interconnection	scalar mult.	Storage for RF
Type 1	addition: 2 MUL, 1 INV	Multiplexers	Right to Left	13 registers
	doubling: 1 MUL, 1 INV		(parallel)	10 registers
Type 2	2 MUL, 1 INV (shared)	Multiplexers	Left to Right	14 registers
Type 3	2 MUL, 1 INV (shared)	Multiplexers BUS	Left to Right	Memory (1,536 bits)

HECC coprocessor onto the target FPGA. Our HECC coprocessor used between 43% and 83% of the slices available in the FPGA.

Table 3 shows the area and time requirements for the three different design options. Type 1 corresponds to our high performance implementation. We were able to compute the scalar multiplication in $436\mu s$, which is about 78.5% faster than the best known implementation presented in [7]. The area requirements decrease almost 50% changing from Type 1 to Type 3. Hence, our Type 3 design utilizes 22.7% of the area that the smallest design described in [7]. Note, that the results presented in [7], present the best previous results in terms of area and performance. However the authors used a different coordinate system, coprocessor architecture, group order, and explicit formulae.

Table 3. Performance of HECC coprocessors at the level of scalar multiplication (Xilinx FPGA XC2V4000 ff1517-6, group order: 2^{178}.

	Size [*slices*]		Frequency [MHz]	Clock cycles	Time [μs]
Type 1	9,950	4,437 FFs	62.9	27,410	436
		16,459 LUTs			
Type 2	7,096	2,702 FFs	50.1	39,630	791
		13,276 LUTs			
Type 3	4,995	2,178 FFs	50.5	51,550	1,020
		8,451 LUTs			

6.2 Analysis

Many publications evaluating the performance of cryptographic implementations compare *only* the throughput of the implementation. In our opinion, one should also consider the amount of hardware resources consumed to achieve the previously mentioned throughput, for example, by using the area-time (AT) product. Therefore, the optimal implementation will achieve the highest throughput in the least amount of area and, thus, the *lowest* area-time product. At this point, we must caution the reader against using the area-time product to compare implementations on different FPGA devices. This is because even within the same family one will get different timing results as a function of available logic and routing resources.

In order to be able to provide a fair comparison between our work and the once previously presented, we did the place and routing for the HECC coprocessor with a target FPGA, Xilinx Virtex II FPGA (XC2V4000ff1517-6). In the case of XC2V4000, our designs used between 21% and 43% of the available slices. Table 4 shows all HECC hardware numbers and the two best known FPGA implementation of ECC. Note that we did not calculate the area-time product for the two ECC implementations, because in [21], the authors used a different type of FPGA, namely the Xilinx FPGA XCV400E.

Analyzing the AT product numbers of our designs, we noticed that they are fairly similar. Thus, considering different application scenarios, e.g. high

Table 4. Comparison of ECC and HECC implementation on FPGA.

	group order	digit size	slices	f [MHz]	Time [μs]	AT
genus-2 HECC						
Clancy [5]	2^{166}	D=1	22,000	-	10,000	50.74
		D=4	60,000	-	9,000	124.54
Elias et al. [7]	2^{226}	D=1	21,550	45.6	7,390	36.73
		D=4	25,271	45.3	2,030	11.83
our work						
Type 1	2^{178}		9,950	62.9	436	1.00
Type 2		D=32	7,096	50.1	791	1.30
Type 3			4,995	50.5	1,020	1.18
ECC						
Orlando et al. [21]	2^{167}	D=16	1,501	76.7	210	-
Gura et al. [6]	2^{163}	D=64	11,845	66.4	143	0.4

speed or moderate area, we are able to provide an adequate solution. Comparing our results to the previously published, one realizes that our designs are an order of magnitude better. Our implementations perform between a factor of 12 and 125 better than previous implementations. Note that the authors in [7] used a different underlying field and, therefore, the factor will be smaller when changing the field. Considering our HECC coprocessors we are now approaching the performance range of ECC FPGA implementations.

The reasons for our improvements are manifold. One of the main reasons is the effort that we put into exploring the parallelism of the HECC in order to achieve the best utilization of the arithmetic units. In addition, we applied various implementation techniques such as loop unfolding on inversion logic, pipelining on field operations and data movements between registers, minimization of hardware complexity in interconnect network and register usage, to advance the speed of the encryption and to lower the area complexity. Furthermore, we used affine coordinates and the most recent and therefore most efficient explicit formulae available. The group operation using affine coordinates are not as complex as the one based on projective coordinates, resulting in a smaller interconnect network.

7 Conclusions

In this paper we have implemented three different designs of a HECC coprocessor, ranging from the high performance to moderate area designs. The choice of the parameters of our efficient implementation is as follows: underlying field GF(2^{89}), curve parameters $h(x) = x$ and $f = x^5 + f_1x + f_0$, and we used explicit formulae based on affine coordinates [16,22]. Moreover, the implementation supports only the most frequent case.

In the case of high performance HECC coprocessor we computed the group operations in parallel and could achieve almost 78.5% speed up to the best timings published. In the case of our low area design we presented a solution for constrained environments. We were able to decrease the area utilization by more than 77.3% compared to the smallest HECC coprocessor published. Considering the area and time requirements for our HECC coprocessors, we are over a factor of 12 better than previous hardware implementations. To achieve these results we parallelized the group operations, minimized the complexity of the interconnect network, and decreased the number of registers.

We provided a milestone by approaching the performance of ECC coprocessors with our HECC implementations. We show that genus-2 HECC coprocessors are potentially well suited for FPGA implementation, because of a) the small field size which allows highly parallel arithmetic units and b) the parallelism in the explicit formulae computing the group operations.

References

1. G. Bertoni, L. Breveglieri, T. Wollinger, and C. Paar. Finding Optimum Parallel Coprocessor Design for Genus 2 Hyperelliptic Curve Cryptosystems. In *International Conference on Information Technology: Coding and Computing - ITCC 2004*. IEEE Computer Society, April 2004.
2. N. Boston, T. Clancy, Y. Liow, and J. Webster. Genus Two Hyperelliptic Curve Coprocessor. In *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume LNCS 2523. Springer-Verlag, 2002.
3. D.G. Cantor. Computing in Jacobian of a Hyperelliptic Curve. In *Mathematics of Computation*, volume 48(177), pages 95 – 101, January 1987.
4. T. Clancy. Analysis of FPGA-based Hyperelliptic Curve Cryptosystems. Master's thesis, University of Illinois Urbana-Champaign, December 2002.
5. T. Clancy. FPGA-based Hyperelliptic Curve Cryptosystems. invited paper presented at AMS Central Section Meeting, April 2003.
6. H. Eberle, N. Gura, and S. Chang-Shantz. A Cryptographic Processor for Arbitrary Elliptic Curves over GF(2^m). In *IEEE International Conference on Application Specific Systems Architectures and Processors — ASAP 2003*, 2003.
7. G. Elias, A. Miri, and T. H. Yeap. High-Performance, FPGA-Based Hyperelliptic Curve Cryptosystems. In *The Proceeding of the 22nd Biennial Symposium on Communications*, May 2004.
8. G. Frey. How to disguise an elliptic curve. Talk at ECC 1998, 1998. `http://cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html`.
9. P. Gaudry and R. Harley. Counting Points on Hyperelliptic Curves over Finite Fields. In *ANTS IV*, LNCS 1838, pages 297 – 312, 2000.
10. P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15(1):19–46, 2002.
11. D. Hankerson, J. López Hernandez, and A. Menezes. Software Implementation of Elliptic Curve Cryptography Over Binary Fields. In *Second International Workshop on Cryptographic Hardware and Embedded Systems — CHES 2000*, LNCS 1965, pages 1–24, Berlin, 2000. Springer-Verlag.
12. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
13. N. Koblitz. A Family of Jacobians Suitable for Discrete Log Cryptosystems. In Shafi Goldwasser, editor, *Advances in Cryptology - Crypto '88*, LNCS 403, pages 94 – 99, Berlin, 1988. Springer-Verlag.

14. N. Koblitz. *Algebraic Aspects of Cryptography.* Springer-Verlag, Berlin, Germany, first edition, 1998.
15. T. Lange. Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/147, 2002. http:eprint.iacr.org.
16. T. Lange. Formulae for Arithmetic on Genus 2 Hyperelliptic Curves, September 2003. http://www.ruhr-uni-bochum.de/itsc/tanja/preprints/expl_ sub.pdf.
17. A. Menezes, Y. Wu, and R. Zuccherato. *An Elementary Introduction to Hyperelliptic Curves.* Springer-Verlag, Berlin, Germany, first edition, 1998.
18. V. Miller. Uses of Elliptic Curves in Cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, LNCS 218, pages 417–426, Berlin, Germany, 1986. Springer-Verlag.
19. P. K. Mishra and P. Sarkar. Parallelizing Explicit Formula for Arithmetic in the Jacobian of Hyperelliptic Curves. In *Asiacrypt '03*, LNCS 2894, pages 93–110, November 2003.
20. D. Mumford. Tata lectures on theta II. In *Prog. Math.*, volume 43. Birkhäuser, 1984.
21. G. Orlando and C. Paar. A High-Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$. In *Cryptographic Hardware and Embedded Systems — CHES 2000*, LNCS 1965. Springer-Verlag, 2000.
22. J. Pelzl, T. Wollinger, and C. Paar. High Performance Arithmetic for Special Hyperelliptic Curve Cryptosystems of Genus Two. In *International Conference on Information Technology: Coding and Computing - ITCC 2004*. IEEE Computer Society, April 5-7 2004.
23. L. Song and K. K. Parhi. Low-Energy Digit-Serial/Parallel Finite Field Multipliers. *Journal of VLSI Signal Processing Systems*, 2(22):1–17, 1997.
24. N. Thériault. Index Calculus Attack for Hyperelliptic Curves of Small Genus. In *Advances in Cryptology - ASIACRYPT '03*, pages 79 – 92, 2003.
25. T. Wollinger. Computer Architectures for Cryptosystems Based on Hyperelliptic Curves. Master's thesis, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2001.
26. T. Wollinger. *Software and Hardware Implementation of Hyperelliptic Curve Cryptosystems.* PhD thesis, Department of Electrical Engineering and Information Sciences, Ruhr-Universitaet Bochum, Bochum, Germany, July 2004.
27. Y.J.Choi, H.W.Kim, and M.S.Kim. A Design and Implementation of the ECC Crypto Processor. Technical Report, ETRI, DaeJeon, Korea, September 2002.

Key-Exchange Protocol Using Pre-agreed Session-ID

Kenji Imamoto and Kouichi Sakurai

Kyushu University, Fukuoka, Japan
imamoto@itslab.csce.kyushu-u.ac.jp, sakurai@csce.kyushu-u.ac.jp
http://itslab.csce.kyushu-u.ac.jp/

Abstract. Technical applications for various uses have been proposed in communication technology in recent years. Although especially the Internet and radio communications are used daily it is known that eavesdropping is easy and the related problem has occurred mostly, especially privacy. In this paper, we introduce *Pre-Agreed Session ID* (PAS) and formalize a key-exchange protocol using it. PAS is a identification which is a disposable unique value used for every session to protect identity from the attacker. The security notion of identity concealment is formulized in this paper. Moreover, we propose a secure key-exchange protocol using PAS under the cryptographic primitives. Furthermore, we argue about the problems which arise when PAS is introduced.

Keywords: Diffie-Hellman based key exchange protocol, Identity concealment, Pre-shared key model, Pre-Agreed Session ID.

1 Introduction

Key-exchange protocols are procedures which enable two parties that communicate over open network (e.g., the Internet) to securely share a common short-term key. To design secure key-exchange protocols is difficult even if completely secure cryptographic primitives exist. The aim of this paper is to construct a formal security model for key-exchange and present a secure protocol (from the definition of our model) under some well known cryptographic primitives.

One of the most important tasks in the provable security area is the design of a security model. Many cryptographers has been discussing, and a number of papers has been presented in this area [2, 3, 12, 1]. In this paper, we especially focus on two papers [2, 3] related to the problem of identity concealment. Identity concealment is the hottest topic from the view of practical applications because of the recently rapid development of mobile network such as mobile phone, RFID, wireless LAN, and so on [9, 10, 8].

Key-exchange protocol can be divided into two types from the possible availability of the peer identity: one is *pre-specified peer* and the other is *post-specified peer* [3]. In the pre-specified peer setting, the peer identities are assumed to be specified and given at the onset of a session activation. Then, the task of the protocol is to guarantee that the party that you are talking to is certainly the pre-specified party. Unlike this, each party learns the peer's identity during the protocol in the post-specified peer setting. A secure protocol in this setting ensures that the learned identity is authentic. Hence, post-specified peer is more general setting than pre-specified one.

The protocol proposed in [3] is a signature-based key exchange and is proved to be secure under the post-specified setting. However, it has two problems: one issue is that it cannot hide an identity of either an initiator or a responder of the protocol from active attacker. The other one is that a system using this protocol can be fragile against denial of service (DoS). DoS attack is the attack method which blocks the service currently offered by the server and causes the remarkable fall of the performance of service. This attack is also an important problem and discussed widely [5, 11].

To solve identity concealment problem, we introduce a notion "*Pre-Agreed Session-ID*" (PAS for short), which is a unique session identifier determined and agreed between each peer before activation of the session. Session-id is used in [2, 3] to uniquely name sessions, and assumed to be unique among all the session id. Similarly, since PAS is unique and agreed privately between each pair before the session, a legitimate party can specify the party who requests a response by checking the corresponding PAS. On the other hand, an attacker cannot guess the peer identities because he/she does not know the relation between PAS and identities. As a direct result of this, by using PAS as party's identity, identity concealment can be provided even if an active attacker exists. Moreover, this property can also result in improving the resistance against DoS attack. We argue it later.

To use PAS for identity concealment, we design a protocol based on preshaerd key model under pre-specified setting. This is the reason why the choice of PAS for intended party needs to specify the party before the activation of the protocol (we generally assumed that each party has plural PAS for several parties). As mentioned above, post-specified setting is more general than pre-specified setting. However, post-specified setting based on pre-shared key model is a practical and common case. That is, in most of protocols using pre-shared key, a session is started when the initiator has recognized the partner.

Paper's Organization. In section 2, we introduce our security model based on the model of [2]. In section 3, we present our protocol (called PAS protocol) which is secure under our definitions. In section 4, we present the formal proof of the proposed protocol in the model introduced in section 2. In section 5, we discuss some problems and properties related to our protocol. Finally, we conclude this paper in section 6.

2 Security Model

Here, we present our model for key-exchange protocols. This model is based on the model of [2], but modified to include the notion of PAS and users' privacy.

2.1 The SK-ID-Security Definition

Each activation of the protocol at a party instantiates a run of the protocol and produces outgoing messages and processes incoming messages. In key-exchange protocol, a session is intended to agree on a session key with one other party. Sessions can run concurrently and incoming messages are directed to its corresponding session via a session identifier. The activation of m-th session at a party P_i has three input parameters (P_i, PAS_{ij}^m, P_j): the local party at which the session is activated, the pre-agreed session-id, and the identity of the intended peer to the session. The output of the session at P_i is (P_i, P_j, PAS_{ij}^{m+1}, the session key). Each party has additional state, such as a list of long-term pre-shared keys, which is accessed during different sessions.

The attacker is a (probabilistic) polynomial-time machine with full control of the communication lines between parties and free to intercept, delay, drop, inject, or change all messaages sent over these lines. The channel is broadcast-type: all messages can be sent to a pool of messages, to a local broadcast network, to a physical or logical address, etc. This model does not make any assumption on who will eventually get a message, how many times, and when (these are all actions decided by the attacker). Also, there is no assumption on the logical connection between the address where a message is delivered and the identity behind that address. The attacker knows all the pairs that prepare a pre-shared key.

In addition, the attacker can have access to secret information via *session exposure* attacks of four types: session-state reveal, session-key query, party corruption, and identity reveal. The first type of attack is directed at a session while still incomplete, and its result is that the attacker learns the session state for that particular session (which does not include long-term secret). The second one is directed at a completed session, and its result is that the attacker learns the corresponding session-key. Party corruption means that the attacker learns all information in the memory of the party (including session states, session-key, long-term secrets). Finally, identity reveal means that the attacker learns the parties' identities that activate the session.

Sessions can be *expired* like the model of [2, 3]. From the time a session is expired, the attacker is not allowed to perform a session-key query or a state-reveal attack against the session, but is allowed to corrupt the party that holds the session.

For defining the security of a key-exchange protocol, we follow the indistinguishability style as used in a lot of papers related to provable security where the success of an attack is measured via its ability to distinguish the real values of session keys from independent random values, or distinguish the real pair of the session from a randomly chosen pair. In this time, the session should not be exposed by any of the above attacks but identity reveal is not allowed against test session for distinction of pairs (explained later). Moreover, the attacker is prohibitted from exposing the matching session, where the input of two sessions (P, s, Q) and (P', s', Q') are called matching if $s = s'$, $P = Q'$, and $Q = P'$.

The reason why an attacker is not allowed to expose matching session is as follows. If an attacker can perform session-key query against a matching session, she can easily distinguish real session key of the corresponding test session from random value because the session key of the matching session is the same as the session key of the test session. Similarly, if an attacker can perform identity reveal against a matching session, she can easily distinguish real pair of the corresponding test session from randomly chosen pair because the pair who participates the matching session is the same as the pair who participates the test session.

The ability of the attacker to distinguish between real and random is formalized via the notion of a *test session* that the attacker is free to choose among all complete sessions in the protocol. Test session has two games: distinction of session-key and distinction of pairs. In the distinction of sesson-key, before the selection of test session, the attacker can perform identity reveal against any session. When the attacker chooses the test session it is provided with a value v which is chosen as follows: a random bit b is tossed, if $b = 0$ then v is the real value of the output session key, otherwise v is a random value chosen under the same distribution of session keys. After receiving v the attacker may continue with the regular actions against the protocol; at the end of its run the attacker outputs a bit b'. The attacker succeeds in its attack if (1) the test session is not exposed (except for identity reveal), and (2) the probability that $b = b'$ is significantly larger than 1/2.

On the other hand, in the distinction of pairs, the attacker is not allowd to perform identity reveal against test session. When the attacker chooses the test session it is provided with identities (A, B) which is chosen as follows: a random bit b is tossed, if $b = 0$ then (A, B) is the true pair of the session, otherwise (A, B) is a pair chosen in a "suitable" way within the list of all pairs that prepare a pre-shared key. After receiving (A, B) the attacker may continue with the regular actions against the protocol; at the end of its run the attacker outputs a bit b'. The attacker succeeds in its attack if (1) the test session is not exposed, and (2) the probability that $b = b'$ is significantly larger than 1/2.

– Here, we consider the method of suitable choice of a random pair. Suppose the attacker knows one of the two parties P_i that activate the test session. Then, since the attacker knows that the true pair must include P_i, the attacker will succeed if a random pair does not include P_i. But if a random pair also includs P_i, the attacker may fail the game. From this reason, we can propose two method of choice of a random pair.
  1. Random choice from all possible pairs except for the real pair
  2. Random choice from all possible pairs that do not include either of the real partys' identities

  The game with the former choice method means that the attacker should guess both parties' identities, while the game with the latter one means that the attacker can win even if only one of the parties can be distinguished (if there are no pair in a list of candidates for random choice, then the attacker wins). Because our aim is to protect both parties' identity, we apply the latter way as "suitable" choice to our model. (Note that, in the case that each party shares a long-term secret with only one special party, e.g., server-client system, the attacker can always win the game using the latter way)

The resultant security notion for key-exchange protocol is called SK-ID-security and is stated as follows:

Definition 1. (SK-ID-security) *An attacker with the above capabilities is called an SK-ID-attacker. A key-exchange protocol π is called SK-ID-secure if for all SK-ID-attackers $\mathcal{A}$ running against π it holds:*

1. *If two uncorrupted parties complete matching sessions in a run of protocol π under attacker $\mathcal{A}$ then, except for a negligible probability, the session key output in these sessions is the same.*
2. *$\mathcal{A}$ succeeds (in its test-session distinguishing attacks) with probability not more than 1/2 plus a negligible fraction.*

3 PAS Protocol

Here we provide a description of our pre-shared key based key-exchange protocol using Pre-Agreed Session-id (we call it PAS protocol). In the next section we provide a proof of this protocol, and in the subsequent section we will pick up some topics about which we should argue. The PAS protocol is as follows.

Initial Information: Primes p, q, $q|p-1$, and g with order q in Z_p^*. Each player has long-term secret key(s) shared with his/her partner(s). The protocol uses a message authentication family MAC, and a pseudorandom function family PRF.

The Protocol Messages: (in m-th session)

– Start message ($I \to R$): PAS_{ij}^m, g^x
– Response message ($R \to I$): $PAS_{ij}^m, g^y, MAC_{k_2}(\text{“1”}, PAS_{ij}^m, P_i, g^y, g^x, g^{xy})$
– Finish message ($I \to R$): $PAS_{ij}^m, MAC_{k_2}(\text{“0”}, PAS_{ij}^m, P_j, g^x, g^y, g^{xy})$

The Protocol Actions: (in m-th session)

1. An initiator P_i selects ID_j as a responder of the key-exchange (ID_j is one of the parties who have already prepared a pre-shared key, PSK_{ij}, and pre-agreed session-id, PAS_{ij}^m, between P_i). The start message is sent by the initiator P_i upon activation with session-id PAS_{ij}^m (after checking that no previous session at P_i was initiated with identifier PAS_{ij}^m); the DH exponent g^x is computed with $x \overset{R}{\leftarrow} Z_q$ and x is stored in the state of session (P_i, s).
2. When a (first) start message with session-id PAS_{ij}^m is delivered to a party P_j, P_j retrieves PSK_{ij} corresponding to PAS_{ij}^m and if session-id PAS_{ij}^m did not exist before at P_j, P_j activates a local session PAS_{ij}^m (as responder). It now generates the response message where the DH exponent g^y is computed with $y \overset{R}{\leftarrow} Z_q$, and produces the MAC_{k_2} value with k_2 where g^{xy} is computed

by P_j as $(g^x)^y$. Here, g^x is the value contained in the start message. Finally, the value $k_0 = PRF_{g^{xy}}(\text{“0”})$ and $k_1 = PRF_{g^{xy}}(\text{“1”})$ are computed and kept in memory, and the values y and g^{xy} are erased.

3. Upon receiving a response message with session-id PAS^m_{ij}, if P_i has never received a valid response message with PAS^m_{ij} beforehand, then P_i retrieves PSK_{ij} corresponding to PAS^m_{ij} contained in the response message and verifies the MAC on the quadruple $(\text{“1”}, PAS^m_{ij}, P_i, g^y, g^x, g^{xy})$ under key $k_2 = PRF_{PSK_{ij}}(\text{“2”})$. Here, g^x is the value sent by P_i in the start message, g^y is the value received in this response message, and g^{xy} is computed by P_i as $(g^y)^x$. If this verification step fails the session is aborted. Otherwise, P_i regards the message as a valid response and completes the session with secret session key k_0 computed as $k_0 = PRF_{g^{xy}}(\text{“0”})$ and the pre-agreed session-id for the next session k_1 (i.e., PAS^{m+1}_{ij}) computed as $k_1 = PRF_{g^{xy}}(\text{“1”})$. The finish message is sent and the session state erased.
4. Upon receiving the finish message of session PAS^{m+1}_{ij}, P_j verifies the MAC value on the quadruple $(\text{“0”}, PAS^m_{ij}, P_j, g^y, g^x, g^{xy})$ under key k_2 computed in step 2 and with g^y being the DH value that P_j sent in the response message. If this verification step fails the session is aborted. Otherwise, P_j completes the session with secret session key k_0 and the pre-agreed session-id for the next session k_1. The session state is erased.

4 Proof of PAS Protocol

4.1 The Statements

We start by formulating the Decisional Diffie-Hellman (DDH) assumption which is the standard assumption underlying the security of the DH key exchange against passive attackers. For simplicity, we formulate this assumption for a specific family of DH groups, but analogous assumptions can be formulated for other groups.

Assumption 2. *Let κ be a security parameter. Let p,q be primes, where q is of length κ bits and $q|p-1$, and g be of order q in Z^*_p. Then the probability distributions of quintuples*
$Q_0 = \left\{\langle p, g, g^x, g^y, g^{xy}\rangle : x, y \overset{R}{\leftarrow} Z_q\right\}$ *and* $Q_1 = \left\{\langle p, g, g^x, g^y, g^r\rangle : x, y, r \overset{R}{\leftarrow} Z_q\right\}$
are computationally indistinguishable.

In addition to DDH assumption, we will assume the security of the other underlying cryptographic primitives in the protocol (i.e., message authentication codes, and pseudorandom functions) under the standard security notions in the cryptographic literature.

Theorem 3 (Main Theorem). *Assuming DDH and the security of the underlying cryptographic functions (i.e., MAC and PRF), PAS protocol is SK-ID-secure as defined in Section 2.*

In order to prove the main theorem, we need to prove the following three properties of SK-ID-secure protocols. (we use the term PAS-attacker to denote an SK-ID-attacker working against the PAS protocol):

P1. If two uncorrupted parties P_i and P_j complete matching sessions ((P_i, s, P_j) and (P_j, s, P_i), respectively) under the PAS protocol then, except for a negligible probability, the outputs of each party (that is, the session key and the pre-agreed session-id) in these sessions are the same.

P2. No efficient PAS-attacker can distinguish a real session key to the test-session query from a random response with non-negligible advantage. More precisely, if for a given PAS-attacker we define:
- $P_{KREAL}(\mathcal{A})$=*Prob*($\mathcal{A}$ outputs 1 when given the real test session key)
- $P_{KRAND}(\mathcal{A})$=*Prob*($\mathcal{A}$ outputs 1 when given a random test session key)

then we need to prove that for any PAS-attacker $\mathcal{A}$:

$$|P_{KREAL}(\mathcal{A}) - P_{KRAND}(\mathcal{A})| \text{ is negligible.}$$

P3. No efficient PAS-attacker can distinguish a real pair who activates the test-session from another randomly chosen pair with non-negligible advantage. More precisely, if for a given PAS-attacker we define:
- $P_{PREAL}(\mathcal{A})$=*Prob*($\mathcal{A}$ outputs 1 when given the real pair of the test session)
- $P_{PRAND}(\mathcal{A})$=*Prob*($\mathcal{A}$ outputs 1 when given a randomly chosen pair of the test session)

then we need to prove that for any PAS-attacker $\mathcal{A}$:

$$|P_{PREAL}(\mathcal{A}) - P_{PRAND}(\mathcal{A})| \text{ is negligible.}$$

In section 4.2, 4.3, and 4.4, we show the proof (or proof sketch) that our protocol can realize above properties (i.e., P1, P2, P3). From this result, the main theorem can be proven.

4.2 Proof of Property P1

Proof: Let $\mathcal{A}$ be a PAS-attacker, and let P_i and P_j be two uncorrupted parties that complete matching sessions (P_i, s, P_j) and (P_j, s, P_i). Clearly, property P1 can be proved by showing that both parties compute the same DH value g^{xy} because the session key k_0 and the pre-agreed session-id are deterministically derived from it. Let us denote u_i the DH exponent sent in the start message by P_i where $u_i = g^{x_i}$ with x_i chosen by P_i, and let v_i denote the DH exponent received in the response message of session s (P_i must receive it to complete the session). Similarly, let u_r be the DH exponent received by P_j in the incoming start message of session s, and v_r be the DH exponent sent by P_j in its response message where $v_r = g^{x_r}$ with x_r chosen by P_j.

The MAC value produced by P_j during session s is $MAC_{k_2}(\text{“1”}, s, P_i, u_r, v_r, w_r)$, while the MAC value P_i verifies in the response message is $MAC_{k_2}(\text{“1”}, s, P_i, u_i, v_i, w_i)$. Since the first MAC value is the only one that P_j ever produces with the value s as the session id, then it must be that either all arguments to the first and second MAC value are the same, or a valid MAC containing the pair (u_i, v_i, w_i) was produced by the attacker even though P_j did not generated such a MAC value. If the later case happens with non-negligible probability then we can use the attacker $\mathcal{A}$ under a simulation of the basic protocol to produce a forger for the underlying MAC scheme. Since we assume MAC to be a secure scheme this event must have negligible probability. Therefore, we get that except for such a negligible probability, $u_r = u_i$ and $v_r = v_i$.

Now the DH key computed by P_i is $v_i^{x_i} = v_r^{x_i} = (g^{x_r})^{x_i} = g^{x_i x_r}$, while the DH key computed by P_j is $u_r^{x_r} = u_i^{x_r} = (g^{x_i})^{x_r} = g^{x_i x_r}$. Therefore, both parties compute the same session key and the pre-agreed session-id for the next session.

4.3 Proof Sketch of Property P2

We prove P2 by showing that if a PAS-attacker can win the game with significant advantage then we can build an attacker against one of the underlying cryptographic primitives used in the protocol (the plan is almost the same as [3]).

Now we will show that from any PAS-attacker $\mathcal{A}$ that succeeds in distinguishing between a real and a random response to the test-session query we can build a DDH distinguisher D that distinguishs triple (g^x, g^y, g^{xy}) from (g^x, g^y, g^r) with the same success advantage as $\mathcal{A}$. D gets (g^x, g^y, z) as input where z is either g^{xy} or g^r for $r \overset{R}{\leftarrow} Z_q$. D starts by simulating a run of $\mathcal{A}$ on a virtual run of PAS protocol, and uses the value g^x and g^y as the DH exponents in the start and response message of one randomly chosen session, say s_0, initiated by $\mathcal{A}$. The idea is that if $\mathcal{A}$ chooses this session s_0 as its test session, then D can provide z as the response to the test-session query by $\mathcal{A}$. If $\mathcal{A}$ outputs that the response was real then D will decide that $z = g^{xy}$, otherwise D will decide that z is random.

4.4 Proof Sketch of Property P3

To prove P3, we have to show that if a PAS-attacker exists that can distinguish the real pair who activates the session chosen by the attacker from a randomly chosen pair with significant advantage, then we can built an attacker against the underlying cryptographic primitive. Unlike the proof of P2, we don't show the direct reduction of the difficulty of the distinction of pairs to the difficulty of the primitive. The proof sketch of P3 is as follows. First, we introduce new game "the distinction of PAS":

– When the attacker chooses the test session it is provided with a value v which is chosen as follows: a random bit b is tossed, if $b = 0$ then v is the real value of the output PAS, otherwise v is an output PAS of a session activated after the completion of the test session. After receiving v the attacker may continue with the regular actions against the protocol, however, the attacker is allowed to perform identity reveal against arbitrary sessions except for sessions activated after the completion of the test session. At the end of its run the attacker outputs a bit b'. The attacker succeeds in its attack if (1) the test session is not exposed, and (2) the probability that $b = b'$ is significantly larger than 1/2.

Here we will show that from any PAS-attacker $\mathcal{A}$ that succeeds in distinguishing between a real and a random response to the above game query we can build a DDH distinguisher D that distinguishs triple (g^x, g^y, g^{xy}) from (g^x, g^y, g^r) with the same success advantage as $\mathcal{A}$. D gets (g^x, g^y, z) as input where z is either g^{xy} or g^r for $r \overset{R}{\leftarrow} Z_q$. D starts by simulating a run of $\mathcal{A}$ on a virtual run of PAS protocol, and uses the value g^x and g^y as the DH exponents in the start and response message of one randomly chosen session, say s_0, initiated by $\mathcal{A}$. The idea is that if $\mathcal{A}$ chooses this session s_0 as its test session, then D can provide $PRF_z(0)$ as the response to the test-session query by $\mathcal{A}$. If $\mathcal{A}$ outputs that the response was real then D will decide that $z = PRF_{g^{xy}}(0)$, otherwise D will decide that z is random.

Next, we will show that from any PAS-attacker $\mathcal{A}'$ that succeeds in distinguishing between a real and a random pair (in the game of the distinction of the pair) we can build any PAS-attacker $\mathcal{A}$ that succeeds in distinguishing between a real and a random response to the above game query with the same success advantage as $\mathcal{A}'$. A gets (g^x, g^y, z) as input where z is either $PRF_{g^{xy}}(1)$ or g^r for $r \overset{R}{\leftarrow} Z_q$. A starts by simulating a run of $\mathcal{A}'$ on a virtual run of PAS protocol, and randomly chose a session s'_0 initiated by $\mathcal{A}$. The idea is that if $\mathcal{A}$ chooses this session s'_0 as its test session, then A can provide (P_i, P_j, z) as the response to the test-session query by $\mathcal{A}'$ (since A knows the real parties by an execution of identity reveal against the test session, A can give real parties' identities in the response). If $\mathcal{A}'$ outputs that the response was real then A will decide that $z = PRF_{g^{xy}}(1)$, otherwise D will decide that z is random.

In consequence of the above discussion, we can built a DDH distinguisher D by using a PAS distinguisher $\mathcal{A}$. Moreover, we can built $\mathcal{A}$ by using a pair distinguisher $\mathcal{A}'$. Therefore, if $\mathcal{A}'$ exists, we can built D, i.e., the DDH assumption can be broken. However, since DDH is assumed to be broken with negligible advantage, the probability of distinguishing a real or random pair is also negligible advantage. Therefore, we can derive property P3.

5 Variants and Discussions

5.1 Synchronization

Due to the use of pre-agreed session-id, our protocol can be regarded as a statefull key-exchange, which results in the need of maintenance of session state. One of the unavoidable problems related to the maintenance is synchronization of the session state between honest parties. Especially in our method, failing in synchronization of PAS causes a possibility that a session cannot be newly activated because legitimate PAS is needed in each message. Therefore, we argue a countermeasure against this problem.

We can guess some origins which cause unsynchronization between honest parties such as an interruption of the protocol, a misunderstanding of their secret, a failure of their devices, etc. To simplify our analysis, let us assume, however, that the causes of the problem are as follows.

- PAS attacker impersonates a honest party and renews its PAS (the victim cannot know new PAS)
- An interruption of the protocol due to a network accident or an attack
- Failing in generating PAS for next session

This restriction of the situations enables us to deal with the synchronization problem. Moreover, under some assumptions, we can guarantee any lag of PAS is within a session. Since the first and third causes can be clearly prevented because of P1 of Main Theorem under the assumptions of cryptographic primitives, we consider the second one in more detail.

Using a situation where a message could be vanished or a session may be aborted at any time, each party could believe what happened as follows. Here, we consider the cases of our protocol from the view of an initiator and a responder.

- In the case of an initiator, P_i
  By verification of the response message with PAS_{ij}^{m}, P_i can believe P_j has correctly computed the session key and PAS_{ij}^{m+1} then, the identifier of the previous session, PAS_{ij}^{m-1}, can be erased from its memory. After that, P_i sends the finish message, however P_i cannot believe P_j's "confirmation" because there is no assumption on the connection (it may be vanished).
- In the case of a responder, P_j
  By verification of the finish message with PAS_{ij}^{m}, P_j can believe P_i has correctly computed the session key and PAS_{ij}^{m+1}, then the identifier of the previous session, PAS_{ij}^{m-1}, can be erased from its memory. However, by just the procedures of m-th session, P_j cannot make P_i believe that P_j has already confirmed because of the connection condition.

By the above analysis, we can conclude that the session state (i.e., PAS_{ij}^{m}) cannot be erased before P_j makes P_i believe its confirmation. That is, the synchronization problem can be solved by holding a session state until its confirmation.

5.2 DoS-Resilient

Here we discuss how to improve a resistance against DoS attack in a key-exchange protocol. There are three types of DoS attack: against responder's bandwidth, memory, and CPU. The purpose of the first attack is that a responder cannot receive any more message. The second one is performed to make a responder store large quantities of waste states. The last one is the attack which makes a responder compute a lot of quite inefficient processing. Since it is difficult to consider the DoS attack against bandwidth only in the area of a key exchange protocol, we consider the other attacks. (The first attack can be prevented in another layer [11].)

To analyse a countermeasure against DoS attack, we need to introduce new concepts of time and processing. But to simplify our analysis, we regard public key operations (e.g., public key encryption/decryption, Diffie-Hellman key exchange, digital signature) as inefficient processing.

Concepts of Time

- *fresh*: information generated within an acceptable time ("acceptable" is defined in each system)
- *new*: information generated after the activation of the session (this notion is used in [13, 14])

In other words, fresh information can provide absolute time, and new information can provide relative time. To check whether a message is fresh and/or new or not, there are three types of nonces, i.e., random number, sequence number, and timestamp. Each nonce must be controlled by a verifier. Table 1 shows properties each nonce can provide.

Table 1. Properties each nonce can provide.

	fresh	new
Random number	○	○
Sequence number		○
Timestamp	○	

By using these notions, we refine the well-known existing properties presented in cryptographic literature ([13, 14] do not have the notion of fresh).

- Entity authentication means that a party can believe that the intended peer sent a fresh message.
- Authentication means that a party can believe that the intended peer sent a fresh and new message.

Fresh information is required for authentication, but new information is required to prevent DoS attack against memory and CPU. This is because new information can make a responder store a session state and/or perform a processing once for all (fresh information may be used once and again within an acceptable time). From this reason, timestamp may not be a suitable tool to improve the tolerance to DoS attack (PAS could be seen as sequence number). The following procedures are needed in order to cope with it to DoS attack against memory and processing.

1. The responder does not respond any request, which asks inefficient processing, from a party the responder cannot specify. (in the case that identified party attempts on DoS attack, the access from the party can be disconnected).
2. The responder only responds to new requests, which ask inefficient processing, to prevent DoS attack based on replay attack.
3. The requester does not change his/her new demand even when aborted.
4. The responder responds new reply to a new request to prevent replay attack which uses new request once and again.

SIGMA is the system which has the low tolerance to DoS attack because it performs inefficient processing, such as g^y, and stores session state to its memory before the responer specifies its partner. Moreover, since it is not taken into consideration about the processing in the case of the aborted protocol by rewriting of the transmitting message, it is fragile against the attacker's obstruction.

On the other hand, our protocol firstly checks PAS to specify the partner, but a responder who receives a start message cannot verify whether a key-agreement material g^x is new or not. This results in that the protocol could be fragile (in the case that the responder rejects any requests for the session if a session is aborted) or low tolerance to DoS attack (in the case that the responder responds to any requests for the session even if a session is aborted).

In order to solve this problem, we slightly modify PAS protocol to verify that the key agreement material g^x received in the start message is new as follows.

The DoS-Resilient Protocol Messages:

– Start message ($I \rightarrow R$): $PAS_{ij}^m, g^x, MAC_{k_2}(PAS_{ij}^m, P_j, g^x)$
– Response message ($R \rightarrow I$): $PAS_{ij}^m, g^y, MAC_{k_2}(\text{“1”}, PAS_{ij}^m, P_i, g^y, g^x, g^{xy})$
– Finish message ($I \rightarrow R$): $PAS_{ij}^m, MAC_{k_2}(\text{“0”}, PAS_{ij}^m, P_j, g^y, g^x, g^{xy})$

Because a responder of this protocol verifies the MAC value contained in the start message, it can believe g^x is new, and it is generated by the initiator. The responder calculates g^y, g^{xy} and stores new session state after the confirmation of the start message. Moreover, the above processing is performed only against the first start message. In this time, the responder replies the same value sent in the first response message to any replay of the start message.

However, by this system, when PSK_{ij} is revealed, an attacker has the danger that correlation of communication will become possible though PAS realizes PFS (the following session explains), since $MAC_{k_2}(PAS_{ij}^m, P_j, g^x)$ is calculable from PSK and eavesdropping information. The measure for this problem is one of the future subjects.

5.3 Perfect Forward Secrecy

Informally, the notion of “Perfect Forward Secrecy” (PFS) is stated as the property that compromise of long-term secret does not compromise past short-secret such as session keys. In other words, it means that even if a party is corrupted then nothing is learned about sessions within that party that were previously unexposed and expired before the party corruption happened.

Since our protocol is based on Diffie-Hellman key exchange, any expired session keys and PAS enjoy PFS under the cryptographic primitives.

6 Conclusion

Based on the existing adversary model, this paper defined the security model which took privacy into consideration. In order to design secure key-exchange protocol, Pre-Agreed Session ID (PAS) which is a disposable identification was introduced for every session, and the key exchange protocol which protected identity on a party from the attacker was proposed. Moreover, it analysed about Synchronization, a DoS attack, and PFS as a problem which arises when PAS is introduced. As a future subject, we need to consider efficient countermeasure to DoS attack, and synchronization.

Acknowledgements

This research was partly supported from the grant of Secom Science and Technology Foundation, and the 21st Century COE Program 'Reconstruction of Social Infrastructure Related to Information Science and Electrical Engineering'. Also, the first author was partly supported from the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for JSPS Fellows, 2004, 06737.

References

1. V. Boyko, P. MacKenzie, and S. Patel, "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman", EUROCRYPT'2000, 2000.
2. R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", Advances in Cryptology - EUROCRYPT'2001.
3. R. Canetti and H. Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol", CRYPTO'2002.
4. W. Diffie and M. E. Hellman, "New directions in cryptography", IEEE Trans. Inform. Theory, IT-22:644–654, 1976.
5. D. Harkins, and D. Carrel, "The Internet Key Exchange (IKE)", RFC2409, 1998. http://www.ietf.org/rfc/rfc2409.txt
6. H. Krawczyk, "The IKE-SIGMA Protocol", Internet Draft, 2001. http://www.ee.technion.ac.il/ hugo/draft-krawczyk-ipsec-ike-sigma-00.txt
7. H. Krawczyk, "SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols", CRYPTO'2003.
8. M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to Privacy-Friendly Tags", RFID Privacy Workshop 2003.
9. R. Perlman, and C. Kaufman, "Analysis of IPSec Key Exchange Standard", WET-ICE2001, 2001.
10. R. Perlman, and C. Kaufman, "Key Exchange in IPSec: Analysis of IKE", 2001.
11. S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support For IP Traceback", SIGCOMM2000.
12. V. Shoup, "On formal models for secure key exchange", IBM Research Report RZ3120, 1999.
13. P. Syverson and P. C. van Oorschot, "A Unified Cryptographic Protocol Logic," NRL CHAOS Report, 1996.
14. P. Syverson and I. Cervesato, "The Logic of Authentication Protocols," FOSAD'00, LNCS2171, pp.63–137, 2001.

A New *k*-Anonymous Message Transmission Protocol

Gang Yao and Dengguo Feng

State Key Laboratory of Information Security
(Institute of Software of Chinese Academy of Sciences)
Beijing 100080, P.R.China
g.yao@is.iscas.ac.cn, fengdg@263.net

Abstract. Public and private communication networks have a growing importance for our daily life. The globally networked society places great demand on the dissemination and sharing of person-specific data. In order to protect the anonymity of individuals to whom released the data refer, data holders often remove or encrypt explicit identifiers such as names, addresses and phone numbers. But these cannot solve the problem well. Anonymous communication protocols address the problem of concealing who communicates with whom, as in the case of letters from a secret admirer. To gain efficiency, k-anonymous message transmission is presented.

Informally, a communication protocol is sender k-anonymous if it can guarantee that an adversary, trying to determine the sender of a particular message, can only narrow down its search to a set of k suspects. Receiver k-anonymity places a similar guarantee on the receiver: an adversary, at best, can only narrow down the possible receivers to a set of size k. In this paper, a k-anonymous transmission protocol is presented. The protocol is based on asymmetric encryption algorithm. All the members in the protocol is divided into smaller groups, and if all the members in the group perform the protocol correctly, the protocol is sender k-anonymous and receiver k-anonymous. Furthermore, as long as the asymmetric encryption algorithm is secure, our protocol is secure, too.

Keywords: anonymous transmission, cryptographic protocol, security

1 Introduction

Anonymity and data privacy are critical for many networked applications. Anonymity is shielding the user's identity from entities such as web servers, network providers and crackers. And data privacy is protecting the personal information of users from attack by outside sources. There are many reasons to hide your real identity when you use the Internet. Sometimes you want to send something without having your real name attached to it. For example, you might want to post personal messages to a Usenet newsgroup without identifying yourself to the whole world as the poster.

Anonymous communication protocols have been studied extensively in the scientific literature. These protocols address the problem of concealing who communicates with whom, as in the case of letters from a secret admirer. In the anonymous protocols, the adversary is allowed to see all the communications in the network, but he still cannot determine the sender or recipient of a message. Anonymous communication has many important applications in the real world, such as casting a ballot in a voting booth, engaging in a cash-based financial transaction, or getting tested for certain medical conditions.

The goal of most former research is usually to guarantee full anonymity. There are three types of anonymous communication properties that can be provided: sender anonymity, receiver anonymity, and unlinkability of sender and receiver [9]. Sender anonymity means that the identity of the party who sent a message is hidden, while its receiver (and the message itself) might not be. Receiver anonymity similarly means that the identity of the receiver is hidden. Unlinkability of sender and receiver means that though the sender and receiver can each be identified as participating in some communication, they cannot be identified as communicating with each other.

To gain efficiency, L. von Ahn, et al. concentrate on a weaker goal, k-anonymity [1]: the adversary is able to learn something about the origin or destination of a particular message, but cannot narrow down its search to a set of less than k participants. In other words, k-anonymity guarantees that in a network with n honest participants, the adversary is not able to guess the sender or recipient of a particular message with probability non-negligibly greater than $1/k$, where k is a constant smaller than, but otherwise not related to n. While k-anonymity is a weaker guarantee, it is still sufficient for a variety of applications. For example, if the participants in the peer-to-peer network were communicating k-anonymously, the music industry could not prosecute individuals in this manner. k-anonymity is also enough for the protection of privacy in everyday transactions, as it effectively breaks data profiling techniques.

In this paper, we present a new k-anonymous transmission protocol. The protocol is based on asymmetric encryption algorithm. All the members in the protocol is divided into smaller groups, and if all the members in the group perform the protocol correctly, the protocol is sender k-anonymous and receiver k-anonymous.

The organization of this paper is as follows: The rest of this section describe a few solutions to the anonymous communication problem. Section 2 presents the basic cryptographic notions we will need for the paper, and also introduces the definitions such as anonymous communication and k-anonymous communication. Section 3 presents the new k-anonymous communication protocol. And Section 4 gives the security analysis of our protocols. Finally, Section 5 concludes this paper with a discussion and some open questions.

1.1 Related Work

Till now, a lot of work in both research and practice has been done for anonymous message transmission (see [14, 15, 4–6, 8, 11]). Below we describe a few of the most influential solutions to the anonymous communication problem.

Proxies and Proxy Chaining [7]. The simplest way to shield the identity of the user is to forward all traffic from that client to a proxy server which submits all request on the user's behalf. For example, if Computer A wants to send a message to Computer B, Computer A sends a message to Computer P which in turn forwards the message to Computer B on A's behalf. This approach suffers from the same setbacks as any centralized system. If an entity can eavesdrop on Computer P the anonymity of all the computers which forward messages to P are compromised. The other concern is if P is trustworthy or not. The proxy method can be improved with the help of *proxy chaining*. Instead of P passing on the message from A to B it forwards to P_1 which in turn forwards the message to B. In this case P_1 does not know the origin of the message, and the identity of the A is more protected.

DC-Nets [3, 16]. DC-Nets, or Dining-Cryptographer networks, originally suggested by Chaum in [3], is an anonymous broadcast protocol that bases its anonymity on the strength of a secure multiparty sum computation. In the absence of trusted parties, this model provides unconditionally or cryptographically secure, depending on whether it is based on one-time-use keys or on public keys, respectively. The original system is improved by Waidner in [16], which allows to send and receive messages anonymously using an arbitrary communication network, and proved to be unconditionally secure. However, the poor scalability of DC-Nets makes it unsuitable for medium or large-scale use.

Mix-Nets [2]. Mix-Nets, introduced by Chaum in 1981, was one of the first concepts for anonymous communication. A model based on public key cryptography is presented that allows an electronic mail system to hide who a participant communicates with as well as the content of the communication. The technique does not require a universally trusted authority. One correspondent can remain anonymous to a second, while allowing the second to respond via an untraceable return address. The system includes a trusted computer called "Mix" to shuffle messages and route them, thus confusing traffic analysis. Chaining Mixes together to form a path, combined with Mix-to-Mix and end-to-end encryption, offers a form of provable security against a completely passive adversary.

Crowds [10]. Crowds is introduced by Reiter and Rubin in 1998 to protect users' anonymity on the world-wide-web. Similar to Mix-Nets, this system provides paths to disguise the originator of a message. Their approach works by grouping web users into a geographically diverse collection, called a *crowd*, which retrieves information on its users' behalf by way of a simple randomized routing protocol. Paths in Crowds are determined randomly by the machines through which a message passes, rather than by the originator of the message. Web servers are unable to learn the true source of a request because it is equally likely to have originated from any member of the crowd. Using degrees of anonymity, they have characterized the anonymity properties provided by the protocol against several classes of attackers.

CliqueNet [12]. CliqueNet, introduced by Sirer, et al., outlines a design for a peer-to-peer, scalable, tamper-resilient communication protocol that provides strong anonymity and privacy. The system combines small DC-Nets with a routing layer to mitigate the scalability problems of DC-Nets. The protocol provides an information-theoretic guarantee: an omnipotent adversary that can wiretap at any location in the network cannot determine the sender of a packet beyond a clique. CliqueNet has the undesirable feature, however, that an adversary who controls l network nodes can completely compromise the anonymity of $l-1$ other nodes of its choice. Furthermore, CliqueNet's routing layer induces a high amount of unnecessary network latency and is not secure against non-participation, allowing an adversary who controls a few nodes to partition the network.

k-Anonymous Message Transmission [1]. A k-Anonymous message transmission protocol is presented by Ahn, et al. in 2003. They introduce the notions of sender and receiver k-anonymity and consider their applications. They show that there exist simple and efficient protocols which are k-anonymous for both the sender and the receiver in a model where a polynomial time adversary can see all traffic in the network and can control up to a constant fraction of the participants. The presented protocol is provably secure, practical, and does not require the existence of trusted third parties.

Receiver Anonymity via Incomparable Public Keys [17]. A receiver anonymity via incomparable public keys is presented by Waters, et al in 2003. They describe a new method for protecting the anonymity of message receivers in an untrusted network. The method relies on the use of multicast, along with a novel cryptographic primitive that we call an Incomparable Public Key cryptosystem, which allows a receiver to efficiently create many anonymous "identities" for itself without divulging that these separate "identities" actually refer to the same receiver, and without increasing the receiver's workload as the number of identities increases.

2 Preliminaries

In this section, we present the basic cryptographic notions and definitions we will need for the paper.

2.1 Notation

Before we present the protocol, we describe the notations used in the protocol:

$\mathcal{M}$	the message space
n	number of members in the protocol
i, j	indices of members (range $[1, n]$)
m	number of groups in the protocol
g	indices of group (range $[1, n/k]$)
P_i	the i-th member
G_i	the i-th group
l_i	the length of message that P_i wants to transfer
e_P	the public key of member P
d_P	the private key of member P
k	the key for symmetric key algorithm

Denote the symmetric encryption and decryption with respect to a secret key k by $E_k(\cdot)$ and $D_k(\cdot)$, respectively, and denote the asymmetric encryption using a public key e and the corresponding asymmetric decryption using a private key d by $(\cdot)^e$ and $(\cdot)^d$, respectively. If a member in the network encrypts a random string and sends it to another member, we call this message a masked message.

2.2 Definition

We briefly review the concept of anonymous communication [1].

An anonymous communication protocol for message space $\mathcal{M}$ is a computation among n parties $P_1, \ldots, P_n$, where each P_i starts with a private input $(msg_i, p_i) \in (\mathcal{M} \times [n]) \cup \{(\text{nil}, \text{nil})\}$, and each party terminates with a private output from M^*. To communicate, time will be split into *rounds* and the protocol will be run at each round. Intuitively, at the end of a round each P_i should learn the set of messages addressed to him ($\{msg_j : p_j = i\}$), but not the identity of the senders.

We let $H \subset \{P_1, \ldots, P_n\}$ denote the set of honest parties. We denote by $\mathcal{P}(P_1(msg_1, p_1), \ldots, P_n(msg_n, p_n))$ the random variable distributed according to the adversary's *view* of the protocol $\mathcal{P}$ when each P_i has input (msg_i, p_i). We denote by $\mathcal{P}(P_i(msg_i, p_i), *)$ the adversary's view of $\mathcal{P}$ when P_i has input (msg_i, p_i) and the other inputs are set arbitrarily.

A protocol $\mathcal{P}$ is *sender anonymous* if for every pair $P_i, P_j \in H$, and every pair $(msg, p) \in (\mathcal{M} \times [n]) \cup \{(\text{nil}, \text{nil})\}$, $\mathcal{P}(P_i(msg, p), *)$ and $\mathcal{P}(P_j(msg, p), *)$ are computationally indistinguishable.

That is, a protocol is sender anonymous if the adversary may not distinguish between any of the honest parties as the sender of a message, regardless of who the receiver is; i.e., the adversary "gains no information" about the sender.

A protocol $\mathcal{P}$ is *receiver anonymous* if for every $P' \in H$, for every $msg \in \mathcal{M}$ and every $P_i, P_j \in H$, $\mathcal{P}(P'(msg, P_i), *)$ and $\mathcal{P}(P'(msg, P_j), *)$ are computationally indistinguishable.

According to the previous definitions, the trivial protocol in which no party transmits anything is both sender and receiver anonymous.

Assuming that the protocol is non-trivial (i.e., useful), sender anonymity requires every honest party, even if they have no message as an input, to send at least one *protocol* message per anonymous message delivered. Thus any protocol which is sender anonymous has a worst-case lower bound of n protocol messages per input message, since *in the worst case*, all parties but one have input (nil, nil). If n is large, this lower bound makes it unlikely that a system providing full anonymity can be fielded in practice.

A protocol $\mathcal{P}$ is *sender k-anonymous* if it induces a partition $\{V_1, \ldots, V_l\}$ of H such that:

1. $|Vs| \geq k$ for all $1 \leq s \leq l$;
2. For every $1 \leq s \leq l$, for all $P_i, P_j \in V_s$, for every $(msg, p) \in (\mathcal{M} \times [n]) \cup \{(\text{nil}, \text{nil})\}$, $\mathcal{P}(P_i(msg, p), *)$ and $\mathcal{P}(P_j(msg, p), *)$ are computationally indistinguishable.

That is, each honest party's messages are indistinguishable from those sent by at least $k-1$ other honest parties.

A protocol $\mathcal{P}$ is *receiver k-anonymous* if it induces a partition $\{V_1, \ldots, V_l\}$ of H such that:

1. $|V_s| \geq k$ for all $1 \leq s \leq l$;
2. For every $1 \leq s \leq l$, for all $P_i, P_j \in V_s$, for every $P' \in H$, $msg \in \mathcal{M}$: $\mathcal{P}(P'(msg, P_i), *)$ and $\mathcal{P}(P'(msg, P_j), *)$ are computationally indistinguishable.

That is, each message sent to an honest party has at least k indistinguishable recipients.

3 The Protocol

Our solution to the k-anonymous message transmission problem uses the asymmetric key algorithm. Suppose that all the member agree on a public key algorithm, say RSA, and a symmetric key algorithm, say the Advanced Encryption Standard (AES).

3.1 Protocol Preparation

Suppose that there are n members using the protocol to transfer message.

Before the k-anonymous message transmission protocol is performed, all the members need to get a long term public/private key pair which can be used for a number of purposes. In fact, this could be the usual public/private key pair that can be used for a variety of purposes such as electronic payment, email, and so on.

In order to band the public/private key pair to a legal member, we may use certificate system. Obtaining such public/private key of a certificate can be done via either a secure online channel or an off-line method such as physical access to a trusted certification authority. Let (e_P, d_P) be the public/private key pair of member P.

When generating the public/private key pair, a member needs to generate a pair of RSA public/private key $\{e, d\}$ and two large prime numbers; the product of the prime numbers forms a modulo n, where $de = 1 \mod \phi(n)$.

In order to increase the efficiency of the protocol, we partition the n members in the protocol into smaller groups of size $O(k)$. The following protocol may be performed by each group individually.

3.2 Collection Phase

In a group, suppose a member wants to transfer a message to another member in the network anonymously. Each member P_i in the group chooses g_i, index of the group that the receiver belongs to, and l_i, the length of the message he wants to transfer. If P_i do not want to send message, he may choose g_i greater than g.

1. Each member P_i chooses g_i and l_i, then he computes $x_i = (g_i||l_i)^{e_0}$ where e_0 is the public key of the group leader.
2. P_i randomly chooses $j \in [1, k]$ and sends x_i to P_j.
3. Received x_i, P_j sends it to the group leader.
4. Received x_i, the group leader computes $y_i = x_i^{d_0}$ to gets g_i and l_i, here d_0 is the private key of the group leader.
5. If $n/k \leq g_i$, the group leader discards this message; if $1 \leq g_i \leq n/k$, the group leader publish g_i. Let l denotes the biggest l_i among all the l_i satisfies $1 \leq g_i \leq n/k$. The group leader also publish l.

After the group leader publish l and all the indexes, each member P_i checks that whether the group index g_i is in the publish index set and whether the length l_i is smaller than l. If both conditions are satisfied, the member P_i broadcasts "yes" message, otherwise, he broadcasts "no" message.

If all the members in the group broadcast "yes" message, the protocol goes to the next step, otherwise, the protocol stops.

3.3 Transmission Phase

In the last step, each member in the group, say group A, knows g_i, the index of the group which a message will be transferred to, and l, the length of the message that can can be transferred. Now, suppose that a message will be transferred to a member in the group g_i, say group B, then all the member in the group may perform the following protocol.

1. P_i, the i-th member in group A, gets the public keys of all the members in group B. The P_i randomly chooses a secret key $k_{i,j}$, $1 \leq j \leq n_B$, where n_B is the number of members in the group B.
2. If P_i wants to transfer message $msg_{i,j}$ to Q_j, the j-th member in the group B, computes $k'_{i,j} = (k_{i,j})^{e_{B,j}}$ and $msg'_{i,j} = E_{k_{i,j}}(msg_{i,j})$, where $e_{B,j}$ is the public key of the j-th member in the group B. If the length of the message $meg_{i,j}$ is smaller than l, P_i may pad 0 at the end of the message.
3. If P_i does not want to transfer message to Q_j, he chooses a random string as the message $msg_{i,j}$. Then, he computes $k'_{i,j} = (k_{i,j})^{e_{B,j}}$ and $msg'_{i,j} = E_{k_{i,j}}(msg_{i,j})$.
4. P_i sends the message $k'_{i,1}||msg'_{i,1}||\ldots||k'_{i,n_B}||msg'_{i,n_B}$ to the group leader.
5. Receiving all the messages, the group leader construct a message M_j and send it to the j-th member of group B, where $1 \leq j \leq n_B$. Here,
$$M_j = k'_{i_1,j}||msg'_{i_1,j}||\ldots||k'_{i_{n_A},j}||msg'_{i_{n_A},j},$$
where $\{i_1, \ldots, i_{n_A}\}$ is a permutation of $\{1, \ldots, n_A\}$.

Receiving the message M_j, Q_j can obtain the $k_{i,j}$ by computing $(k'_{i,j})^{d_{B,j}}$, and $msg_{i,j}$ by $D_{k_{i,j}}(msg'_{i,j})$.

4 Analysis

The objective of our research is to protect users' anonymity on Internet. Here, we informally analyze two important issues: performance and security, for evaluating the proposed protocol.

Suppose that P_i, a member in group A, wants to transfer a message msg to Q_j, a member in group B.

4.1 Performance

In the last section, we introduce our k-anonymous transmission protocol. The security of our protocol is based on asymmetric encryption algorithm. All the members in the protocol needs to get a public/private key pair. When the members are divided into small groups, some members who trust each other may construct a group, or a member may be appointed to a group by the system.

In the collection phase, if a member in a group wants to send messages to several members, he may encrypt each index and send each encrypted message to different member in the group simultaneously. Then all the encrypted messages will be forwarded to the group leader and all the indexes will be published by the group leader. In the protocol, the group leader publish the length of the longest message wanted to transfer. In order to reduce the transmission bits, for each group index, the group leader may publish the length of the longest message that is transferred to that group.

In the transmission phase, a member may sends different message to different member in the group B. He just encrypts each message by the public key of corresponding member, then sends the messages with masked messages to the group leader. Reconstructed by the group leader, the messages will be sent ro the corresponding messages. Moreover, a member in the group B may receive several messages from the different members in the group A at one time.

4.2 Robustness

In collection phase, P_i encrypts the index of group B and sends it to another member in the group. That member forwards this message to the group leader. The group leader can obtain the index of the group B by decrypting the message, and then he can publish this index.

In transmission phase, P_i encrypts the message msg and sends it to the group leader with some masked messages. The group leader constructs a new message M_j including msg' and some other masked messages, and then sends M_j to Q_j. Receiving the message M_j, Q_j may obtain the message msg by decrypting the message M_j.

Consequently, if all the members in the group perform the protocol correctly, the protocol may transfer the message msg of P_i to the member Q_j.

4.3 Anonymity

Let us consider the anonymity in this subsection. First, we consider the sender's anonymity.

In our protocol, since both the messages transferred in the group A and the messages transferred between group A and group B are encrypted, and all the members in the group A send the message to the group leader, the adversary outside the group cannot decide which message is the transferred message and which message is just a encrypted random string. Thus, the adversary outside the group cannot know which member in the group sends the message.

The group leader cannot decide which member in the group sends a message. The reasons are as follows: In the collection phase, the group leader receives a forward message, he cannot know who is the original sender. In the transmission phase, the group leader receives encrypted messages from every member in the group, and he cannot decide which message is the transferred message.

In the collection phase, the group member forwards a message to the group leader, but he cannot know the message is encrypted from an index or just a random number. In the transmission phase, he cannot decide who transfer a message, too. Thus, the group member cannot decide which member in the group sends a message.

For the member is the group B, if he receives a real message, he only knows that the message is forwarded by the group leader or the group A, but he cannot decide which member in the group A sends the message.

To sum up, if all the members in the group perform the protocol correctly, the protocol is sender k-anonymous for all the group.

Second, we consider the receiver's anonymity.

In our protocol, since the messages transferred between group A and group B are encrypted, and all the members in the group B receives the message from the group leader of the group A, the adversary outside the group cannot decide which message is the transferred message and which message is just a encrypted random string. Thus, the adversary outside the group cannot know which member in the group B receives the message.

The group leader cannot decide which member in the group B receives a message. This is because the group leader receives encrypted messages from every member in the group, and he cannot decide which message he construct include the transferred message. Similarly, the group member cannot decide which member in the group B receives a message.

To sum up, if all the members in the group perform the protocol correctly, the protocol is receiver k-anonymous.

4.4 Privacy

In our protocol, all the transferred messages are encrypted. If a member P_i sends the message msg to the member Q_j, this message is encrypted by the secret key $k_{i,j}$. This secret key is encrypted by the public key of Q_j and sends to Q_j. Thus, the message msg can be read only by Q_j. The others cannot get the message msg since they do not know the private key of Q_j.

4.5 More Discussion on Receiver

In our protocol, we use certificate system to band the public/private key pair to a legal member. Thus, the sender knows who receives the messages. If we want to fully protect receiver anonymity, we may use the method presented by Water, et al in [17]. The holder of a secret key generate a large number of public encryption keys such that any message encrypted with any of these public keys can be decrypted by one secret key. All the members in a group publish their public keys in a list. The message transmitted to a group will be encrypted by one of the public keys in the group public key list. Then, the message will be received by one member in the group.

4.6 Efficiency

First, we evaluate the message complexity of our protocol. In the collection phase, each member in the group can send one or more messages to the other members, so the message generated by a member is $O(1)$. Thus, the total messages in this step is $O(k)$, since there is $O(k)$ members in a group. Then, all the messages are forwarded by a member in the group, thus, the total messages in this step is $O(k)$, too. Therefore, the total messages in the collection phase is $O(k)$. In the transmission phase, each member sends a message to the group leader. Then the group leader sends a constructed message to each member in the group B. Since both the group A and group B have $O(k)$ members, the total messages in the transmission phase is $O(k)$. From all above, the total message transferred in one round is $O(k)$.

Second, we evaluate the bit complexity of our protocol. In the collection phase, the total messages in the collection phase is $O(k)$. From the protocol, we see that each message in this phase is just to encrypt a number, so, if the length of the module in the RSA encryption system is l, the length of every message in this phase is l. Thus, the total transferred bits in the collection phase is $O(kl)$. In the transmission phase, each member sends a message to the group leader. From the protocol, we can see that each message can be divided into $O(k)$ parts, and the length of each part is $l + L$, where L is the biggest length of message to transfer. So, the total bits of each message is $O(k(l + L))$, and the total transferred bits in this step is $O(k^2(l + L))$, since there are $O(k)$ member in the group. Then the group leader sends a constructed message to each member in the group B. These messages are just a reconstruction of the messages sends to the group, so the total transferred bits in the transmission phase is $O(k^2(l + L))$, too. That is to say that the total transferred bits in this step is $O(k^2(l+L))$. From all above, the total bits transferred in one round is $O(k^2(l + L)) + O(kl) = O(k^2(l + L))$.

Third, we evaluate the computational complexity of our protocol. In the collection phase, each member sends one or more encrypted messages to the other members, and then the other member forwards the message, thus, in this phase, each member do $O(1)$ asymmetric encryption operation, and the total asymmetric encryption operation is $O(k)$. In the transmission phase, each member need to do $O(k)$ asymmetric encryption operation and $O(k)$ symmetric encryption operation. Thus, the total asymmetric encryption operation is $O(k^2)$, and the total symmetric encryption operation is $O(k^2)$, too. Receiving the message, Q_j, a member in group B, need to do $O(k)$ asymmetric decryption operation and $O(k)$ symmetric decryption operation. Thus, the total asymmetric decryption operation is $O(k^2)$, and the total symmetric decryption operation is $O(k^2)$, too.

5 Conclusion and Future Works

In this paper, we present a k-anonymous transmission protocol based on asymmetric encryption algorithm. All the members in the protocol is divided into smaller groups of size $O(k)$, and if all the members in the group perform the protocol correctly, the protocol is sender k-anonymous and receiver k-anonymous. The new features of this protocol are:

– Our protocol provides the privacy for the message. As long as the asymmetric encryption algorithm is secure, our protocol is secure, too.
– In our protocol, a member in a group may sends different message to different member in another group at one time.
– In our protocol, a member in a group may receive several messages from the different members in another group at one time.
– The total message transferred in one round is $O(k)$ and the total bits transferred in one round is $O(k^2(l + L)) + O(kl) = O(k^2(l + L))$.

In our protocol, if all the members in the group perform the protocol correctly, the protocol may transfer the message *msg* of a group member to a member in another group. In the future research, we will focus on how to detect the cheater in the group. Otherwise, in each group, there is a group leader served as a centralized host. In the future research, we are looking into techniques for distributing the data and signatories in a decentralized way (see [13, 18]).

Acknowledgements

This work is supported by the National Grand Fundamental Research 973 Program of China under Grant No. G1999035802; the Youth Foundation of the National Natural Science of China under Grant No. 60025205. The authors would like to thanks the anonymous referees for their helpful comments.

References

1. L. von Ahn, A. Bortz and N. J. Hopper. "k-Anonymous Message Transmission". *the 10th ACM Conference on Computer and Communication Security.* pages 122-130, 2003.
2. D. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". *Communications of the ACM.* 24(2), pages 84-88, 1981.
3. D. Chaum. "The Dining Cryptographers Problem : Unconditional Sender and Recipient Untraceability". *Journal of Cryptology.* 1(1), pages 65-75, 1988.
4. D. Denning. *Cryptography and Data Security.* Addison-Wesley. 1982.
5. D. Denning, P. Denning, and M. Schwartz. "The tracker: A threat to statistical database security". *ACM Trans. on Database Systems.* 4(1), pages 76-96, 1979.
6. G. Duncan and S. Mukherjee. "Microdata disclosure limitation in statistical databases: query size and random sample query control". *The 1991 IEEE Symposium on Research in Security and Privacy.* pages 278-287, 1991.

7. R. Duraikannu. “Anonymity in Web Transactions”. Unpublished manuscript. 2003. Available electronically at http://gaia.ecs.csus.edu/∼ghansahi/classes/projects/502/duraikannu/Ramkumar_Duraikannu_finalreport_11_21_03.doc
8. D. Mazières and M. F. Kaashoek. “The design, implementation and operation of an email pseudonym server”. *The 5th ACM Conference on Computer and Communications Security.* pages 27C36, 1998.
9. A. Pfitzmann and M. Waidner. “Networks without user observability — design options”. *Advances in Cryptology — Eurocrypt’85*, LNCS 219, Springer-Verlag, pages 245-253, 1986.
10. M. K. Reiter and A. D. Rubin. “Crowds: Anonymity for Web Transactions”. *ACM Transactions on Information and System Security.* 1(1), pages 66-92, 1998.
11. C. Shields and B. Levine. “A protocol for anonymous communication over the internet”. *The 7th ACM Conference on Computer and Communication Security.* pages 33-42, 2000.
12. E. G. Sirer, M. Polte and M. Robson. “CliqueNet: A Self-Organizing, Scalable, Peer-to-Peer Anonymous Communication Substrate”. Unpublished manuscript. 2001. Available electronically at http://www.cs.cornell.edu/People/egs/papers/cliquenet-iptp.pdf.
13. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. “Chord: A Peer-to-Peer Lookup Service for Internet Applications”. *ACM SIGCOMM Conference.* pages 149-160, 2001.
14. L. Sweeney. “k-Anonymity: a Model for Protecting Privacy”. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems.* 10(5), pages 557-570, 2002.
15. L. Sweeney. “Achieving k-anonymity privacy protection using generalization and suppression”. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), pages 571- 588, 2002.
16. M. Waidner. “Unconditional sender and recipient untraceability in spite of active attacks”. *Advances in Cryptology — Eurocrypt’89.* LNCS 434, Springer-Verlag, pages 302-319, 1989.
17. B. R. Waters, E. W. Felten and A. Sahai. “Receiver anonymity via incomparable public keys”. *The 10th ACM Conference on Computer and Communication Security.* pages 112-121, 2003.
18. L. Zhou, F. B. Schneider, and R. van Renesse. “COCA: A Secure Distributed Online Certification Authority”. Technical Report 2000-1828, Department of Computer Science, Cornell University, Ithaca, NY USA. December 2000.

Onions Based on Universal Re-encryption – Anonymous Communication Immune Against Repetitive Attack*

Marcin Gomułkiewicz, Marek Klonowski, and Mirosław Kutyłowski

Institute of Mathematics, Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27
50-370 Wrocław, Poland
Marcin.Gomulkiewicz@pwr.wroc.pl, klonowski@im.pwr.wroc.pl,
Miroslaw.Kutylowski@pwr.wroc.pl

Abstract. Encapsulating messages in onions is one of the major techniques providing anonymous communication in computer networks. To some extent, it provides security against traffic analysis by a passive adversary. However, it can be highly vulnerable to attacks by an active adversary. For instance, the adversary may perform a simple so–called *repetitive attack*: a malicious server sends the same massage twice, then the adversary traces places where the same message appears twice – revealing the route of the original message. A repetitive attack was examined for mix–networks. However, none of the countermeasures designed is suitable for onion–routing.

In this paper we propose an "onion-like" encoding design based on universal re-encryption. The onions constructed in this way can be used in a protocol that achieves the same goals as the classical onions, however, at the same time we achieve immunity against a repetitive attack. Even if an adversary disturbs communication and prevents processing a message somewhere on the onion path, it is easy to identify the malicious server performing the attack and provide an evidence of its illegal behavior.

Keywords: anonymous communication, unlinkability, onion, universal re-encryption, repetitive attack

1 Introduction

1.1 Anonymous Communication

Providing anonymous communication in public networks is a problem of growing importance. Demands for anonymity emerge both in the personal sphere and in e-commerce. In recent years a lot of papers about safe (or even provably anonymous) communication have appeared. Many protocols have been proposed – the most significant ones are DC-networks and MIX networks introduced by David Chaum [4, 3]. Later Rackoff and Simon proposed a fairly practical scheme providing anonymity based on an idea of Chaumian MIXes [14]. That was the first time when onions were explicitly

* Partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

used however, not under this name. This scheme was examined in details [13, 11] in different adversary models.

The idea of onions was used in a number of protocols, e.g. Babel [12] – a protocol aimed at anonymous email transfer, or Onion Routing protocol [7–9] – a protocol in which the connection between two peers is established via an anonymous path: servers on the path get information only on immediate predecessors and successors on the path. Recently, Fairbrother [6] has proposed a scheme based on onions for sending long messages (however, the scheme has turned out to be insecure). The onion mechanism is used also in TOR protocol (the second generation onion routing) [5] as one of the basic building blocks.

1.2 Active Attacks – Repetitive Attack

In many papers (i.e. [14]) it is assumed that an adversary can only eavesdrop the network and observe messages coming in and out of the servers, but cannot decode the packets, initiate new messages and/or destroy the old ones. While the assumption about an inability to read messages can be easily fulfilled using encryption techniques, preventing creation or deletion of messages is extremely difficult (for obvious reasons we cannot rely on the mechanisms such as PKI.)

If an adversary is given the possibility to send new messages of his choice, he can often compromise anonymity in the system. In order to trace the route of any given message it suffices to send it again, and search for double occurrences of identical messages, which shall ultimately reveal the identity of the final recipient. This attack is called a “repetitive attack”: it was proposed to compromise mix-networks ([3].) In that case the problem is well studied and resolved: to avoid such an attack several solutions have been suggested. Unfortunately, these methods might be inadequate for the protection of the onion communication protocols.

In Onion–Routing a repetitive attack is always effective when an adversary controls all links between servers and no precautions have been used.

1.3 New Results

In this paper we propose a new simple encoding scheme of “onions” immune against a repetitive attack and similar attacks leading to tracing messages. We call them URE-onions, since our solution is based on an extension of Universal–Re-Encryption by Golle, Jakobsson, Juels and Syverson described in [10]. By using this technique we are fairly able to limit the possibility of a repetitive attack – if a message is sent for the second time, it is re-encrypted at random at each point of the path. Therefore, the adversary cannot detect any repetition. Moreover, in the case when a server inserts faults into messages transmitted, it can be detected with an overwhelming probability and the evidence can be provided easily.

The new way of encoding the onions does not solve the problems that arise due to the traffic analysis of dynamic connections based on onions – these problems are a major issue for anonymous communication protocols – but it seems no encoding scheme can solve them.

2 Onions and Their Weaknesses

2.1 Onion Encoding

The goal of the onions is to protect communication so that the recipients and the sender cannot be linked by an adversary analyzing the network traffic. In the scheme we consider a network consisting of n servers. We assume that each server can communicate directly with other servers (like in P2P networks.)

Each server has a public and a private key, all public keys are widely accessible. The simplest version of the onion protocol looks as follows: in order to send a message m to server D, server S chooses intermediate servers at random, say $J_1, \ldots, J_\lambda$, and then encodes m as an *onion* (Enc_X means encryption with the public key of X):

$$\mathrm{Enc}_{J_1}(\mathrm{Enc}_{J_2}(\ldots(\mathrm{Enc}_{J_\lambda}(\mathrm{Enc}_D(m), D), J_\lambda)\ldots), J_3), J_2) .$$

This onion is sent by S to J_1. Node J_1 decrypts the message – the plaintext obtained consists of two parts: the second one is J_2, the first one is an onion with one layer peeled off:

$$\mathrm{Enc}_{J_2}(\ldots(\mathrm{Enc}_{J_\lambda}(\mathrm{Enc}_D(m), D), J_\lambda)\ldots), J_3) .$$

Then J_1 sends this onion to J_2. Nodes $J_2, \ldots, J_\lambda$ work similarly, the onion is "peeled off" until it finally arrives at D.

The general idea is that a server processing an onion (and an adversary tracing the traffic) cannot read the content of an onion, only the consecutive decoding with appropriate private keys gives each intermediate server sufficient information to route the message.

In fact, additional countermeasures must be taken to avoid some simple attacks (see for instance [3]):

– For an outgoing sub-onion O sent by server J_i, an adversary may attach "J_i" to O, encrypt the result with the public key of J_i, and compare the result with the messages received by J_i a step before to find out the source of O. One can prevent such an attack by attaching a random string at every layer of an onion or by using a probabilistic encryption scheme.
– The length of the onions should be fixed (otherwise the size could reveal the route of a message); some kind of padding can be used to cope with this problem.

The onion protocol acts similarly to a network of mixes: if two onions enter the same (honest) server simultaneously, an adversary cannot determine the relation between incoming and outgoing onions. Determining the number of rounds necessary for the protocol to ensure anonymity in this way is a challenging problem. Some discussion on the topic can be found in [14, 1, 11].

2.2 Adversary Model

The goal of an adversary might be a communication interruption and/or an unauthorized access to information. Our concern here is an anonymity breach, that is linking the senders and recipients of encoded messages in an anonymous communication protocol.

In many papers only passive adversaries are considered: they can observe (eavesdrop) some communication links and servers, but cannot interfere with the traffic in any way. Unfortunately, in a real world, this assumption is often too strong, e.g. in a typical Ethernet network sending packets without authorization is relatively very easy. Such "pirate" packets, geared at confusing legitimate parties of the protocol, may even appear to be sent according to the original protocol.

In our paper we assume that an adversary controls all links and some servers. This is quite a pessimistic model. Moreover, the adversary is "global," which means he always has the knowledge of all corrupted system components and can use them arbitrarily. In particular he has an access to all the private keys of controlled servers.

2.3 Repetitive Attack for Onions

The goal is to establish a connection between the sender and the recipient of a message. To carry out this attack an adversary sends a traced message twice from a controlled server and then observes the traffic on all links. When a certain message show up twice somewhere, it could be the message duplicated by the adversary (partially decoded according to the onion protocol). Then an adversary can immediately see the route of the message. An adversary can also send a copy of a message any time later and trace its route by comparing traffic in controlled links when the original message and its copy were sent.

2.4 Ways to Protect Against Repetitive Attack

In the case of onions, invariant elements are the onions that have to be sent further and the random strings (included in the onion to cope with the attack that was mentioned at the end of Section 2.1.) Unfortunately, we cannot remove the random strings or recode all layers of the onion simultaneously – at least for the classical onions. The reason is that such a recoding procedure should be performed without the knowledge of public keys, and certainly must be performed without the knowledge of private keys. Moreover, re-coding should be performed on internal parts of an onion, while on the other hand an intermediate server has no access to the internal parts of the onion processed. In fact, this is a fundamental feature of the encoding scheme.

For mixing networks there are some simple countermeasures against a repetitive attack. The first one requires the sender to prove his knowledge of what he is sending; obviously, since the layers are peeled off one by one, and we do not want to disclose their contents in any way, such a proof is not possible directly in the case of onions.

The second countermeasure is discarding any duplicate messages. Unfortunately, since an adversary may re-send a message at any later time, this would require from every server a lot of space for storing all traffic (or at least some fingerprints) that passes through it. There would also be a time overhead in processing the traffic – each single onion would be checked for re-occurrence. Perhaps the most worrying aspect is that recording traffic by servers would make eavesdropping much easier than before – coping the records could be much easier.

The second technique can be enhanced by appending to each package some information about the intervals of time when it should arrive at subsequent servers. In this solution packages “out of date” are simply discarded, so the servers can collect data from a short period of time only. This approach presented in [2] demands quite precise synchronization of time (otherwise one can mount an attack based on observing which messages get discarded).

MIX-networks work in “rounds.” In such systems we can propose to change keys or parameters in order to make a repetitive attack impossible. Unfortunately, it is not a practical solution for distributed communication systems which have to work continuously.

Handshake based solutions, like the one used in TOR protocol, have a disadvantage of a large latency and bidirectional communication.

3 Onions Based on Universal Re-encryption

3.1 Universal Re-encryption

Let us recall El-Gamal encryption scheme: p is an appropriate prime number (with a hard discrete logarithm problem), g is a generator of $\mathbb{Z}_p^*$, a random, nonzero $x < p-1$ is the private key, the corresponding public key is y, where $y = g^x \bmod p$. A message $m < p$ is encrypted in the following way. First a number k, $0 < k < p-1$, is chosen uniformly at random. Then we put $r := g^k \bmod p$ and $s = m \cdot y^k \bmod p$. The pair (s, r) is a ciphertext of m.

The El-Gamal cryptosystem has a very useful property: the same message encrypted for the second time yields a different ciphertext. Moreover, given two ciphertexts, it is impossible to say whether they were encrypted under the same key (unless, of course, the decryption key is given). This property is called *key-privacy* (see [10]). El-Gamal cryptosystem has yet another interesting feature. Everyone can re-encrypt a ciphertext (α, β) so that any relation between the old and the new ciphertext (α', β') is hidden for the observer not equipped with the decryption key. Namely, if y is the public key used for ciphertext creation, one can choose k' at random and set $\alpha' := \alpha \cdot y^{k'} \bmod p$, $\beta' := \beta \cdot g^{k'} \bmod p$. Obviously, (α', β') is a ciphertext of the same plaintext as before, but both its parts are “blinded” by random factors $y^{k'}$ and $g^{k'}$.

It is an astonishing feature that the above re-encryption trick can be modified slightly so that the public key does not need to be known ([10]): the inventors of the scheme, Golle, Jakobsson, Juels and Syverson, call it *universal re-encryption*, or *URE* for short. The scheme looks as follows:

- **Preliminaries.** A cyclic group G is chosen such that the discrete logarithm problem is computationally hard (e.g. $\mathbb{Z}_p^*$ for an appropriate prime number p). An arbitrary generator of G (say g) is chosen. Then G and g are published.
- **Key Setup.** Alice chooses a private key x at random; then the corresponding public key y is computed as $y = g^x$.
- **Encryption.** To encrypt message m for Alice, Bob generates numbers $0 < k_0, k_1 < |G|$ uniformly at random. Then, the ciphertext of m is computed as a quadruple:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1) := \left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}\right)$$

  In fact, this is an El-Gamal encryption made twice: the encrypted messages are m and 1, respectively.

– **Decryption.** Alice computes

$$m_0 := \frac{\alpha_0}{\beta_0^x} \quad \text{and} \quad m_1 := \frac{\alpha_1}{\beta_1^x},$$

and accepts message $m = m_0$, if and only if $m_1 = 1$.

– **Re-encryption.** Random values k'_0 and k'_1 are chosen. Re-encrypted message is described by the following formula:

$$\left(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1}\right).$$

During re-encryption all four components of a ciphertext change in a provably secure way (see [10]).

3.2 Extension of Universal Re-encryption

Let us assume that we have chosen a path of λ servers. We would like to encrypt a message so that it must be processed by the servers from the path and according to the order on the path. Simultaneously, we would like to retain the outstanding features of regular URE.

– **Key Setup.** Let x_i be the private key of the ith server ($1 \leq i \leq \lambda$). Let $y_i = g^{x_i}$ be the corresponding public key; y_i is published. Obviously each server determines his keys on its own and does not need to cooperate with others in this phase.

– **Encryption.** To encrypt a message m, two random values k_0 and k_1 are generated. The ciphertext has the following form:

$$\begin{aligned} E_{x_1,x_2,\ldots,x_\lambda}(m) &= (\alpha_0, \beta_0; \alpha_1, \beta_1) = \\ &= \left(m \cdot (y_1 y_2 \ldots y_\lambda)^{k_0}, g^{k_0}; (y_1 y_2 \ldots y_\lambda)^{k_1}, g^{k_1}\right) \end{aligned}$$

Hence,

$$E_{x_1,x_2,\ldots,x_\lambda}(m) = \left(m \cdot g^{k_0 \cdot \sum_{i=1}^{\lambda} x_i}, g^{k_0}; g^{k_1 \cdot \sum_{i=1}^{\lambda} x_i}, g^{k_1}\right).$$

So $E_{x_1,\ldots,x_\lambda}(m)$ is a ciphertext with decryption key $\sum_{i=1}^{\lambda} x_i$, and therefore it can be re-encrypted in the usual way. At any moment such a ciphertext can be partially decrypted. For instance, the first server can do it as follows:

$$E_{x_2,\ldots,x_\lambda}(m) = \left(\frac{\alpha_0}{\beta_0^{x_1}}, \beta_0; \frac{\alpha_1}{\beta_1^{x_1}}, \beta_1\right)$$

It is obvious that it is still a correct URE ciphertext with decryption key $\sum_{i=2}^{\lambda} x_i$, and therefore it can also be re-encrypted as it was mentioned above.

4 Modified Onion Protocol

In this section we show how to introduce non-determinism into processing of onions. For this purpose we modify the encoding used and propose so called *URE-onions*. We use a notation similar to those used in the previous section. Let x_i denote a secret key of server S_i; the corresponding public key $y_i = g^{x_i}$ is widely known. Also, g is a public parameter, it is a generator of a group such that finding discrete logarithms is computationally hard. Let $E_x(m)$ denote a ciphertext of a message m obtained with a public key corresponding to x according to schema of Golle *et al.*

As a first step, a random path of servers is chosen: $S_{i_1}, S_{i_2}, \ldots, S_{i_\lambda}$. Then a URE-onion consists of λ ciphertexts, called *blocks*. The jth block, for $1 \leq j \leq \lambda - 1$, has the form:

$$E_{x_{i_1}+\cdots+x_{i_j}}(\text{“send to } S_{i_{j+1}}\text{”})$$

The last block has the form

$$E_{x_{i_1}+\cdots+x_{i_\lambda}}(m)$$

The main difference between URE-onions and the classical onions is that we deviate from the original encapsulation idea: the messages for different routing steps are included in separate ciphertexts.

Another feature that differs this approach from the classical one is that we exclude any random contents from the intermediate messages. Obviously, random strings included in a message would betray duplication of messages and hence also some information on the route. The general rule is that auxiliary messages may contain only information that could be available for an adversary analyzing the traffic.

4.1 Routing

First, all blocks described above are sent together to server S_{i_1}. When a server S_j receives a URE-onion, it partially decrypts, re-encrypts, and changes the order of its blocks:

Partial Decryption Phase: each block $(\alpha_0, \beta_0; \alpha_1, \beta_1)$ is replaced by

$$D_{x_j}\left(E_{x_j}(m_i)\right) = \left(\frac{\alpha_0}{(\beta_0)^{x_j}}, \beta_0; \frac{\alpha_1}{(\beta_1)^{x_j}}, \beta_1\right) .$$

Re-encryption Phase: now S_j re-encrypts each block. So in place of the original block $(\alpha_0, \beta_0; \alpha_1, \beta_1)$ we obtain for some randomly chosen k_1, k_2:

$$\left(\frac{\alpha_0}{(\beta_0)^{x_j}} \cdot \left(\frac{\alpha_1}{(\beta_1)^{x_j}}\right)^{k_1}, \beta_0 \cdot (\beta_1)^{k_1}; \left(\frac{\alpha_1}{(\beta_1)^{x_j}}\right)^{k_2}, (\beta_1)^{k_2}\right) .$$

Permutating Phase: All blocks are permuted at random.

After the decryption phase, exactly one block should contain the next destination S, unless the URE-onion has reached its target. It is easy to notice that the length of the URE-onion remains fixed and the server processing a URE-onion cannot say how many hops remain. Also, re-encryption guarantees that the address of S, the next server on the route, remains hidden for all servers except S_j.

At the next step the URE-onion is sent to the destination S retrieved from a block at a partial decryption phase.

4.2 Immunity Against Repetitive Attack

Let us argue shortly why a repetitive attack does not work for the proposed protocol. Assume that there is at least one honest server "on the path" between two malicious servers controlled by an adversary. Then the adversary cannot detect the repetition of an onion, since the honest server re-encrypts each onion processed at random.

We have also eliminated all the information available for intermediate servers except the next destination. Sending a URE-onion multiple times would only increase the number of URE-onions with message "send to S_i," and so an adversary can only hope to provide additional data to the traffic analysis, which is not our concern here. Note that a URE-onion provides no more data for the traffic analysis than the regular onions.

4.3 Attempts to Change the Route

Since the blocks on a URE-onion are given, a malicious server can reorder them, eliminate some of them, or inject its own blocks:

Reordering: Since at each phase (except the last one) there is exactly one block that represents a valid server name, the order of the blocks is irrelevant. This has no effect on the protocol security.

Inserting Own Blocks: certainly, one can inject a number of blocks encoding initial servers on a path. This is possible, since encoding is based on public keys only. Then the URE-onion will be routed through such a "detour". The problem is that if at least one server on this detour is honest and performs partial decryption then the original blocks will be partially decrypted unnecessarily. Consequently, the blocks of the original URE-onion become unreadable. If the additional blocks are inserted somewhere in the middle then the processing will go on until the first inserted block is encountered. Then the inserted block will be unreadable as well due to partial decryptions that have occurred in between.

Removing a Block: if a block is removed then at some point some server, say S_{i_j} will not find the ciphertext

$$E_{x_{i_j}}(\text{“send to } S_{i_{j+1}}\text{”})$$

in the delivered blocks. Certainly, the server S_{i_j} cannot find the next server on the path so it must stop processing this URE-onion. Potentially an adversary can also remove some blocks and then insert new ones. We address this problem in the next subsections.

Modification of a Block: in fact, it is possible to change the contents of a block without decrypting it. We also discuss this problem in the next subsection.

4.4 Multiplicative Attack

An adversary can carry out an a bit more sophisticated attack than a repetitive attack. We call it a "multiplicative attack."

Let a URE-onion contain blocks $E_{k_i}(m_i)$ for $i \leq \lambda$. Each of them, except a single one, is dedicated to a particular server and keeps information about its successor on the path. Let

$$E_{k_i}(m_i) = (\alpha_0, \beta_0; \alpha_1, \beta_1) = (m_i \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})$$

A malicious server processing the URE-onion can choose some block blindly, say the block $E_{k_i}(m_i) = (\alpha_0, \beta_0; \alpha_1, \beta_1)$, and replace α_0 by $\alpha_0 \cdot \gamma$ for an arbitrary γ. In such a situation one of the further servers on the path obtains $\gamma \cdot m_i$ instead of m_i. If this server is also under the adversary's control, it knows the value γ so it can easily recover the value m_i and can carry on the protocol. If this server is not under control of the adversary, then m_i remains scrambled and the server finds that the address decoded is faulty.

During the attack described an adversary can manage to get some information about an onion path. However, if the first of the attacking servers misses an opportunity to disturb a proper block, the URE-onion will not be delivered to the final destination. So the multiplicative attack is less efficient than the repetitive attack for the regular onion protocol, but it is still unacceptable. For this reason we propose an "investigation" subprotocol to defend the scheme against the mentioned attack.

4.5 Investigation – Finding Out Dishonest Servers

If an honest server obtains an invalid URE-onion (i.e. none of blocks or more than one decrypted block represent the name of the next server on the route or a valid message) it can complain about the previous server from the path. In such a situation both servers – the previous server as well as the complaining one – must prove that they have behaved correctly, otherwise one of them is recognized guilty. If they manage to prove their compliance with the protocol, the next predecessor on the path is interrogated. The procedure is repeated until a cheater is detected. The main goal is to build an appropriate procedure for verifying a server. We assume that each server knows from whom it gets each packet and that it can prove it to other servers. The evidence might come for instance from the signed hash values of the traffic transmitted.

Let us consider a single server S_j from the path. It has a private key x_j such that $y_j = g^{x_j}$. Each block $(\alpha_0, \beta_0; \alpha_1, \beta_1)$ of the URE-onion should be processed by S_j in two phases – a partial decryption phase and a re-encryption phase. Assume also that S_j is asked to prove its honest behaviour. It must show that the URE-onion obtained from $(\alpha_0, \beta_0; \alpha_1, \beta_1)$ through the partial decryption and re-encryption is correctly built i.e. it has the form:

$$(\widehat{\alpha_0}, \widehat{\beta_0}; \widehat{\alpha_1}, \widehat{\beta_1}) = \left(\frac{\alpha_0}{(\beta_0)^{x_j}} \left(\frac{\alpha_1}{(\beta_1)^{x_j}} \right)^{k_1}, \beta_0 (\beta_1)^{k_1}; \left(\frac{\alpha_1}{(\beta_1)^{x_j}} \right)^{k_2}, (\beta_1)^{k_2} \right)$$

for some randomly chosen k_1, k_2. For verification, the numbers k_1 and k_2 are revealed, as well as

$$(\alpha_0', \beta_0; \alpha_1', \beta_1) = \left(\frac{\alpha_0}{(\beta_0)^{x_j}}, \beta_0; \frac{\alpha_1}{(\beta_1)^{x_j}}, \beta_1 \right) .$$

but of course x_j must remain secret. The re-encryption phase can be checked in a straightforward way. For examining partial decryption we use a zero-knowledge protocol for showing the equality of discrete logarithms[15]: recall that the aim of a protocol called EQDL$(A, B, C; a, b, c)$ is to prove that there is a number x such that $A = a^x, B = b^x, C = c^x$. So the server proving its behaviour presents a proof

$$\mathrm{EQDL}(\alpha_0/\alpha'_0, \alpha_1/\alpha'_1, y_j; \beta_0, \beta_1, g) .$$

Since $y_j = g^{x_j}$ the proof should convince that $\alpha_0/\alpha'_0 = \beta_0^{x_j}$ and $\alpha_1/\alpha'_1 = \beta_1^{x_j}$ which was our goal.

Let us note that the EQDL proof scheme is not really interactive, so it is better suited for showing honesty afterwards.

A server is immediately rejected from the protocol if it fails to prove its correct behaviour. The only drawback of this method is the necessity of storing all random parameters that have been used by re-encryption (or ability to reconstruct them from a random seed). Fortunately, the time of storing them can be limited to some time bound within which an onion normally reaches its target.

5 Concluding Remarks

Thanks to Universal Re-Encryption scheme URE-onions are immune to a repetitive attack. Any attempt of a multiplicative attack can be detected with a probability proportional to the ratio of honest servers in the whole network. Also changing a block leads to detection of a dishonest server with a significant probability so active tracing of an URE-onion becomes a very risky business. Moreover, URE-onions do not require additional interaction except for the case of "cheating investigation," and even in this case the interaction is kept minimal. Of course the described scheme does not automatically ensure security against all theoretically possible active attacks.

The new onions might be more expensive than the original ones regarding processing time: each server must perform λ decryptions of λ blocks instead of one decryption of a large block, as it happens for the regular onion protocol.

A serious disadvantage is that an URE-onion cannot be combined with symmetric encryption in the same way as it can be done for regular onions. So it might be suited for small (e.g. control) messages only. On the other hand, URE-onions offer much more flexibility that can be used for diverse purposes.

References

1. Berman, R., Fiat, A., Ta-Shma, A.: *Provable Unlinkability Against Traffic Analysis*, Financial Cryptography'2004, Lecture Notes in Computer Science 3110, Springer-Verlag
2. Büschkes, R., Egner, J., Kesdogan, D.: *Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System*, Information Hiding Workshop '1998 Lecture Notes in Computer Science 1525, Springer-Verlag, pp. 83-98
3. Chaum, D.: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of ACM 24(2) (1981) pp. 84-88

4. Chaum, D.: *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*, Journal of Cryptology 1.1 (1988), 65-75
5. Dingledine, R., Mathewson, N., Syverson, P.: *Tor: the Second Generation Onion Router*, USENIX Security, 2004
6. Fairbrother, P.: *An Improved Construction for Universal Re-encryption*, Privacy Enhancing Technologies '2004, Lecture Notes in Computer Science , Springer-Verlag.
7. Goldschlag, D. M., Reed, M. G., Syverson, P. F.: *Hiding Routing Information*, Information Hiding Workshop '1996, Lecture Notes in Computer Science 1174, Springer-Verlag, 137-150
8. Goldschlag, D. M., Reed, M. G., Syverson, P. F.: *Private Web Browsing*, Journal of Computer Security, Special Issue on Web Security 5 (1997), 237-248
9. Goldschlag, D. M., Reed, M. G., Syverson, P. F.: *Anonymous Connections and Onion Routing*, IEEE Journal on Selected Areas in Communication, 1998, 16(4):482-494
10. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: *Universal Re-encryption for Mixnets*, RSA Conference, Cryptographers' Track, 2004, 163-178
11. Gomułkiewicz, M., Klonowski, M., Kutyłowski, M.: *Provable Unlinkability Against Traffic Analysis Already After $\mathcal{O}(\log(n))$ Steps!*, 7th Information Security Conference (ISC'2004), Lecture Notes in Computer Science , Springer-Verlag
12. Gülcü, C., Tsudik, G.: *Mixing E-mail with BABEL*, ISOC Symposium on Network and Distributed System Security, IEEE 1996, 2-16
13. Jakobsson, M., Juels, A.: *An optimally robust hybrid mix network*, 20 ACM Symposium on Principles of Distributed Computing 2001, 284-292
14. Rackoff, C., Simon, D. R.: *Cryptographic Defense Against Traffic Analysis*, 25 ACM Symposium on Theory of Computing (1993), pp. 672-681
15. Schnorr, C.P.: *Efficient signature generation by smart cards*, Journal of Cryptology 4, 1991: 161-174

Side Channel Cryptanalysis on SEED[⋆]

HyungSo Yoo[1], ChangKyun Kim[1], JaeCheol Ha[2,⋆⋆],
SangJae Moon[1], and IlHwan Park[3]

[1] School of Electrical Engineering and Computer Science,
Kyungpook National Univ., Daegu, 702-701, Korea
{hsyoo,dreamps}@m80.knu.ac.kr, sjmoon@knu.ac.kr

[2] Division of Information Science, Korea Nazarene Univ.,
Cheonan, Choongnam, 330-718, Korea
jcha@kornu.ac.kr

[3] National Security Research Institute, Daejeon, Korea
ilhpark@etri.re.kr

Abstract. The Korea standard block cipher, SEED, is a 128-bit symmetric block cipher with a more complex F function than DES. This paper shows that SEED is vulnerable to two types of side channel attacks, a fault analysis attack and a power analysis attack. The first one is a fault insertion analysis which induces permanent faults on the whole left register of 15-round. This attack allows one to obtain the secret key by using only two faulty cipher texts for encryption and decryption processing respectively. The second attack is a more realistic differential power analysis. This attack requires about 1000 power traces to find the full secret key. The above two attacks use a reverse property of the F function to obtain secret key, where the reverse property is derived from the our research.

Keywords: Side channel attack, Fault insertion analysis, Differential power analysis, block cipher, SEED.

1 Introduction

In September 1996, Boneh *et al.* announced a new cryptanalytic attack which could affect security of cryptographic modules [6]. They succeeded in breaking the RSA with CRT by using one correct signature and one faulty one. In this attack, hardware faults and errors which occur during the operations of a cryptographic device might leak information about the private key. Lenstra *et al.* improved this attack by finding two secret prime number using only one faulty signature of a message [15, 19]. Consequently, many papers have been published concerning the resistance of RSA cryptosystems with CRT to fault attacks [2,

[⋆] This research has been supported by University IT Research Center Project.

[⋆⋆] The third author was also supported in part by Korea Nazarene University research fund.

26, 27]. Also the fault attack by optical illumination [23] or by spike generator [2] has been reported and is much more feasible and potential for breaking cryptosystems.

In addition, Biham and Shamir have published a paper [4] detailing a fault analysis attack which is applicable to secret key cryptosystems such as Data Encryption Standard (DES). Assuming the same register faults that Boneh *et al.* considered [6], they showed that DES could be broken. They combined two attack techniques differential cryptanalysis [3] and fault analysis. Dusart *et al.* showed how DFA (Differential Fault Analysis) works on Advanced Encryption Standard (AES) [9]. They implemented this attack on a PC, then found the full AES-128 key by analyzing less than 50 cipher texts. In 2002, Giraud also presented a paper describing two types of DFA attacks on AES [11] using between 50 and 250 faulty cipher texts.

On the other hand, Kocher *et al.* [17] firstly introduced power attacks including the simple and differential power analyses (referred to as SPA and DPA, respectively). In SPA, a single power consumption of a cryptographic execution is measured and a trace is analyzed to classify operations which are related to secret information. In DPA, an adversary measures hundreds of power signal traces, divides them into two groups using a classification criterion, and makes a differential computation between the two averaged values. Since the averaging and subtraction of two signal groups results in the amplification of small power differences which occur during the execution of an algorithm. In general, DPA is more powerful than SPA [7, 12, 25]. In fact, it has been reported that secret key cryptosystems (AES and DES) as well as public key cryptosystems (RSA and ECC) are vulnerable to DPA [5, 17, 20, 21].

In this paper, we show that the SEED which is a national industrial association standard algorithm in Korea (TTAS.KO-12.0004, 1999) and an international standard candidate for ISO/IEC SC27 CD 18033-3 [14, 18] is vulnerable to both fault attack and power attack. This paper is mostly divided in two parts. In the first part, the SEED is vulnerable to a fault insertion analysis which induces permanent faults on a whole left register of 15-round. This attack allows us to obtain the secret key by using only two faulty cipher texts for encryption and decryption processing respectively. The first reason having vulnerable property is that F function of SEED is recoverable, that is, if input and output are known, then adversary can find round key. The second reason is that if 1-round and 16-round key are known, then he can completely find full secret key. In the second part, the SEED is also vulnerable to a DPA. By using this attack, we can get a output value of a F function of 1-round during a decryption processing. The rest of this attack is similar to fault attack. This paper shows our experimental results of DPA on SEED.

This paper is organized as follows. In section 2, we give a brief overview of SEED. In section 3, we present our fault analysis, including our assumptions and the theory behind the attack. Section 4 shows the DPA method on SEED and our experimental results. Finally in section 5 we briefly discuss the countermeasure against the two types of attack on SEED and make a conclusion.

2 SEED Algorithm

The Korea standard block cipher, SEED is a secret key block cipher with a 128-bit data block and a 128-bit secret key. This algorithm has a Feistel structure with 16 rounds and a 128-bit input-output data block. The following notations are used throughout this paper.

$\boxplus$: addition in modular 2^{32}
$\boxminus$: subtraction in modular 2^{32}
$\oplus$: bitwise exclusive OR(XOR)
& : bitwise AND
$<< n$: left circular rotation by n bits
$>> n$: right circular rotation by n bits
$\|$: concatenation

2.1 Structure of SEED

A input plain text of a 128-bits is divided into two 64-bit blocks. The right 64-bit block R_0 is an input to the F function with a first 64-bit round subkey which is generated from the round key generation processing. The output of F function is XORed with the left 64-bit block L_0. After 16 round encryption processings, the final 128-bit output is a cipher text. The overview of SEED structure is shown in Figure 1.

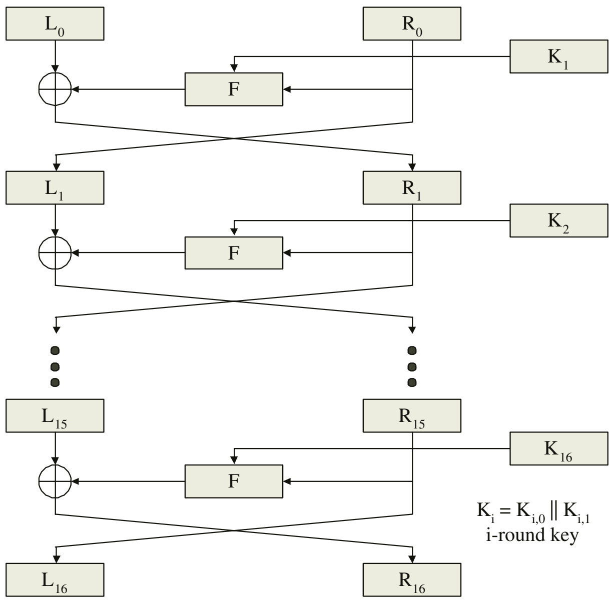


Fig. 1. Structure of SEED.

The F function also has a 64-bit Feistel structure. The input block is divided into two blocks (C, D) and XORed with two 32-bit subkeys ($K_{i,0}, K_{i,1}$). After mixing subkeys, two blocks are passed through three layers of G function with addition in modular 2^{32}. The structure of F function is shown in Figure 2.

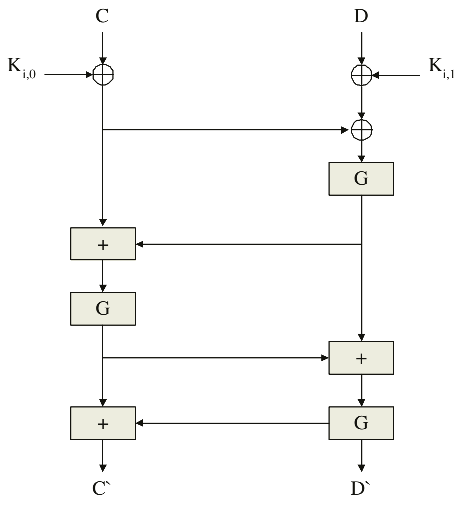


Fig. 2. The F function.

As shown in Figure 3, the G function used in F function has two security layers. The first layer consists of two S-boxes generated from boolean functions X^{247} and X^{251}. Here, S boxes can be represented by two lookup tables. The 32-bit input of G function is divided into 4 blocks. Each 8-bit block pass through the 8×8 S-boxes, S_2 and S_1. The second layer consists of permutation processing of S-box outputs which is a computation by AND operation with four specific values from m_0 to m_3. After XOR processing of expanded 16 blocks, G function generates a final 32-bit output.

$$\begin{aligned}
&Y_3 = S_2(X_3), Y_2 = S_1(X_2), Y_1 = S_2(X_1), Y_0 = S_1(X_0),\\
&Z_3 = (Y_0 \& m_3) \oplus (Y_1 \& m_0) \oplus (Y_2 \& m_1) \oplus (Y_3 \& m_2),\\
&Z_2 = (Y_0 \& m_2) \oplus (Y_1 \& m_3) \oplus (Y_2 \& m_0) \oplus (Y_3 \& m_1),\\
&Z_1 = (Y_0 \& m_1) \oplus (Y_1 \& m_2) \oplus (Y_2 \& m_3) \oplus (Y_3 \& m_0),\\
&Z_0 = (Y_0 \& m_0) \oplus (Y_1 \& m_1) \oplus (Y_2 \& m_2) \oplus (Y_3 \& m_3).
\end{aligned}$$

2.2 Round Key Generation

The round key generation function uses the G function, addition in modular 2^{32}, subtraction in modular 2^{32}, and left (right) circular rotation by 8 bits.

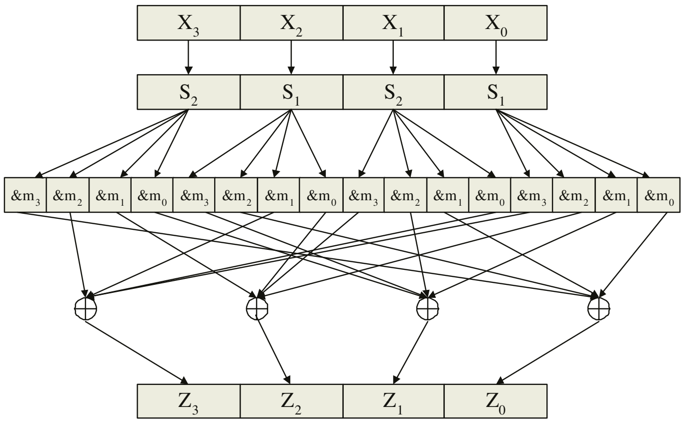


Fig. 3. The G function.

A 128-bit input key is separated into four 32-bit blocks (A, B, C, D). The two divided blocks perform addition (or subtraction) in modular 2^{32} and subtraction (addition) by the constant KC_1. The 1-round keys $K_{1,0}$ and $K_{1,1}$ are generated using final G function operations. After an 8-bit right rotation of the left 64-bits the second round keys $K_{2,0}$ and $K_{2,1}$ are generated by addition, subtraction by the constant KC_2, and G functions. The rest of the subkeys are generated in the same way, as shown in Figure 4.

3 Fault Analysis Attack on SEED

On a smart card, a fault may be induced in many ways, such as by a power glitch, by a clock pulse, or by radiation from a laser, etc. In this paper, the fault attack assumes that the permanent fault occurs on whole bit of a register. This attack is similar to one of two assumptions made by Biham and Shamir in fault attack for DES [4]. The other fault attack assumption is difficult to apply to our fault attack and is difficult to implement experimentally. One criticism against this second model, a differential fault analysis attack, is that the transient fault attack assumed by Biham and Shamir [4] is not realistic. Therefore, we assume a more practical fault model that will be less controversial. Our fault attack assumes that we can cut a wire or destroy a memory cell in a cryptoprocessor such as a smart card. As a result, the values in affected location can be considered to be permanently fixed. We assume that an adversary can insert these permanent faulty values into some memory cells. Some papers make similar assumptions in their work [1, 4, 22].

3.1 Fault Analysis Attack

We assume that SEED is implemented in hardware as 16 unrolled hardware rounds. For this attack, it suffices to destroy all the bits of the LSB of register L_{15}

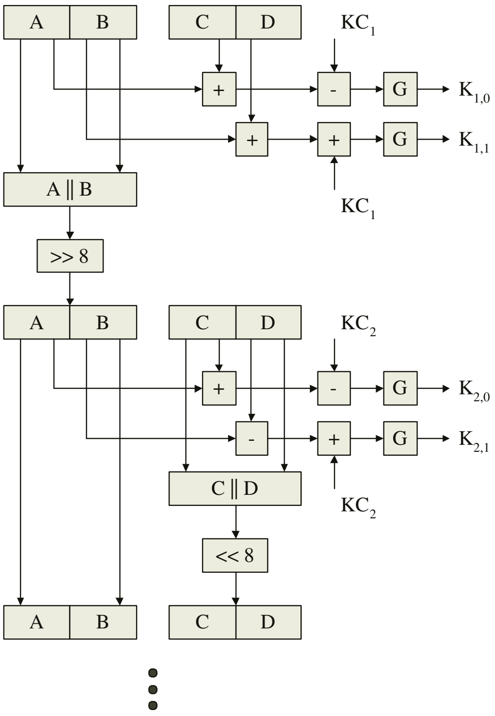


Fig. 4. Round key generation.

or set them to known values. This attack assumes that we can destroy a memory cell in a register or cut a wire. As a result, we consider the memory cells to be fixed to known values. This attack is a pure cipher text only attack, which does not require any information about plain texts. Based upon this assumption, we can find the 16-round keys, $K_{16,0}$ and $K_{16,1}$. The 16-round in the implementation of SEED is shown in Figure 5.

Our attack to find out the 16-round subkeys is composed of 4 steps as follows.

Step 1

We induce the faults to destroy all the bits of the LSB register L_{15} or set them to known values. In addition, we can see the final output of SEED, L_{16} and R_{16}. Therefore, we can find the input and outputs of the F function. In the end, we want to find the 16-round keys $K_{16,0}$ and $K_{16,1}$ when we know the input and output of the F function.

Step 2

Given that inputs and outputs in F function of Figure 2 (C, D, C', and D') are known, we want to find the keys $K_{16,0}$ and $K_{16,1}$. As for the addition in modular 2^{32}, the inverse operation is subtraction in the same modular. So, based on the assumption that the output of G function is known, if we can find the input of it, we safely say that we can extract the 16-round keys.

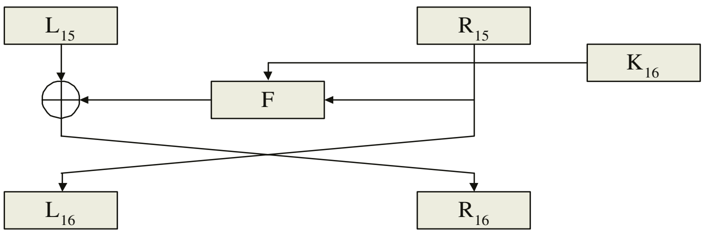


Fig. 5. 16-round of SEED.

Step 3

Let us attempt to find the inputs of G function given the outputs. In Figure 3, we denote the 32-bit inputs as X_0, X_1, X_2 and X_3, and the outputs as Z_0, Z_1, Z_2 and Z_3. It is enough to know the inverse values of function S by S-box tables S_1 and S_2. Consequently, the outputs of S-boxes are the result of $S_2(X_3)||S_1(X_2)||S_2(X_1)||S_1(X_0)$. An important point to emphasize is that the least significant bits Z_{00}, Z_{10} , Z_{20} and Z_{30} are determined by 4 bit inputs $S_{20}(X_3)||S_{10}(X_2)||S_{20}(X_1)||S_{10}(X_0)$. Here, Z_{ij} is a jth bit of Z_i. By selecting 8 out of 16 inputs combined by $S_{20}(X_3),\ S_{10}(X_2),\ S_{20}(X_1),$ and $S_{10}(X_0)$ we can calculate the correct output bit Z_{00}. Similarly, by selecting 4 out of 8 inputs to check bit Z_{00}, we can calculate the correct output bit Z_{10} and by selecting 2 out of 4 inputs to check Z_{10} bit, we can calculate the correct output bit Z_{20}. Finally, by selecting 1 out of 2 inputs to check bit Z_{20}, we can calculate the correct output bit Z_{30}. As a result, if we know the outputs of the G function, then we can compute its inputs, that is, the G function becomes reversible; there is a reverse computational method to find the inputs for given outputs. We will use the notation G^{-1} function to represent this inverse computational algorithm for G function.

Step 4

We know the inverse method for a G function and the addition function in modular 2^{32}. Therefore, the reverse computation for a F function shown in Figure 2 is possible. Finally, given the input and output of F function, we can find the 16-round keys, $K_{16,0}$ and $K_{16,1}$.

3.2 Secret Key Attack Using Two Round Keys

After fault insertion at 16-round for encryption of plain text, we can calculate 16-round keys by using the final cipher text. Additionally, after fault insertion at 16-round for decryption of the cipher text, we can compute 1-round keys using the final plain text. Note that it is not necessary to use a genuine cipher text as an input. It is sufficient to use random data as an input because an adversary is only interested in the decryption operation.

Now, let us examine how to search the 128-bit secret key given a 1-round and a 16-round keys. As you see in Figure 4, a 128-bit input key is divided into four 32-bit blocks, and then used to generate a 64-bit round key at each round. The round key generation algorithm uses four computational operations: circular rotation by 8 bits, addition (subtraction) in modular 2^{32} and G function. As has been noted, we can compute inverse values of function G and addition (subtraction) in modular 2^{32}. Therefore, if we know the 1-round keys $K_{1,0}$ and $K_{1,1}$ in Figure 4, then we know two temporary values $(A + C)$ and $(B - D)$ as follows.

$$\begin{aligned} A + C = G^{-1}(K_{1,0}) + KC_1 &= T_{1,0} \\ B - D = G^{-1}(K_{1,1}) - KC_1 &= T_{1,1} \\ A_3||A_2||A_1||A_0 + C_3||C_2||C_1||C_0 &= T_{1,0} \\ B_3||B_2||B_1||B_0 - D_3||D_2||D_1||D_0 &= T_{1,1} \end{aligned}$$

Note that the A and B used in 16-round key generation are the same that were used in 1-round due to 8 right circular rotations by 8 bits. Furthermore, C and D used in the 16-round key generation are the same that operated by only one right circular rotations by 8 bits for 1-round C and D.

$$\begin{aligned} A + L((C||D) >> 8) = G^{-1}(K_{16,0}) + KC_{16} &= T_{16,0} \\ B - R((C||D) >> 8) = G^{-1}(K_{16,1}) + KC_{16} &= T_{16,1} \\ A_3||A_2||A_1||A_0 + D_0||C_3||C_2||C_1 &= T_{16,0} \\ B_3||B_2||B_1||B_0 - C_0||D_3||D_2||D_1 &= T_{16,1} \end{aligned}$$

Here, $L(\ )$ means the extraction of the left 32 bits from 64 bits data and $R(\ )$ means right extraction. In order to compute the secret key, we subtract two equations as follows.

$$\begin{aligned} T_{16,1} - T_{1,1} &= D_3||D_2||D_1||D_0 - C_0||D_3||D_2||D_1 \\ T_{1,0} - T_{16,0} &= C_3||C_2||C_1||C_0 - D_0||C_3||C_2||C_1 \end{aligned}$$

As you know, if C and D are known, then A and B can easily be computed. Figure 6 describes the operation to subtract two temporary round keys. As shown in Figure 6, knowing $T_{16,1} - T_{1,1}$ and $T_{1,0} - T_{16,0}$, we can compute 256 possible secret keys. Now, let D_0 be a random 8 bit value. Then we can compute D_1 as $D_0 - R_{7-0}(T_{16,1} - T_{1,1})$ where $R_{7-0}(K)$ denotes the bits from the LSB to the 7th bit of K. Consecutively, we can compute D_2 from $D_1 - R_{15-8}(T_{16,1} - T_{1,1})$ as follows.

$$\begin{aligned} D_1 &= D_0 - R_{7-0}(T_{16,1} - T_{1,1}) \\ D_2 &= D_1 - R_{15-8}(T_{16,1} - T_{1,1}) \\ D_3 &= D_2 - R_{23-16}(T_{16,1} - T_{1,1}) \\ C_0 &= D_3 - R_{31-24}(T_{16,1} - T_{1,1}) \\ C_1 &= C_0 - R_{7-0}(T_{1,0} - T_{16,0}) \\ C_2 &= C_1 - R_{15-8}(T_{1,0} - T_{16,0}) \\ C_3 &= C_2 - R_{23-16}(T_{1,0} - T_{16,0}) \\ D_0 &= C_3 - R_{31-24}(T_{1,0} - T_{16,0}) \end{aligned}$$

Note that D_0 used to compute $D_1 = (D_0 - R_{7-0}(T_{16,1} - T_{1,1}))$ is same with the final value $C_3 - R_{31-24}(T_{1,0} - T_{16,0})$. Finally, we can compute 256 C and D from 256 D_0. If 256 C and D are known, then 256 A and B can simply be computed. These 256 secret keys can always generate secret keys which satisfy the relationship between 1-round and 16-round keys. The final step to find a complete secret key is to do an exhaustive search of the 256 possible secret keys. Known plain text/cipher text pair (using the fault-inserted cipher text), a computer can find the unique secret key satisfied the text-pair from the 256 possible secret keys by an exhaustive software search. Thus, we can find the full secret key using only two faulty cipher texts, where one is the output for encryption processing, and the other is for decryption processing.

We assume that SEED is implemented as a single round, which is used 16 times. This is an iterated hardware implementation of SEED. In this case we generate a permanent fault in the left-half register, in which all of the bits are permanently fixed. At this point, it doesn't matter whether the value of the left-half register is zero or known values. In iterated implementations, we can also apply the proposed attack which can compute the 1-round and 16-round keys.

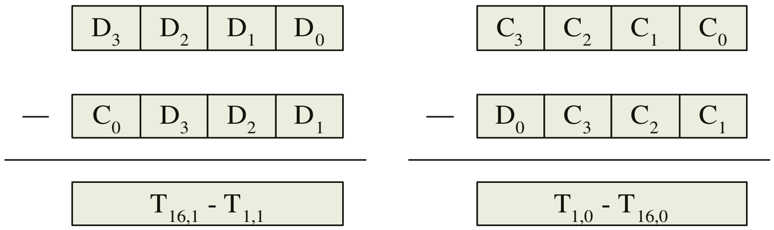


Fig. 6. Differential value between two temporary round keys.

4 Power Analysis Attack on SEED

Proposition 1. *If both the input and output value of the F function in i-round are known, i-round key can be computed using a reverse algorithm of the F function.*

DPA is a powerful attack in which an adversary collects a number of power traces from a hardware device as it repeatedly executes a cryptographic operation. In our DPA attack, an adversary must have knowledge of inputs processed by the device. Furthermore the same secret key is used in encryption (decryption) over multiple plain texts (cipher texts).

Our basic implementation of DPA is as follows. Assume that an adversary is able to input two 64-bit plain texts R_0 and L_0, and measures power consumption traces. Now, an adversary would like to attack the 1-round keys $K_{1,0}$ and $K_{1,1}$ as shown in Figure 7. Since the input and output of F function are known, according to Proposition 1, he can extract the 1-round key.

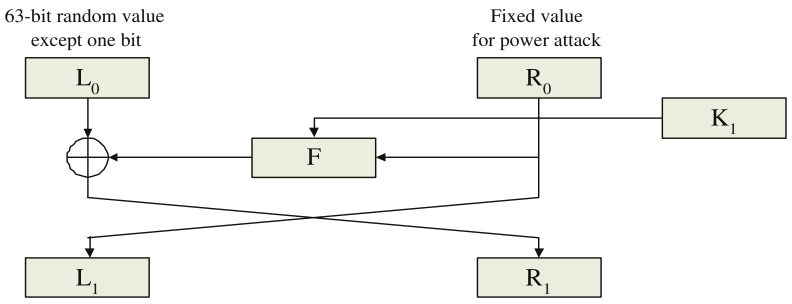

63-bit random value except one bit
Fixed value for power attack
L0
R0
K1
F
L1
R1

Fig. 7. The DPA attack in 1-round of SEED.

For example, we will find the least significant bit (LSB) of the output of F function at the 1-round. First, the value of the right half input R_0 is fixed during attack processing. Then, the left half input L_0 is randomly selected. We use a left input value which sets the LSB of L_0 to 0 or 1 and then measure the power signal at register R_1 during encryption processing. Let T_{it} be a sampled type of the power consumed. The i index corresponds to the ith power signal and t index corresponds to the time of the sample. Given the random L_0, the T_{it} are split into two sets according to the LSB of L_0 as follows.

$$T_0 = \{T_{it} \mid \text{LSB of } L_0 = 0\}$$
$$T_1 = \{T_{it} \mid \text{LSB of } L_0 = 1\}$$

Let T_0 be the set of measured traces where LSB is 0 and T_1, the set where LSB is 1. Both T_0 and T_1 will contain the same number of traces. Then we compute the average of the partitioning traces as follows:

$$A_0[t] = \frac{1}{|T_0|} \sum_{T_{it} \in T_0} T_{it}$$
$$A_1[t] = \frac{1}{|T_1|} \sum_{T_{it} \in T_1} T_{it}$$

Here, the number of measurements in a trace, $N = |T_0| = |T_1|$, depends on sampling rate and memory capacity. The differential trace of $A_0(t)$ and $A_1(t)$ is defined for $t = 1, ..., m$ as:

$$\triangle[t] = A_1[t] - A_0[t]$$

$$\lim_{N \to \infty} \triangle[t] = A_1[t] - A_0[t] = \begin{cases} 0 & \text{if } t \neq t^* \\ \varepsilon & \text{if } t = t^* \end{cases}$$

Assume that the register R_1 is stored at time t^* and t is equal to t^*. If the expected difference of power traces has a positive peak, $\varepsilon > 0$, then the LSB of F function of the 1-round is 0 because the storing power consumption for bit “1” is more than for bit “0”. Similarly, if the difference between power traces has a

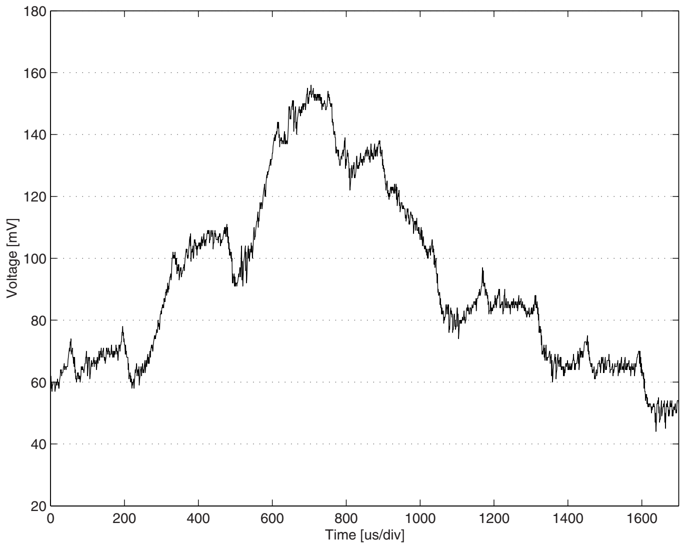


Fig. 8. Single power trace of an XOR operation which is $L_0 \oplus$ 64-bit of F function.

negative peak $\varepsilon < 0$, then the LSB of F function of the 1-round is 1. So, we can find the LSB bit of the output of F function at the 1-round. In addition, when t is not equal to t^*, the power dissipation is independent of the LSB because the smart card is manipulating bits other than the LSB.

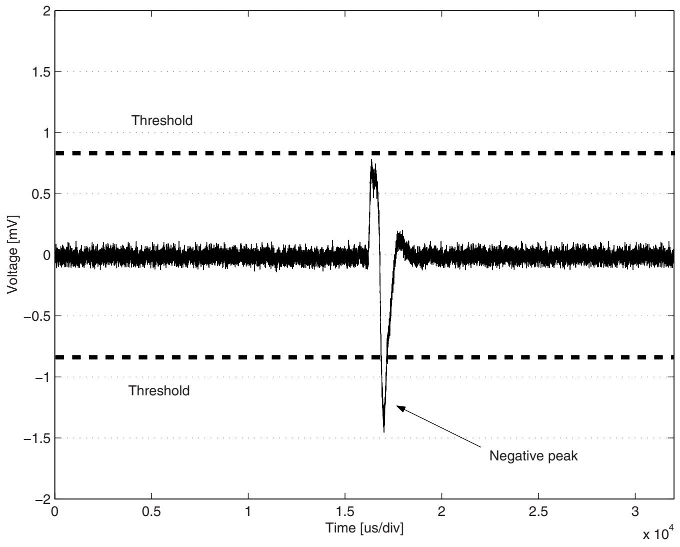
We show that SEED is vulnerable to our DPA by an experimental result. Our experiment was made on the 1-round implementation of SEED. Figure 8 shows a single power trace of an XOR operation to find the output of F function. Figure 9 illustrates the average difference between $A_0(t)$ and $A_1(t)$, in which we know whether the LSB of F function of the 1-round is 0 or 1. The signals in Figure 9 were obtained by averaging 5000 random power traces to observe a clear view of peak, but we have also been able to mount this attack with only 1000 power traces.

In our example, we have only discussed finding the LSB of F function at the 1-round. However, by a similar partition method to other bit using above measured traces, an adversary can steal all the bits of F function of the 1-round. As an above result and Proposition 1, we can extract the 1-round key.

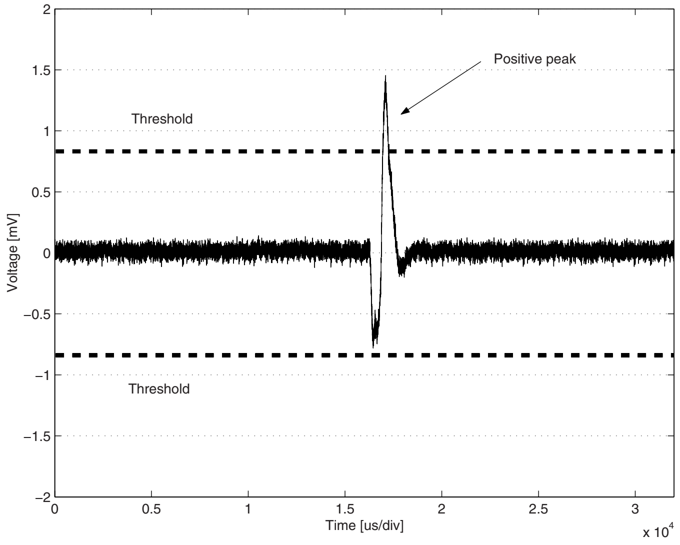
As mentioned above, if the 1-round key is vulnerable to DPA during the encryption processing, then the 16-round key can also be revealed during decryption. Therefore, an adversary can be compute a complete secret key from the 1-round and 16-round keys. The rest of this attack is similar to the fault attack described in section 3.2.

5 A Remark on Countermeasures and Conclusion

In this paper, we have shown that SEED is vulnerable to both a fault attack and a power attack. The basic assumption of the two attacks is that we can induce the



(a)



(b)

Fig. 9. Differential power traces. In the case (a), a negative peak is observed since the LSB bit of F function is 1, while in the case (b), a positive peak is observed since the LSB bit of F function is 0. We can make a decision about the threshold value by many experiments.

output values of F function through the side channel attack techniques. Since F function of SEED is recoverable if input and output are known, this assumption is quite reasonable and realistic. To achieve the real attack, we demonstrated by power analysis experiment described in section 4. Our attacks is applicable to other block ciphers with Feistel structure. Furthermore, if target cipher has a recoverable properties for F function under our assumption, the round key is easily extracted by an adversary. Unfortunately, the SEED has a weakness to leak the full secret key by analyzing with only two round keys.

Countermeasures for resisting the fault insertion attacks can be implemented using both hardware [16] and software methods to detect any intrusions by external voltage variations, external clock variations, and physical fault induction attack. To defeat the power analysis, a power signal reduction technique, a self-timed dual-rail method, a sense amplifier based logic, a non-deterministic processor, etc., are needed [8, 10, 13, 24]. Implementers need to consider these side channel attacks when designing secure smart card systems. It is also important to prove the validity of some countermeasures not given in this paper.

Acknowledgement

The authors wish to thank the anonymous referees for their helpful remarks and suggestions.

References

1. R. Anderson and M. Kuhn, "Tamper resistance – a cautionary note," In *Proc. of the Second USENIX Workshop on Electronic Commerce*, pp. 1-11, November 1996. Available from http://www.usenix.org
2. C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J. Seifrert, "Fault attacks on RSA with CRT: Concrete results and practical countermeasures," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 260–275, Springer-Verlag, 2002.
3. E. Biham and A. Shmir, "Differential cryptanalysis of the full 16-round DES," In *Advances in Cryptology – CRYPTO '92*, LNCS 740, pp. 487–496, Springer-Verlag, 1992.
4. E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," In *Advances in Cryptology – CRYPTO '97*, LNCS 1294, pp. 513–525, Springer-Verlag, 1997.
5. B. Boer, K. Lemke, and G. Wieke, "A DPA attack against the modular reduction within a CRT implementation of RSA," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 228–243, Springer-Verlag, 2002.
6. D. Boneh, R.A. DeMillo, and R.J. Liption, "On the importance of checking cryptographic protocols for faults," In *Advances in Cryptology – EUROCRYPT '97*, LNCS 1233, pp. 37–51, Springer-Verlag, 1997.
7. J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," In *Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 292–302, Springer-Verlag, 1999.
8. J.F. Dhem and N. Feyt, "Hardware and software symbiosis helps smartcard evolution," In *IEEE Micro 21*, pp. 14-25, 2001.
9. P. Dusart, G. Letourneux, and O. Vivolo, "Differential fault analysis on A.E.S," In *Applied Cryptography and Network Security – ACNS '03*, LNCS 2846, pp. 293–306, Springer-Verlag, 2003.

10. N. Feyt, "Countermeasure method for a microcontroller based on a pipeline architecture," United States Patent 20030115478, June 19, 2003.
11. C. Giraud, "DFA on AES," In *IACR, Cryptology ePrint Archive,* Available from http://eprint.iacr.org/2003/008/, 2003
12. J.C. Ha and S.J. Moon, "Randomized signed-scalar multiplication of ECC to resist power attacks," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 551–563, Springer-Verlag, 2002.
13. J.I. den Hartog, J. Verschuren, E.P. de Vink, J. Vos, and W. Wiersma, "PINPAS: a tool for power analysis of smartcards," In *Information Security Conference – SEC '03*, pp. 453–457, Kluwer Academic, 2003.
14. ISO/IEC JTC 1/SC27, *"Third Party Evaluation on SEED by CRYPTEC,"* ISO/IEC JTC 1/SC27 N3213, April 23, 2002.
15. M. Joye, A.K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults," In *Journal of Cryptology*, vol. 12, no. 4, pp. 241–245, 1999.
16. R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection of fault-based side-channal cryptanalysis of 128-bit symmetric block ciphers," Proc. of *IEEE DEsign Automation Conference*, pp. 579–585, 2001.
17. P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," In *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
18. Korea Information Security Agency, *Block Cipher Algorithm SEED*, Available from http://www.kisa.or.kr/seed/seed_eng.html.
19. A.K. Lenstra, "Memo on RSA signature generation in the presence of faults," September 1996.
20. T. Messerges, E. Dabbish, and R. Sloan, "Power analysis attacks of modular exponentiation in smartcards," In *Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 144–157, Springer-Verlag, 1999.
21. T. Messerges, "Securing the AES finalists against power analysis attacks," In *Fast Software Encryption – FSE '00*, LNCS 1978, pp. 150–164, Springer-Verlag, 2000.
22. J.A. Muir *Techniques of Side Channel Cryptanalysis*, masters thesis, 2001. Available from http://www.math.uwaterloo.ca/j̃amuir/sidechannel.htm.
23. S.P. Skorobogatov and R.J. Anderson, "Optical fault induction attacks," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 2–12, Springer-Verlag, 2002.
24. K. Tiri, M. Akmal, I. Verbauwhede, "A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards," In *28th European Solid-State Circuits Conference – ESSCIRC '02*, September, 2002.
25. C.D. Walter, "Some security aspacts of the MIST randomized exponentiation algorithm," In *Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 564–578, Springer-Verlag, 2002.
26. S.M. Yen, S.J. Moon, and J.C. Ha, "Hardware fault attack on RSA with CRT revisited," In *Information Security and Cryptology – ICISC '02*, LNCS 2587, pp. 374–388, Springer-Verlag, 2002.
27. S.M Yen, S.J. Moon, and J.C. Ha, "Permanent fault attack on the parameters of RSA with CRT," In *Information Security and Privacy – ACISP '03*, LNCS 2727, pp. 285–296, Springer-Verlag, 2003.

Secure and Efficient AES Software Implementation for Smart Cards

Elena Trichina[1] and Lesya Korkishko[2]

[1] Department of Computer Science, University of Kuopio,
P.O.B 1627 FIN-70211 Kuopio Finland
etrichin@cs.uku.fi

[2] Institute of Computer Information Technologies,
Ternopol Academy of National Economy, Ukraine
tko@tanet.edu.te.ua

Abstract. In implementing cryptographic algorithms on limited devices such as smart cards, speed and memory requirements had always presented a challenge. With the advent of side channel attacks, this task became even more difficult because a programmer must take into account countermeasures against such attacks, which often increases computational time, or memory requirements, or both.
In this paper we describe a new method for secure implementation of the Advanced Encryption Standard algorithm. The method is based on a data masking technique, which is the most widely used countermeasure against power analysis and timing attacks at a software level. The change of element representation allows us to achieve an efficient solution that combines low memory requirements with high speed and resistance to attacks.

1 Introduction

The symmetric block cipher Rijndael [4] was standardized by the National Institute of Science and Technology (NIST) in November 2001, and will be used in a large variety of applications, from mobile consumer products to high-end servers. Consequently, the requirements and design criteria for AES implementations vary considerably.

Small footprint, stringent memory requirements, low power consumption and high throughput used to be standard criteria for implementation of cryptographic algorithms designated for smart cards and related embedded devices. With the advent of side channel attacks, one of the major concerns is resistance to such attacks.

The most general method to counter side channel attacks is to randomize data that may leak through various side channels, such as power consumption [10], electromagnetic radiation [17], or execution time [11]. The problem is to guarantee that an attacker may obtain only random information, and thus cannot gain any useful knowledge about the actual initial and/or intermediate data involved in computations [12].

In the AES algorithm most operations work on bytes. To protect against side channel attacks, every byte that appears as an intermediate result must look random. However, this is not easy to achieve. The problem is that the algorithm combines additive and multiplicative operations, which implies complex transformations on masks [3, 7, 1]. Moreover, as it turned out, a straightforward multiplicative mask [1] is not secure against so-called zero attack [6].

The main contribution of this paper is that we suggest a method that combines a full protection against side channel attacks (including zero attack) with low memory requirements and low computational costs. This became possible due to a change of representation of field elements and using so called *log*- and *alog*-tables [20, 16] for arithmetic computations in Galois fields directly on masked data. To reinforce security, ideas of computations on masked tables [19] were incorporated in our implementation.

The rest of the paper is organized as follows. After a brief description of the Advanced Encryption Standard algorithm in the next chapter, we proceed in Chapter 3 with details of a very efficient AES implementation based on a so-called discrete logarithm representation of elements in $GF(2^n)$. The latter allows us to reduce multiplication and inversion in Galois fields to table lookups and simple integer arithmetic operations.

Chapters 4 and 5 discuss the difficulties of inversion on masked data (i.e., data that are obtained by XOR-ing every (i, j)-th byte of the state with a byte of a random mask) and outline an efficient and secure method of inversion using masked *log*/*alog* lookup tables.

The paper is concluded with the the summary of the novel secure AES software implementation suitable for even the most limited smart cards and other embedded devices.

2 AES Reminder

AES encryption and decryption are based on four different transformations that are performed repeatedly in a certain sequence; each transformation maps an input state into an output state. The transformations are grouped in rounds and are slightly different for encryption and decryption. The number of rounds depends on the key/block size.

Figure 1 illustrates the general structure of the AES algorithm. Compared to encryption, decryption is simply an execution of the inverse transformations in the inverse order.

For simplicity, we describe only the 128-bit block- and and key-size version of the algorithm; although important design parameters, block and key sizes have no bearing on the content of the paper. For a complete mathematical specification of the AES algorithm we refer readers to [5].

In the standard Rijndael, a 128-bit data block is considered as a 4×4 array of bytes (usually referred as a state). The algorithm consists of an initial data/key addition, 9 full rounds, and a final (modified) round. A separate key scheduling module is used to generate all the sub-keys, or *round keys*, from the initial key. A sub-key is also represented as 4×4 array of bytes.

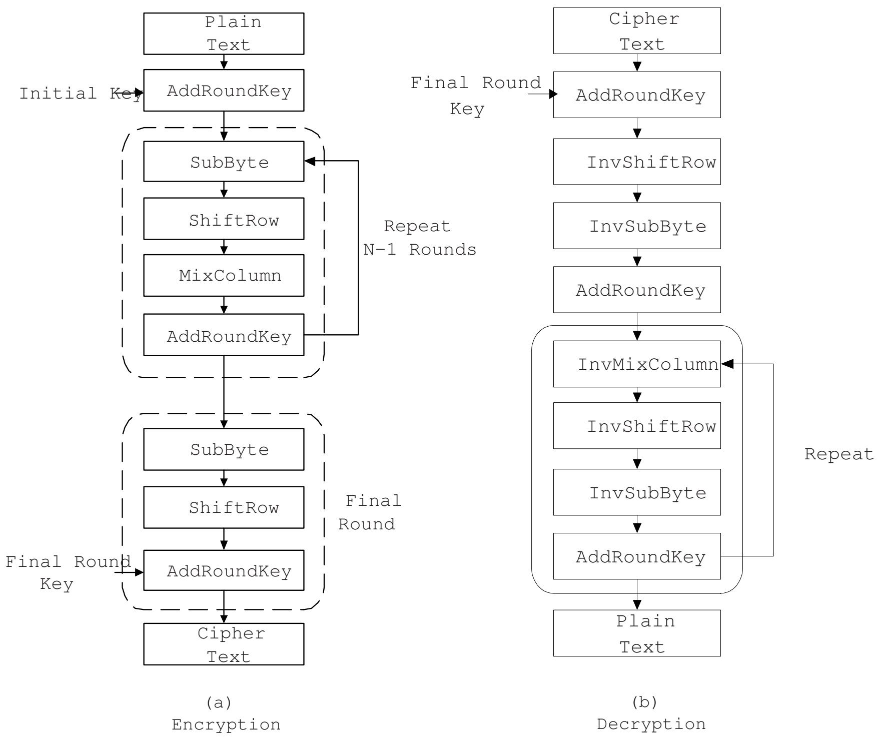


Fig. 1. The structure of the AES encryption and decryption algorithms.

The full round involves four steps.

– The *Byte Substitution*, or *SubBytes* step replaces each byte in a block by its substitute in an S-box. The S-box is an invertible substitution which is constructed by a composition of two transformations:
  - First, each byte A of a state is replaced with its reciprocal in $GF(2^8)$ except that 0, which has no reciprocal, is replaced by itself.
  - Then, an affine transformation f is applied. It consists of a bitwise matrix multiply with a fixed 8×8 binary matrix M, after which the resultant byte is XOR-ed with the hexadecimal number $\{63\}$.

  The S-box is usually implemented as a look-up table consisting of 256 1-byte entries, but also can be computed "on-the-fly".
– Next comes the *Shift Row* step. Each row in a 4×4 array of bytes of the state is shifted 0, 1, 2 or 3 bytes to the left in a round fashion, producing a new 4×4 array of bytes.
– In the *Mix Column* operation, each column in the 4×4 array of bytes is considered as polynomial over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.
  Since multiplication is carried out in $GF(2^8)$, the product is calculated modulo irreducible polynomial $m(x) = x^8+x^4+x^3+x+1$, or $1\{1b\}$ in hexadecimal representation.

– The final step, *Add Round Key*, simply XOR-es the result with the sub-key for the current round.

In parallel to the round transformation, the round key is computed in the *Key Scheduling Block*. The round key is derived from the cipher key by means of key expansion and round key selection.

– The expanded key represents a linear array of 4-byte words, where the first N_k words (where N_k is equal to the key length divided by 32) are filled in with the cipher key. Every following word $W[i]$ is obtained by XOR-ing the words $W[i-1]$ and $W[i-N_k]$. For words in positions that are multiples of N_k, the word is first rotated by one byte to the left; then its bytes are transformed using the S-box from the *Byte Substitution* step, and XOR-ed with the round-dependent constant.
– Round keys are taken from the expanded key in the following way: the first round key consists of the first N_b words (where N_b is equal to the block length divided by 32), the second of the following N_b words, etc.

There are many design tradeoffs to consider when implementing the AES algorithm. In applications such as smart cards, the program's footprint, memory requirements, power consumption and throughput are important considerations.

In the next section we propose an implementation method that can realize both, encryption and decryption. The method requires only 512 bytes of memory and totally avoids Mips-intensive computations such as multiplication. What is more important, the method can be extended to accommodate secure computations on masked data without a penalty of extra memory.

3 Implementing AES Encryption and Decryption Using log/alog Tables

To do calculations in a finite field $GF(2^n)$, the field elements are represented in a basis. Most software implementations use a standard basis, where elements are represented as polynomials of the form $a_0 + a_1x + ... + a_{n-1}x^{n-1}$, where all a_i are elements in $GF(2)$, and addition is done modulo 2. Field operations on these elements consist of computations on polynomials, e.g., field multiplication can be calculated as a multiplication of two polynomials followed by a reduction of the result modulo some fixed irreducible polynomial degree n.

In [20] it was suggested to use a different representation of field elements. The new representation is based on the fact that all non-zero elements in a finite field $GF(2^n)$ can be obtained by exponentiation of a *generator* in this field. So after choosing a basis for $GF(2^n)$, we look for a field generator γ and calculate all pairs (α, i) such that $\alpha = \gamma^i, 0 \leq i \leq 2^n - 1$, $\alpha \in GF(2^n) \setminus \{0\}$. Such representation of non-zero elements is $GF(2^n)$ is unique for a fixed primitive element γ; i is the discrete logarithm of α with respect to γ.

The pairs are stored in two tables, a *log*-table sorted on α, and an *alog*-table sorted on i. Each table takes $2^n - 1$ words of n bits.

3.1 Computations in Galois Fields Using log- and alog Tables

The *log/alog* tables can be used to calculate in $GF(2^n)$ very efficiently, avoiding all Mips-intensive operations [20].

A **sum** of two field elements, α and β is calculated as a bitwise *XOR* operation:

$$\alpha + \beta = \alpha \oplus \beta.$$

A **product** of two non-zero elements can be calculated with three table lookups:

$$\alpha \cdot \beta = alog[(log[\alpha] + log[\beta]) \mod (2^n - 1)].$$

An **inversion** operation on a non-zero element from $GF(2^n)$ can be calculated with two table lookups only:

$$\alpha^{-1} = alog[-log[\alpha] \mod (2^n - 1)].$$

Inversion in the field is defined only for non-zero elements, while zero is always mapped into itself by convention. In order to avoid checking for zero, we use an idea from [9] and augment the *log* and *alog* tables with one more value. Namely, let $log[0] = 2^n - 1$ and $alog[2^n - 1] = 0$; then

```
procedure invert (alpha)  // alpha is in the interval 0 .. 2^n - 1
{
  temp = (2^n - 1) - alpha;
  if(0 < temp && temp < 2^n - 1) return alog[temp];
  else                           return alog[alpha];
}.
```

Taking this augmentation into account, the multiplication can be expressed as

```
procedure mult (alpha, beta)
{
   temp = (log[alpha] + log[beta]) mod (2^n -1);
   switch(temp) {
      case temp == alpha: return alog[beta];
      case temp == beta:  return alog[alpha];
      default:            return alog[temp];
   }
}.
```

The new representation allows us to obtain a very compact, efficient and flexible implementation of the AES algorithm that can be used for both, encryption and decryption. How it is done is described below.

3.2 Implementation of Round Operations Using log/alog Tables

Maintaining pre-computed tables to simplify operations and improve performance is a common practice. For AES, a method to combine different operations of the round transformation in a single set of table lookups was suggested in [4]. This approach basically combines the matrix multiplication required in the *MixColumn* operation with the *S-box*, and involves 4 tables with 256 4-byte entries, i.e., 4KByte of memory. Because encryption and decryption must use different tables, the total memory requirement amounts to 8KByte.

Another solution is to trade memory for speed, and use two 256-byte lookup tables for the *SubByte* and *InvSubByte* operations, while implementing the *MixColumn/InvMixColumn* operations separately. Here, again, various trade-offs are possible.

Each call to the *MixColumn* or *InvMixColumn* operations results in sixteen field multiplications. A straightforward implementation of the multiplication operation in the field is Mips-intensive. Since one of the multiplicands is fixed (with values limited to 6 field elements, i.e., $\{01\}, \{02\}$ and $\{03\}$ for *MixColumn* and $\{0b\}, \{0d\}$, and $\{09\}$ for *InvMixColumn*), a conventional field multiplication operation can be replaced by a table lookup, requiring a new 6×256 table, each element of which is 8-bit wide.

Another solution is to implement field multiplication by repeated application of the `xtime` operation. The latter can be realized either as a left shift and a subsequent conditional bitwise XOR with $\{1b\}$, or as a table lookup, requiring only 256-byte table. This approach involves slight computational overhead (on the average three *xtime* and three field addition operations per one multiplication in the field) and is less efficient than table lookups, but saves memory, and is often used for memory-constrained devices and 8-bit microprocessors.

The *log*/*alog*-tables have been used in [16] to provide an efficient software method to compute the *MixColumn* and *InvMixColumn* operations. In comparison with a conventional table lookup for *MixColumn/InvMixColumn*, the new solution reduces memory requirements from 6×256 bytes to 2×256 bytes only, which is an important factor for smart cards.

Further memory optimization can be achieved by using the *log/alog*-tables to compute the *SubByte/InvSubByte* operations by exploiting similarities between encryption and decryption. For this, it is necessary to implement byte substitution in two separate steps, i.e., as a composition of inversion in the field and an affine transformation. While the affine transformations used for encryption and decryption are slightly different, inversion in $GF(2^8)$ can be calculated using the same *log/alog*-tables.

The affine transformation used for encryption consists of shifts and simple multiplications:

```
procedure affine_encrypt (x) // x contains the input byte of the state
{
   tmp=(((x&1)*0xff)&c1);
   tmp^=((((x&2)>>1)*0xff)&c1);
   tmp^=((((x&4)>>2)*0xff)&c2);
   tmp^=((((x&8)>>3)*0xff)&c3);
   tmp^=((((x&16)>>4)*0xff)&c4);
   tmp^=((((x&32)>>5)*0xff)&c5);
   tmp^=((((x&64)>>6)*0xff)&c6);
   tmp^=((((x&128)>>7)*0xff)&c7);

   return tmp^=c;  // tmp contains the output byte of the state
}
```

The inverse affine transformation can be implemented in a similar way. Thus, the same *log/alog* tables can be used in both, the *MixColumn* and *SubByte* operations for encryption, and in the *InvMixColumn* and *InvSubByte* operations for decryption. Sharing *log*- and *alog*-tables reduces the total memory requirements for complete AES implementation to 512 bytes only. At the same time, all Mips-intensive (and thus, power consuming) field arithmetic operations are replaced by table lookups, ensuring not only memory- but also time- and power-efficient implementation. As an additional bonus, an overall program that realizes both, encryption and decryption, has a small footprint.

The combination of these three properties makes the described solution ideal for smart cards and related embedded devices. In the remaining part of the paper we show how *log/alog* tables can be used for efficient and secure computations on *masked* data.

4 Protection Against Side Channel Attacks with Data Masking

Basically, side-channel attacks work because there is a correlation between the physical measurements taken during computations (e.g., power consumption [10], EMF radiation [17], time of computations [11]) and the internal state of the processing device, which itself is related to a secret key.

Among many attacks, the Differential Power Analysis (DPA) is the most dangerous (see, for example, [15]). It uses statistical analysis to extract information from a collection of power consumption curves obtained by running an algorithm many times with different inputs. Then an analysis of a probability distribution of points on the curves is carried on. The DPA attack uses correlations between power consumption patterns and specific key-dependent bits which appear at known steps of the cryptographic computations. For example, a selected bit b at the output of one S-box of the first round of the AES will depend on the known input message and 8 unknown bits of the key. The correlation between power consumption and b can be computed for all 256 values of 8 unknown bits of the key. The correlation is likely to be maximal for the correct guess of the 8 bits of the key. Then an attack can be repeated for the remaining S-boxes.

There are many strategies to combat side-channel attacks. Among software countermeasures against SPA/DPA are such techniques as introducing dummy instructions and/or random wait states, balancing of Hamming weight of internal data, randomization of instruction executing sequence. The most powerful software countermeasure appears to be so-called *bit splitting* [2, 8, 15], which in case when each bit is split into two shares can be reduced to *masking* data with random values. The idea how to apply data masking to AES is simple: the message, as well as the key, are masked with some random masks at the beginning of computations, and thereafter everything is almost as usual. Of course, the value of the mask at the end of some fixed step (e.g., at the end of a linear part of computations or at the end of a round) must be known in order to re-establish the expected value at the end of the execution; we call this *mask correction.*

A traditional XOR operation is used as a masking countermeasure; however, the mask is arithmetic on $GF(2^8)$ [3]. The operation is compatible with the AES structure except for *SubByte*, which is the only non-linear transformation since it uses inversion in the field. In other words, it is easy to compute mask correction for all transformations in a round, apart from the inversion step of the *SubByte*. Namely, if every byte A of the (initial or intermediate) state is masked with some random mask R, then $\mathbf{OP}(A \oplus R) = \mathbf{OP}(A) \oplus \mathbf{OP}(R)$, where $\mathbf{OP} \in \{$*MixColumn/InvMixColumn*, *ShiftRow/InvShiftRow*, *AddRoundKey*$\}$. Thus, given any random mask, it is easy either to pre-compute the corresponding mask correction, or to compute it on-the-fly, in parallel with computations on *masked* data.

This property also holds for an affine part of the *SubByte* operation, but does not hold for inversion in the field. To overcome this difficulty, Akkar and Giraud [1] proposed a so-called transformed masking. In this method, first an additive mask is replaced by a multiplicative mask in a series of multiply and add operations in $GF(2^8)$, after which normal inversion takes place, and finally, a transformation of a multiplicative mask into an additive mask is carried out. Unfortunately, as was pointed out in [6], a multiplicative mask does not blind zero element in $GF(2^n)$, enabling a so-called zero attack.

The paper [19] suggested a simplification of the aforementioned technique. At the same time it proposed a simple but efficient way to enhance security with respect to the zero attack by conduction on-the-fly generation of the *masked lookup tables* used for implementation of the inversion operation.

In what follows we tackle the problem of non-linear operations on masked data from a different angle. Using a non-standard representation of elements in $GF(2^n)$ allows us to implement inversion on masked data and to compute the corresponding mask correction in a secure and efficient manner.

5 Inversion on Masked Data Using log/alog-Tables

The problem can be formulated as follows. Given $A \oplus R$, where A is a byte of a state and R some uniformly distributed random value, find an efficient method to compute the value $A^{-1} \oplus \tilde{R}$, never revealing A or A^{-1} in a process. Here $\tilde{R}$ can be either equal to R or any other (uniformly distributed) random value.

The solution is based on the following observations.

– First, a new representation allows us to infer how $(A \oplus R)^{-1}$ differs from A^{-1}:
  1. A field element $A \oplus R$ can be represented as $\gamma^y = \gamma^i \oplus \gamma^r$, where γ^i is a representation of the unknown byte A of the state, and γ^r is a representation of the known random mask R.
  2. By simple formulae manipulations, we obtain
$$\gamma^y = \gamma^i \oplus \gamma^r = \gamma^i \cdot (\gamma^{r-i} \oplus \mathbf{1}).$$
  3. Therefore, we can write
$$\gamma^{-y} = (\gamma^i \oplus \gamma^r)^{-1} = (\gamma^i)^{-1} \cdot (\gamma^{r-i} \oplus \mathbf{1})^{-1}.$$

Hence, $(\gamma^{r-i} \oplus \mathbf{1})^{-1}$ is the mask correction for "masked inversion".

– Next, we show how to compute this mask correction without revealing A or A^{-1}.
  1. We can consider $A \oplus R$ from a different view point, namely as
  $$\gamma^y = \gamma^i \oplus \gamma^r = \gamma^r \cdot (\gamma^{i-r} \oplus \mathbf{1}).$$
  2. Hence, if we multiply γ^y by γ^{-r}, we get
  $$\gamma^y \cdot \gamma^{-r} = \gamma^{-r} \cdot \gamma^r \cdot (\gamma^{i-r} \oplus \mathbf{1}) = \gamma^{i-r} \oplus \mathbf{1}.$$
  3. Executing $(\gamma^{i-r} \oplus \mathbf{1}) \oplus \mathbf{1}$, we find γ^{i-r}, after which using *log*- and *alog*-tables, we easily compute $(\gamma^{i-r})^{-1} = \gamma^{r-i}$.
  4. Finally, after *XOR*-ing γ^{r-i} with $\mathbf{1}$ and inverting the result, we find the mask correction $(\gamma^{r-i} \oplus \mathbf{1})^{-1}$.

Figure 2(a) summarizes the computation flow for calculating inversion on masked data and the respective mask correction.

5.1 MixColumn/InvMixComumn on Masked Data with log/alog Tables

It is easy to implement the *MixColumn* and *InvMixColumn* operations on masked data using *log*/*alog* tables as well. Indeed, since one of the terms in each field multiplication involved in these operation is fixed, the operation is linear. Let I denotes a fixed term, then $(A \oplus R) \cdot I = (A \cdot I) \oplus (R \cdot I)$. The corresponding mask correction can be computed trivially as $R \cdot I$.

Hence, each of the *MixColumn/InvMixColumn* operations on masked data is reduced to $2 \times (16 \times 3)$ table lookups using the same *log/alog*-tables that were used to compute inversion.

6 Coping with "Zero Attack"

The previously published proposals to randomize AES efficiently, such as multiplicative [1] and simplified [19] masking techniques, had a subtle flaw, namely, they were vulnerable to a zero attack [6]. The attack is based on the fact that a multiplicative mask masks only non-zero values. In other words, if the actual data byte A is zero, then for any mask X, $(A \otimes X) = 0$.

Using Figure 2(a) as a reference, let us analyze how robust the new method is with respect to the zero attack.

First of all, notice that all manipulations on discrete logarithms are fully protected as long as random masks R change from one run to another.

On the other hand, detecting that an intermediate value γ^{i-r} is zero provides some dangerous information. Indeed, γ^{i-r} is in fact equivalent to $A \otimes R^{-1}$, where $A = (data \oplus key)$, and R^{-1} is a mask. $\gamma^{i-r} = 0$ implies that either $A = 0$ or $R^{-1} = 0$. Hence an attacker may systematically try all 256 possible values for *data* in order to find the one which turns A into zero.

In order to protect the inversion on masked data from this situation, we will have to implement *log/alog* table lookups in a secure way.

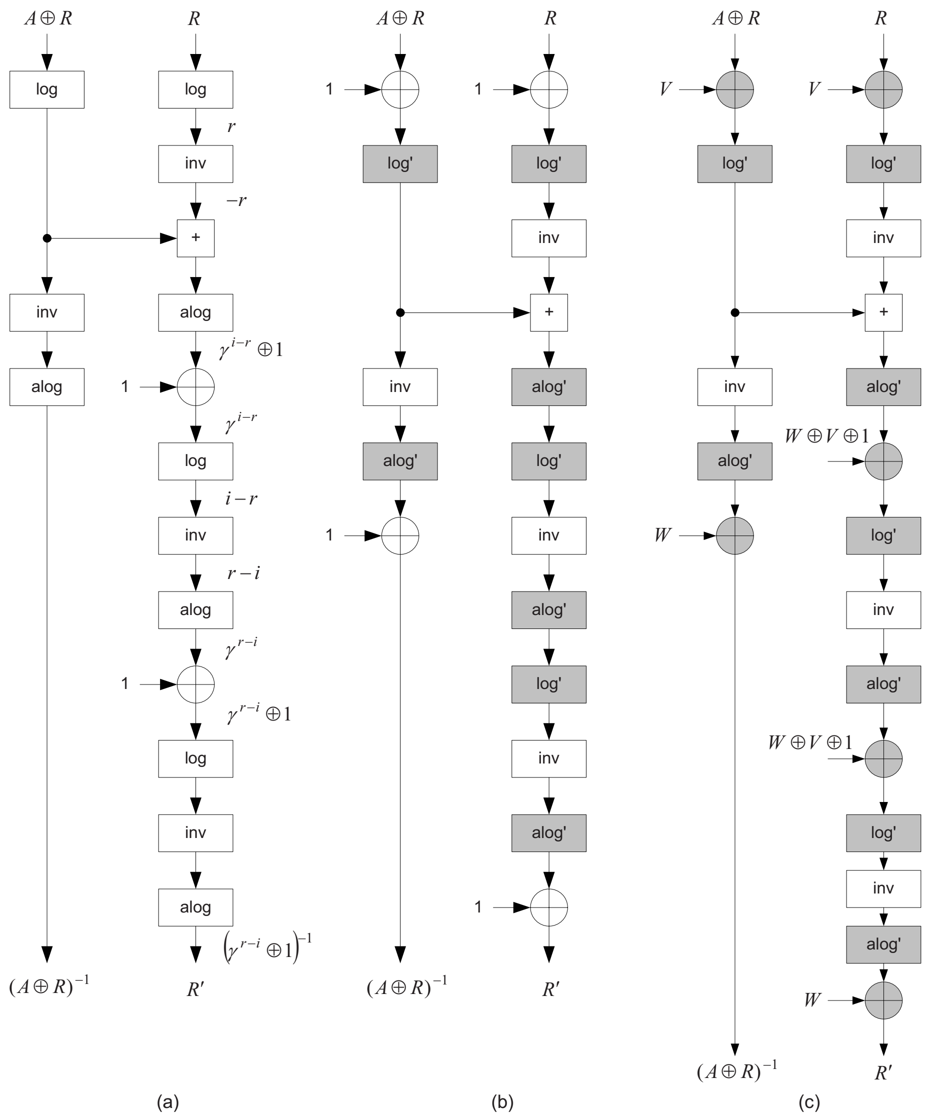


Fig. 2. Computing inversion on masked data with log/alog tables.

6.1 Masked log/alog Tables

The simplest and the least expensive solution is to modify the inversion algorithm in the following way. Instead of *XOR*-ing the result of inversion in the mask correction branch with 1 and thus exposing γ^{i-r}, we use fixed "correction" tables *log'* and *alog'*, such that $log'[\gamma^i \oplus 1] = i$ and $(alog'[i] = \gamma^i \oplus 1$. This necessitates slight data corrections in the processing flow, as shown in Figure 2(b).

A more general and more secure solution consists in using the technique similar to the one suggested in [14]. We assume that the standard *log* and *alog* tables are stored in EEPROM. Prior to running AES, we fix a pair of independent random bytes V and W. V is the value that masks every input byte of the original *log* table, while W masks every output byte of the original *alog* table. Then we recompute the *log* and *alog* tables (using, for example, the fast "split and swap" algorithm from [19]) in such a way that for every $\alpha = \gamma^i$, $\alpha \in GF(2^8)$, the following holds:

$$log'[\alpha \oplus V] = i \quad and \quad alog'[i] = \alpha \oplus W.$$

Re-computations can be done while downloading the tables from EEPROM to RAM prior to running the AES algorithm.

Of course, the computation flow for inversion (as well as for multiplications in *MixColumn/InvMixColumn*) should be modified by *XOR*-ing, where necessary, inputs to the *log*' table with V and outputs of the *alog*' table with W.

The complete information flow for computations of inversion in the field using masked *log/alog* tables is depicted in Figure 2(c). The data correction for the *MixColumn/InvMixColumn* operations is trivial.

6.2 Combining log/alog Tables with Random Data Partitioning

In this section we describe a new solution that re-enforces the previous result. It is based on the data splitting [2] idea applied to the mask correction computations.

Indeed, we can represent a masked data as three "slices":

$$A \oplus R = \gamma^i \oplus \gamma^r = \gamma^{i-k} \cdot (\gamma^{r-i+k} \oplus \gamma^k),$$

where γ^k, γ^r are some uniformly distributed random values. Then after inversion we obtain:

$$(A \oplus R)^{-1} = \gamma^{-i+k} \cdot (\gamma^{r-i+k} \oplus \gamma^k)^{-1} = \gamma^{-i} \cdot \gamma^k \cdot (\gamma^{r-i+k} \oplus \gamma^k)^{-1}.$$

From the last equation we infer the mask correction:

$$\gamma^k \cdot (\gamma^{r-i+k} \oplus \gamma^k)^{-1}.$$

In order to understand how to compute this mask correction, we re-write the masked data as:

$$A \oplus R = \gamma^i \oplus \gamma^r = \gamma^{r+k} \cdot (\gamma^{i-r-k} \oplus \gamma^{-k}).$$

This equation is used as a basis for our mask correction algorithm. Indeed, multiplying $(A \oplus R)$ by γ^{-r} and γ^{-k} (in any order), we get:

$$(A \oplus R) \cdot \gamma^{-r} \cdot \gamma^{-k} = \gamma^{r+k} \cdot (\gamma^{i-r-k} \oplus \gamma^{-k}) \cdot \gamma^{-r} \cdot \gamma^{-k} = \gamma^{i-r-k} \oplus \gamma^{-k}.$$

Then *XOR*-ing the result with γ^{-k} and inverting the sum, we compute γ^{r-i+k}. Next, *XOR*-ing this value with with γ^k (i.e., $\gamma^{r-i+k} \oplus \gamma^k$), inverting the result and multiplying it by γ^k eventually produces a mask correction:

$$\gamma^k \cdot (\gamma^{r-i+k} \oplus \gamma^k)^{-1}.$$

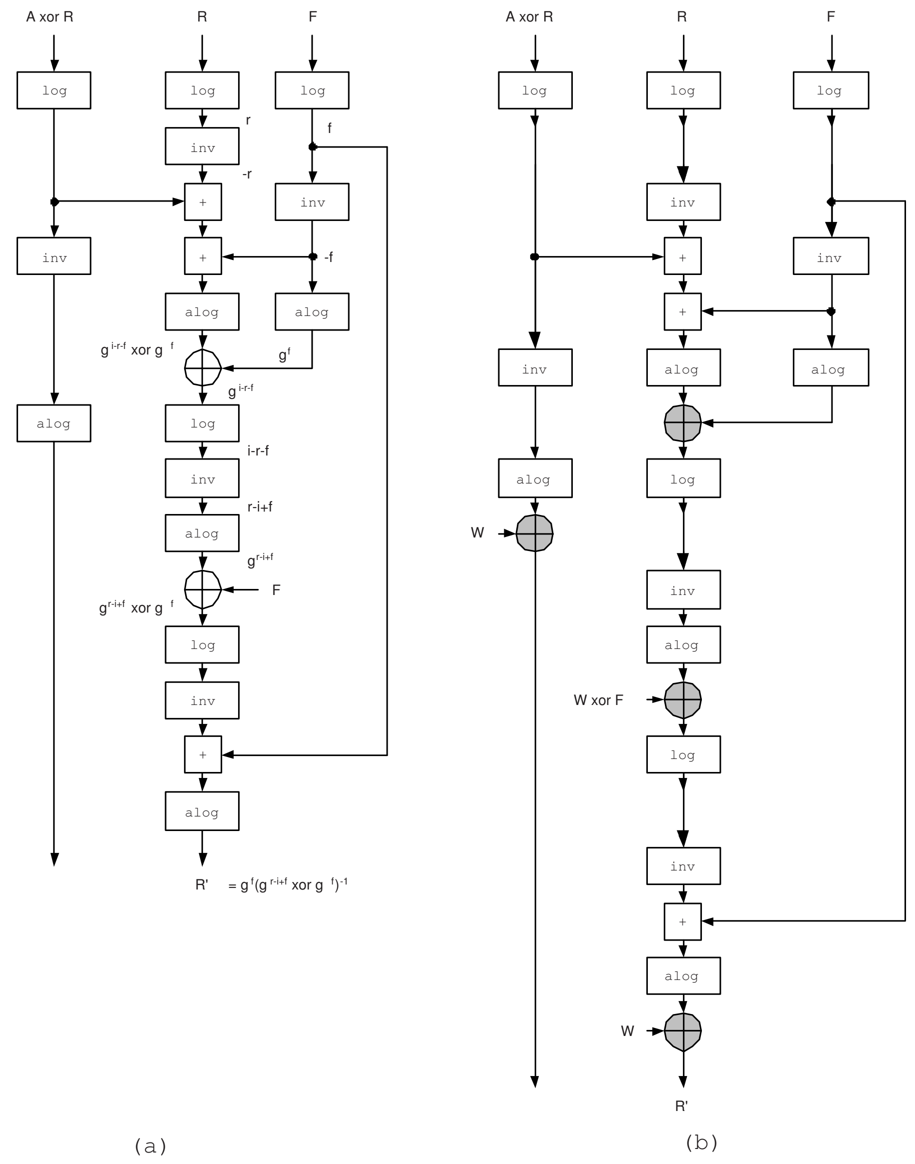


Fig. 3. Information processing flow for computing inversion on masked data using log/alog tables with random data partitioning.

The information processing flow for computing inversion on masked data and the corresponding mask correction is depicted in Figure 3(a).

If this algorithm uses original *log* and *alog* tables, there is still a part of the computation flow that is potentially vulnerable to the zero attack. Namely, it is the part between two *XOR* operations, where after the first *XOR* with γ^k we obtain a value equivalent to γ^{i-r-k} and after a subsequent inversion we get

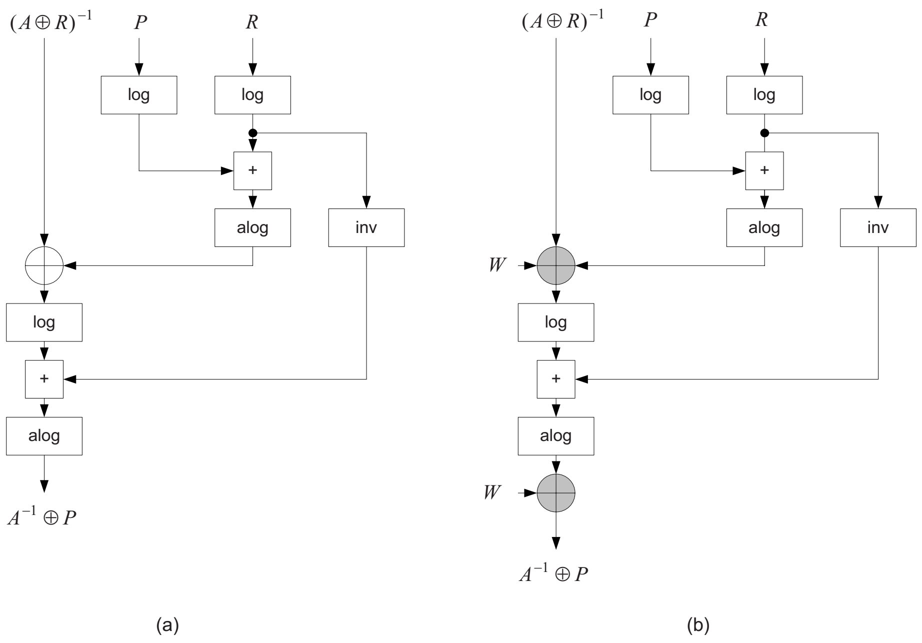


Fig. 4. Transformation of a multiplicative mask into a boolean mask.

γ^{r-i+k}. If these values are both zero, there can be three potential cases: $\gamma^i = 0$, or $\gamma^{r+k} = 0$, or $\gamma^k = 0$ (we assume that $R \neq 0$). Hence, an attacker can mount a second-order DPA attack, and discover the key.

In order to protect against this hazard, we mask all outputs of the *alog* table by some random value W. When W is fixed prior to running of the AES algorithm, the *alog* table can recomputed (using the algorithm from [19]) on-the-fly during downloading it from EEPROM to RAM: $alog'[i] = \gamma^i \oplus W$. The corresponding modification in the information flow processing for computing the mask correction for inversion in the field is depicted in Figure 3(b).

After having computed the mask correction, the transformation of the multiplicative mask back into a boolean mask can be computed as shown in Figure 4.

7 Conclusion

In this paper we propose a new solution to the problem of software implementation of the AES encryption/decryption algorithm secure against DPA attacks. Our solution is based on the discrete logarithm representation of elements in $GF(2^n)$. The field operations are effectively reduced to table lookups and simple operations like shift, *XOR* and integer arithmetic.

Only two tables are used for the entire round, one is a 256-byte *log*-table, sorted on field elements, and another is a 256-byte *alog* table sorted on discrete logarithms. The same tables are used for *SubByte* and *MixColumn* operations in encryption, and their counterparts in decryption process.

The implementation can be easily extended to accommodate the most versatile countermeasure against power analysis attacks, namely, data masking. There is no memory overhead involved, and computational overhead includes only simple table manipulations while downloading it from EEPROM to RAM and unavoidable computations of mask corrections. The latter amounts to *XOR* and simple integer arithmetic operations.

A danger of the zero attack is eliminated at the price of few additional *XOR* operations and on-the-fly table re-computations for every run of the algorithm. The latter can be efficiently executed using algorithm from [19] while coping the content of *log/alog* tables from EEPROM to RAM prior to running AES.

To our best knowledge, the proposed solution is the most efficient secure software implementation of the AES encryption/decryption algorithm published so far. Four factors make it ideal for limited devices such as smart cards, namely, (1) the program has a very small footprint, (2) it is fully protected against DPA/SPA attacks (including zero attacks), (3) the program does not have Mips-intensive power consuming operations using table lookups instead, and (4) only 512 bytes of RAM are necessary to store lookup tables that are used for both, encryption and decryption.

References

1. Akkar, M., Giraud, C.: An implementation of DES and AES, secure against some attacks, In Proc. *Cryptographic Hardware and Embedded Systems: CHES 2001*, volume 2162 of Lecture Notes in Computer Science, pp. 309-318, Springer-Verlag, 2001.
2. Chari, S., Jutla, C., Rao, J., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks, In Proc. *Advances in Cryptology – Crypto'99*, volume 1666 of Lecture Notes in Computer Science, pp. 398-412, Springer-Verlag, 1999.
3. Coron, J.-S., Goubin, L.: *On boolean and arithmetic masking against differential power analysis*. In Proc. *Cryptographic Hardware and Embedded Systems: CHES 2000*, volume 1965 of Lecture Notes in Computer Science, pp. 231-237, Springer-Verlag, 2000.
4. Daemen, J., Rijmen, V.: *AES Proposal: Rijndael*, AES submission, 1998. Available at URL `<http://csrc.nist.gov/encryotion/aes/aes_home.htm>`.
5. Daemen J., Rijmen, V.:*The design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag Berlin Heidelberg, 2002.
6. Golić, J., Tymen, Ch.: *Multiplicative masking and power analysis of for AES*. In Proc. *Cryptographic Hardware and Embedded Systems: CHES 2002*, volume 2523 of Lecture Notes in Computer Science, pp. 198-212, Springer-Verlag, 2002.
7. Goubin, L.: *A sound method for swirching between boolean and arithmetic masking*. In Proc. *Cryptographic Hardware and Embedded Systems: CHES 2001*, volume 2162 of Lecture Notes in Computer Science, pp. 3-15, Springer-Verlag, 2001.
8. Goubin, L., Patarin, J.: *DES and differential power analysis*, In Proc. *Cryptographic Hardware and Embedded Systems: CHES'99*, volume 1717 of Lecture Notes in Computer Science, pp. 158-172, Springer-Verlag, 1999.
9. Huang, C., Xu, L.: *Fast software implementation of finite field operations*, Technical Report, Washington University in St. Louis, December 2003. Available at URL `<http://www.nisl.wustl.edu/Papers/Tech/GF.pdf>`.

10. Kocher, P., Jaffe, J., Jun,B.: *Differential power analysis*, In Advances in Cryptology – CRYPTO'99, volume 1666 of Lecture Notes in Computer Science, pp. 388-397, Springer-Verlag, 1999.
11. Kocher, P.: *Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems*, In Proc. *Advances in Cryptology – Crypto'96*, volume 1109 of Lecture Notes in Computer Science, pp. 104-113, Springer-Verlag, 1996.
12. Kocher, P., Jaffe, J., Jun, B.: *Using unpredictable information to minimize leakage from smartcards and other cryptosystems*, USA patent, International Publication number WO 99/63696, December 9, 1999.
13. C. C. Lu, S-Y. Tseng, *Integrated design of AES (Advanced Encryption Srandard) encryptor and decryptor*, In proceedings IEEE conf. on Application-Specific Systems, Architectures, and Processors (ASAP'02), pp. xxx, IEEE, 2002.
14. Messerges, T. S.: *Securing the AES finalists against power analysis attacks*, In Proc. *Fast Software Encryption Workshop 2000*, volume 1978 of Lecture Notes in Computer Science, pp. 150-165, Springer-Verlag, 2000.
15. Messerges, T. S., Dabbish, E. A., Sloan, R. H.: *Examining smart-card security under the thread of power analysis*, IEEE Trans. Computers, vol. 51, no. 5, pp. 541-522, May 2002.
16. Jing Zheng Ouyang, *Efficient method for multiplication over Galois fields*, U.S. patent pub. number US2003/0128841 A1; July 10, 2003.
17. J. J. Quisquater, D. Samide, *Electromagnetic analysis (ema): measures and counter-measures for smart cards*, In proceedings *Smartcard Programming and Security*, volume 2140 of Lecture Notes in Computer Science, pp. 200-210, Springer-Verlag, 2001.
18. V. Rijmen, *Efficient implementation of Rijndael SBox*, `<http://www.esat.kuleuven.ac.be/~rijmen/rijndael>`
19. Trichina, E., De Seta, D., Germani, L.: *Simplified Adaptive Multiplicative Masking for AES and its secure implementation*, In Proc. *Cryptographic Hardware and Embedded Systems: CHES 2002*, volume 2523 of Lecture Notes in Computer Science, pp. 187-197, Springer-Verlag, 2002.
20. De Win, E., Bosselaers, A., Vandenberghe, S. De Gersem, P., Vandewalle, J.: *A fast software imlementation for arithmetic operations in* $GF(2^n)$. In Proc. *Advances in Cryptology: ASIACRYPT 96*, LNCS 1163, 65-76, Springer-Verlag, 1996.

Practical Template Attacks*

Christian Rechberger[1,2] and Elisabeth Oswald[1,2]

[1] A-SIT Secure Information Technology Center – Austria
Inffeldgasse 16a, A-8010 Graz, Austria
[2] Institute for Applied Information Processing and Communciations (IAIK)
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria
{christian.rechberger,elisabeth.oswald}@iaik.tugraz.at

Abstract. Side-channel attacks pose a serious threat to implementations of cryptographic algorithms. In the pioneering article of Chari, Rao and Rohatgi, the general idea behind template attacks was introduced. Template attacks apply advanced statistical methods and can break implementations secure against other forms of side-channel attacks.
However, in our research it turned out that several details, which are essential to practical implementations of template attacks, still need to be answered. In this article we provide answers to open issues, such as how to select points of interest in an efficient way, or how to preprocess noisy data. In addition, we show the benefits of trial classifications and we point out that in practice so-called amplified template attacks have to be considered as a potential threat.

Keywords: Side-Channel Analysis, Template Attack, DPA, DEMA

1 Introduction

Devices performing cryptographic operations can be analyzed by various means. Traditional cryptanalysis looks at the relations between input and output data and the used keys. However, even if the implemented algorithms are secure from a cryptanalysis point of view, side-channel attacks pose a serious threat. Side-channel attacks are a subgroup of implementation attacks. Examples thereof are timing attacks [Koc96], power attacks like DPA or SPA [KJJ99], EM attacks [AARR02], error message attacks [Ble98,KPR03], or combinations of different sources [WT01,ARR03].

Traditional DPA/DEMA style attacks assume the following threat model: The secret key stored in the device is used to perform some cryptographic operations. The attacker monitors these operations using captured side-channel information like power consumption or electromagnetic emanation. The attack is successful if the used secret key can be reconstructed after a certain number of operations. If the number of operations is limited by the protocol used to initiate

* This work was supported by A-SIT and by the FWF "Investigations of Simple and Differential Power Analysis" project (P16110-N04).

these operations, the attacker has an upper bound on the number of operations he can observe.

If the operation, which leaks usable side-channel information, is executed just once, the threat model is different: The attacker has to reconstruct the secret key using a single trace of side-channel information. Besides protocol limitations, ephemeral keys can be the reason for such a constraint. Techniques like SPA or SEMA [QS01] are a general way to tackle this problem. These techniques use easily distinguishable features of operations like double and add, or add and multiply, to infer key-bits. The majority of the available literature deals with these two types of scenarios.

If the observed signal-to-noise ratio is not high enough, or the implementation is done in a way that ensures the used operations being independent of the key(*i. e.* no key-dependent jumps), SPA/SEMA style attacks are not possible anymore. The attacker has to think of other ways to get hold of the secret key: One way to do this is to use a similar device and build a model of it. Using this model, an attacker might now be able to recover the secret key. Only very few publications [BS99,FP99,CRR03] deal with this model.

A very general and powerful way to perform such a two-stage attack, called template attack, is given in [CRR03]. The key concept is to store the probability distribution of leaking information for each device state. Signal classification techniques are subsequently used to assign a captured trace from a device to one device state. This technique is computationally intensive, execution time and storage requirements are very high. The result is a reduced set of probable secret keys.

1.1 Contribution of This Work

In order to make template attacks more practical, two goals need to be considered. The first one is to decrease the computational requirements of the attack. Secondly, the size of the set of probable keys should be kept as low as possible, subsequently referred to as the accuracy of the attack. To reach these goals, we address issues in this article such as:

– **Trial Classification.** A separate step in the course of a template attack named *trial classification* is introduced in Section 3.1. We propose to use trial classifications to improve the accuracy of a template attack.
– **Points of Interest.** For a template attack to be practical, it is paramount that not all points of a trace are part of the template. To reduce the number of points, a standard technique called principal component analysis could be used. Due to the high computational requirements of this technique, we propose a much simpler and faster approach in Section 3.2. Additionally, we suggest several properties of the selection of points of interest. Hence the algorithm is improved even further.
– **Preprocessing.** We introduce a *preprocessing step* in Section 3.3. We show that the use of discrete Fourier transformation on traces significantly improves attack results in practice. This is illustrated using side-channel information from power consumption and EM emanation. In Section 4, we give

evidence that in cases where too much data independent (ambient) noise in the input data renders a template attack impossible in the *time domain*, a transformation of input traces into the *frequency domain* is highly advantageous.

– **Amplified Template Attacks.** By extending the threat model, the classification results of template attacks can be increased. This approach is especially useful when the number of allowed iterations of critical operations is not high enough to allow a DPA style attack.

The remainder of the paper is organized as follows: The next section revisits template attacks and covers the ideas behind it. Requirements of an attack to be practical are stated. Section 3 proposes several means to meat these requirements. In Section 4 we illustrate the power of our proposals considering a concrete example.

2 Template Attacks

The method used in the template attack is as follows: First a strategy to build a model of all possible operations is carried out. The term "all possible operations" usually refers to the cryptographic algorithm (or a part of it) being executed using all possible key values.

The model of captured side-channel information for one operation is called *template* and consists of information about the typical signal to expect as well as a noise-characterization for that particular case. After that, the attack is started and the trace of a single operation (*e. g.* the key-scheduling of an algorithm) is captured. Using the templates created before which represent all key-values, the side-channel information of the attacked device is classified and assigned to one or more of the templates. The goal is to significantly reduce the number of possible keys or even come up with *the* used key.

However, this idealized approach faces several real-world problems. To lay the foundations for a successful attack, these problems are dealt with now.

2.1 Approaching the Template Attack

Assuming the captured trace consists of b sampled points and each sampled point consists of a signal and a noise part, one gets a b-dimensional noise-vector per trace. In general, all elements of this vector are drawn from different unknown probability distributions. A key point here is that the data-dependance of the noise is reflected by the probability distribution it is coming from. The challenge of modeling this noise vector can for example be simplified by assuming a normal distribution for each element and linear combinations thereof. This leads to the well-known multivariate-gaussian noise model. Simpler noise models like univariate models have been shown to be inadequate for practical purposes [CRR03].

2.2 Building Templates

In order to classify a trace, a template for each of the possible operations has to be built. The template reflects the statistical properties of such a trace, or in other words, the properties of the probability distribution of all its points. Using the multivariate gaussian model, a template consists of a vector of means and a matrix of covariances. To create such a template, a number of traces have to be captured; more traces lead to a more accurate model.

Let us assume that n different operations need to be distinguished and for every operation a number of p traces named t_1, .. t_p have been captured. Subsequently, we consider just one out of n operations.

The vector of means can be calculated as follows:

$$\overline{t} = M = \frac{1}{p}\sum_{j=1}^{p} t_j \tag{1}$$

Keep in mind, that all traces t_j are taken from the same operation. The next step is to calculate the noise vectors. For every trace t_j of a operation, the corresponding noise vectors N_j are $t_j - M$. They are the basis for the covariance matrix of the noise, which is the second part of the template. But first we define the covariance, which is sometimes referred to as a means to measure the linear dependance between two random variables. The empirical covariance of two random variables X and Y is defined as

$$\mathrm{cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{2}$$

In order to describe the covariances between more than two random variables, a matrix of covariances is needed. In our case, the entries in the noise covariance matrix can be defined as:

$$CM(u,v) = \mathrm{cov}(N_u, N_v) \tag{3}$$

Hence the covariance of all pairs of noise vectors is included. Note that the covariance matrix is symmetric and that elements on the diagonal of the covariance represent the variance for that column.

For each of the n possible operations, the corresponding template is the tuple (M, CM).

2.3 Classifying Traces

Once all templates are derived using a programmable device, they can be used to classify a trace t captured from the device under attack. Using the maximum-likelihood principle, the goal is to find the template, that "fits" best to the captured trace. The following steps have to be done to accomplish this.

1. For each template (which represents a certain key value), compute the noise N_i of the trace as if the device under attack was using the key value the template is associated to. M_i refers to the vector of means of the used template.

$$N_i = M_i - t \tag{4}$$

2. The probabilities of observing this n-dimensional noise vector N_i can be calculated using the noise covariance matrix of each template. Assuming that every point of the trace is drawn from a gaussian shaped probability distribution, the formula for the n-dimensional multivariate gaussian probability distribution can be used:

$$p(N_i) = \frac{1}{\sqrt{(2\pi)^n |CM_i|}} \cdot e^{-\frac{1}{2} N_i^T CM_i^{-1} N_i} \tag{5}$$

If the assumption of a gaussian shaped noise holds, then the maximum likelihood approach of selecting that template with the highest probability of observing the calculated noise is optimal.

2.4 Requirements for an Attack in Practice

In this subsection we focus on some properties of the template attack, which can render it impractical if they are not considered properly.

Classification Performance. It is not possible to perform the classification process on the entire key space at once, since building that many templates is not feasible. The iterative approach to cope with this problem, called extend and prune strategy, is explained in [CRR03].

The goal is to keep the number of possible keys which remain after the pruning step small at every iteration. More efficient classification performance results in better pruning steps. Depending on the power of the classification, or their *accuracy*, the number of required templates explodes more-or-less towards the end of the attack. For a successful attack, it is sufficient that the exhaustive search on the remaining possible keys is feasible.

Trace Length. Depending on the measurement setup and the data acquisition strategy, captured traces can be quite big (*i. e.* the number of sampled points is high). This has several implications for the attack:

– In the multivariate model, the covariance matrix has the highest storage requirements. The size of a covariance matrix depends on the number of points considered. Hence, the memory requirements of the templates grow quadratically with the number of points.
– Calculating the observation probability involves a matrix inversion. As a consequence, the running time of the device characterization step grows nearly cubically (depending on the used algorithm for matrix inversion) with the number of points.

Especially the latter observation can have devastating effects on the practicality of an attack. Consequently a general way has to be found to deal with this problem. The goal is as follows:

Having a trace of size N, select those $n < N$ points of the trace, which provide the most information on the used key.

Subsequently these n points are referred to as *points of interest* or *selected points*. Choosing a reasonable number (which is explained subsequently) always leads to template sizes significantly smaller than the size of one trace. For a template attack to be practical it is paramount that n is small enough to allow a fast calculation of observation probabilities. Our efficient way to tackle this problem is shown in Section 3.2.

The classification performance is equally important, since it directly affects the pruning step which in turn limits the number of templates to be built. In Section 3.3 we propose a novel way to increase classification performance in practice.

3 Our Way to Make Template Attacks More Practical

An RC4 [Sch96] implementation on an 8-bit microcontroller is considered subsequently. Side-channel leakage in the form of power consumption or EM emanation of the initialization part of RC4, also known as the key scheduling, is used to identify the used key. A more detailed description of the attacked implementation and the measurement setup can be found in Appendix A.

3.1 Trial Classification

We introduce an extension to the general two-stage model: The programmable device in the first step of the attack can not only be used to build templates but also to perform trial classifications in order to obtain good parameters for the actual attack. Note that this step is possible for every template attack and significantly improves results. Following this approach, the efficiency of building templates carries even more weight.

3.2 Efficiently Choosing Points of Interest

In order to find those n out of all N points providing the most information for a template, principal component analysis [Jol02] (PCA) can be used. PCA is mainly used in multivariate statistics to reduce the dimensionality of a data set. The procedure rotates existing axes to new positions such that maximum variabilities are projected onto the axes. This way, the most interesting features of the data-set are preserved. It was applied to similar problems in side-channel analysis as well [BNSQ03]. Adapted to our problem, this approach delivers a ranking of the points in our traces concerning their dependance on keys. However, this comes at the price of high computational requirements. Therefore we suggest the following simpler and more efficient strategy:

– Take the vectors of means calculated while building the templates.
– Compute differences of each pair of mean vectors and sum them up.
– Select n point among the highest peaks.

Note that the last step needs to be done, even if PCA is used. For each trial classification, the process of choosing points of interest needs to be repeated.

Hence any improvement in this step has an even greater impact on the practicality of an attack. We achieved a speedup factor 1000 for a typical setting, without affecting the classification performance.

In Figure 1, the sum of differences trace for one round of the RC4 key-schedule is plotted. Several distinct sets of traces (each containing 100 traces of $100\mu s$ length) were used as input. The area between the points $40\mu s$ and $70\mu s$ is clearly visible as the most interesting part of the plot, since the rest of the trace is not dependent on the key.

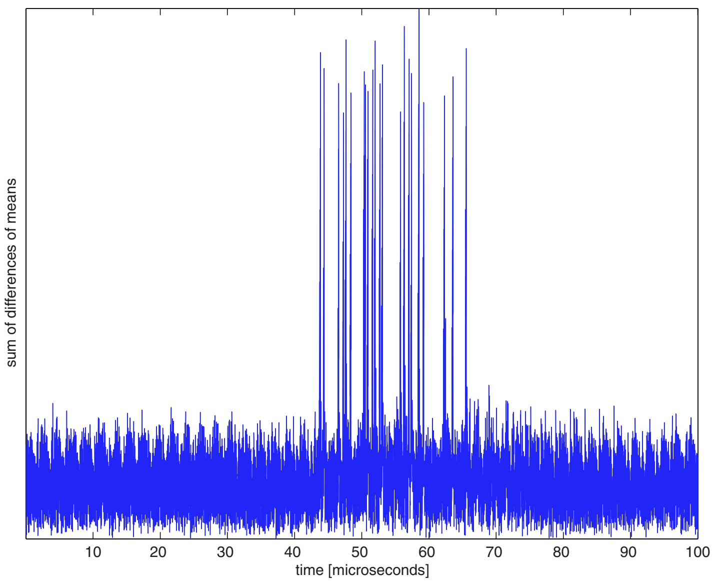


Fig. 1. Sum of differences trace.

Choosing the n highest points to serve as the points of interest has been proven to be non-optimal during our experiments. A more efficient strategy is possible if some properties of good selections are known in advance. We propose, that selected points should have the following properties:

- The minimum distance between these points should be approximately a clock cycle or more, since additional points in the same clock cycle do not provide additional information.
- The minimal height of a selected point should be higher than the noise floor of the sum of differences trace.

An algorithm for choosing the n highest peaks that follows these constraints was used in all our subsequent experiments. Disobeying these constraints leads to poor classification performance even if a higher number of points of interest is chosen.

To find a good trade-off between these parameters, trial classifications should be used. To highlight the influence of the number of selected points on the

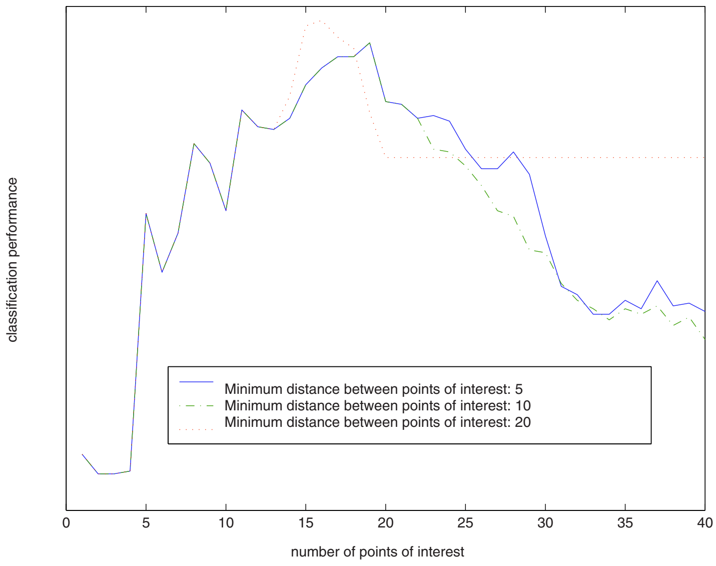


Fig. 2. Performance as a function of the number of selected points.

classification performance, a number of experiments were carried out. A template attack on a small part of the used key was performed with different numbers of points of interest. Figure 2 shows the classifications performance as a function of the number of selected points. The considered numbers of points range from 1 to 40. Additionally three different minimum distances were chosen. Results show that 15–20 is a good number of points of interest. If the minimum distance was chosen to be 20, not more than 20 points of interest were found above a level of 40% of the highest peak. The level starting from 20 is indicating this fact.

Figure 3 gives another point of view processing the same behavior. For each graph, the number of points of interest was kept constant and the minimal distance between these points was varied between 10 and 500 ns. Even though the graph seems to be chaotic, one conclusion can be drawn: The minimal distance is this attacking setup should be at least 50.

Even though the given values depend on the implementation and can not be used directly for a template attack on another implementation, the outlined strategy can be used for other implementations. Additionally, our results lead to the conclusion, that classification results are declining if the number of points of interest gets too high.

3.3 Preprocessing

In practical side-channel analysis, the raw input data is often preprocessed. Sometimes this is just due to simplicity or efficiency reasons, *e. g.* summarizing sampled points. There are however cases where the preprocessing step heavily affects the results. Even if no thinkable transformation can add additional information to a signal, information extraction procedures do improve. The template

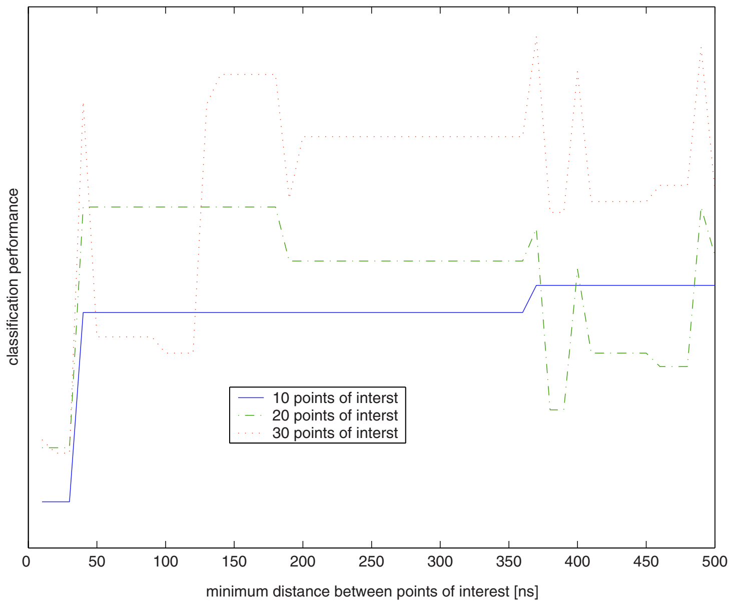


Fig. 3. Performance as a function of the minimum distance.

attack under consideration is such a case and a lucrative preprocessing transformation is described subsequently.

It turns out that the transformation of the input traces from the time domain into the frequency domain is such a lucrative transformation. In our practical work, an FFT algorithm was used to accomplish this transformation (a fast algorithm to calculate the discrete Fourier transform, for background information refer to [BP85]). In order to show the impact of this preprocessing step a number of experiments were carried out. First some characteristic differences between time domain analysis and frequency domain analysis are illustrated. Afterwards, to highlight the influence of the number of selected points on the classification performance in the frequency domain, a number of experiments were carried out.

Figure 4 contrasts both possibilities. It shows the power trace of one round of the RC4 key schedule in the time domain (ranges from 0–100μs) and in the frequency domain(ranges from 0–50MHz). The transformed input can also be interpreted as the frequency spectrum. The regions with the highest peaks indicate the clocking frequency and multiples of it. To amplify lower peaks, the vertical axis is logarithmical. In Figure 5, the sum of differences of means trace is plotted. Naturally, the key dependent areas are also in the range of the clocking frequency and multiples thereof.

After preprocessing, the resulting traces can be used to perform a template attack in exact the same way as without preprocessing. There is however a difference in the number of points to consider. Figure 6 shows the classifications results as a function of the number of selected points after preprocessing. The considered numbers of points are ranging between 1 and 40. Additionally three different minimum distances where chosen. Results show that much less points are sufficient in comparison to a template attack without the preprocessing step.

At the price of performing an FFT on every input trace (those used to build up the templates as well as those to classify) we get a major advantage. The

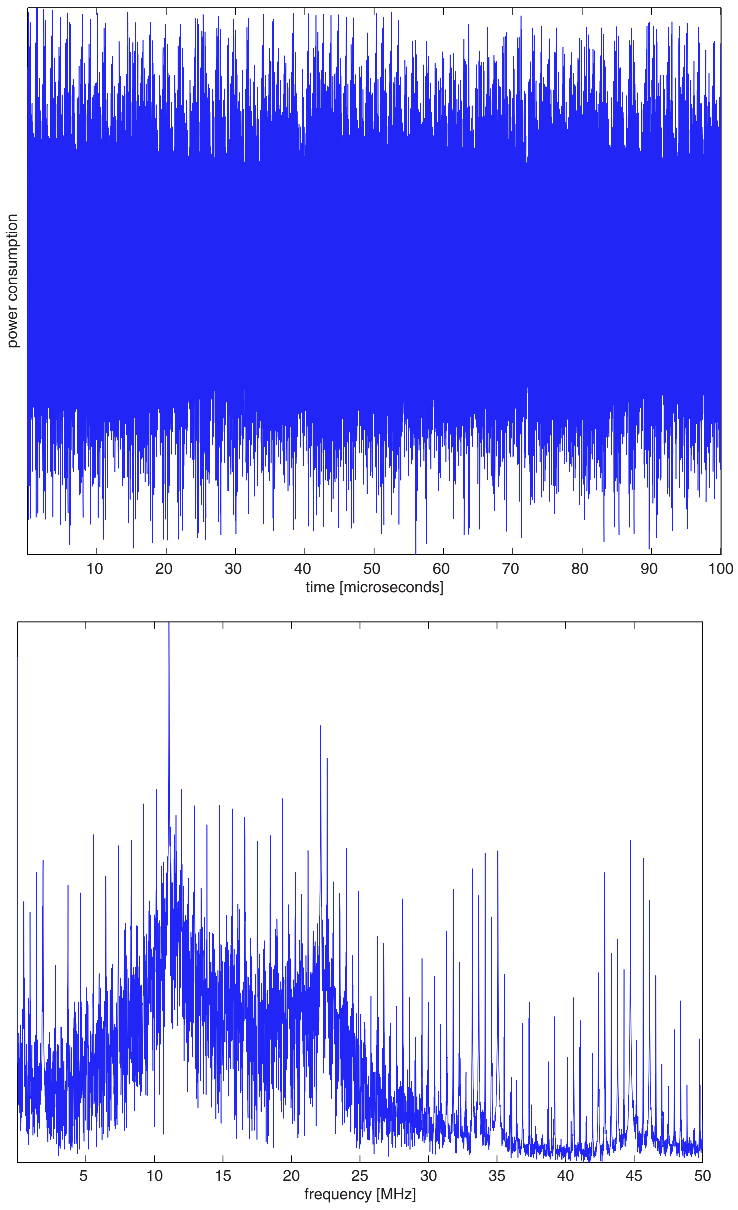


Fig. 4. Power trace of one round of the RC4 key schedule in the time domain (0–100μs) and in the frequency domain(0–50MHz).

template classification can be done much faster and storage requirements are lower, since the size of n influences those requirements as outlined in Section 2.4.

Attention has to be paid that considerations on the lower bound for selected peaks – as done in the time domain – are not directly applicable. In practice the lower bound is much smaller in the frequency domain.

3.4 Amplified Template Attack

Even if the aim of a template attack is to recover the secret key using a single trace, in many real world settings implementations allow for several iterations of the same operation with the same secret key. The application of template attacks is not restricted to stream ciphers like RC4 and can be applied to block ciphers as well. Since every symmetric cipher contains some sort of key scheduling mechanism which processes the secret key, this generalization is possible. Smartcards often use block ciphers for encryption or authentication, hence let us consider the following example: A malicious petrol station tenant, named Eve, is using a modified smartcard based payment terminal. Everytime a customer uses this terminal, Eve captures one trace of side-channel information. This single trace could already be used by Eve to carry out a template attack.

However, some customers are coming again and Eve gets hold of another trace. The template attack can easily be extended to take advantage of such situations, e.g. by adding up noise-probabilities $p(N_i)$ of every captured trace and applying the maximum-likelihood approach on these sums. As a consequence, the power of the attacker is *amplified*. Using this approach, if n is the number

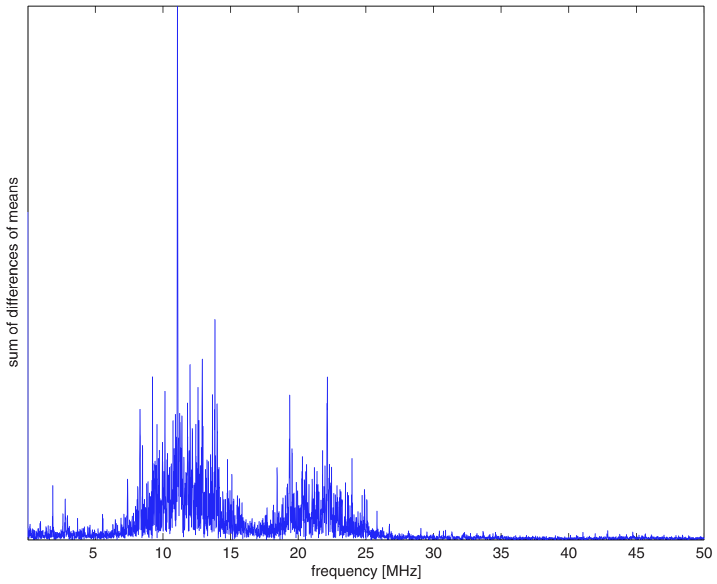


Fig. 5. Sum of differences of means trace for the preprocessed inputs).

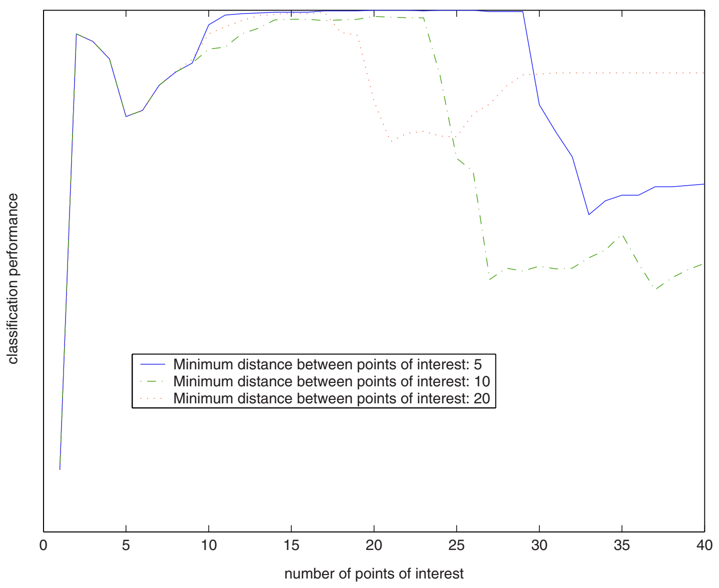


Fig. 6. Performance as a function of the number of selected points.

of iterations, the error probability of template classification is reduced by the factor $\sqrt{n}$. We have experimentally verified this improvement.

This improvement illustrates several important advantages over DPA style attacks. The first one is a property of the template attack: there is no need to know input or output data of the used algorithm. Additionally, in cases where too few traces are available to facilitate a DPA style attack, template attacks can take advantage of every single trace using the amplified template attack.

4 Applying the Improvements in Practice

We give an example where a classification is not possible at all without preprocessing in Section 4.1. Thereafter, in Section 4.2, we show that with the same measurement setup and with preprocessing, the template attack is successful. The measurement setup used in both scenarios is described in Appendix A.

4.1 Template Attack Without Preprocessing

If measurements of the power consumption of the attacked device are not possible for an attacker, EM probes are the next alternative. Even if information about the power consumption is available for the attacker it can be advantageous to consider EM channels additionally [ARR03].

Compared to power measurements, EM measurements are much noisier. Even if the character of the noise is an important part of the built templates, ambient noise originating outside the device is of no value for the attack. In order to highlight the difference we call this noise data independent noise. As in Section 3.2,

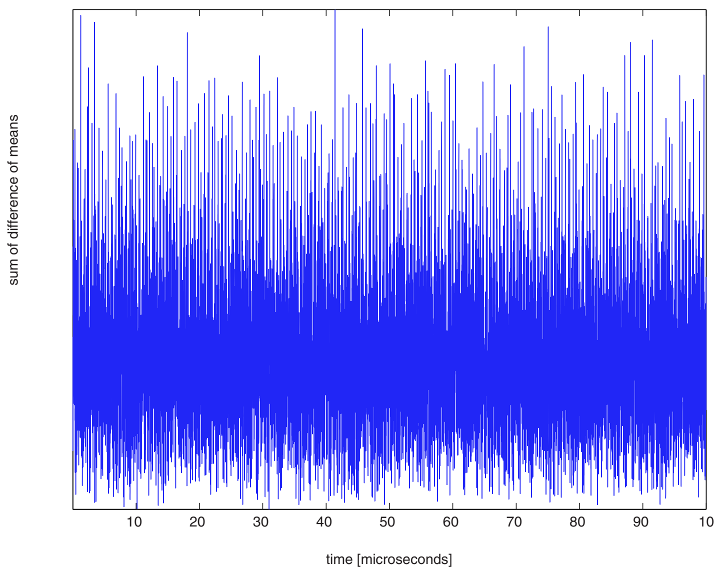


Fig. 7. Sum of differences of means trace for captured EM traces.

in a first step we capture traces while using different key values and sum up their pairwise differences. The result is depicted in Figure 7. Compared to Figure 1 there are no key-dependant areas observable anymore. Subsequent classification trials delivered no usable result.

If data independent noise is superposing the signal originating from the device, averaging techniques have to be used to eliminate it. However this can be quite cumbersome. In our particular setup, increasing the number of needed traces to build a template from 100 (which was sufficient for power traces) to 1000 did not change the bad classification results. Increasing the number of traces even more might lead to better results at some point, but on the other hand it would render the actual template attack for the complete keyspace impractical.

4.2 Template Attack with Preprocessing

Using the preprocessing step explained in Section 3.3, increasing the number of traces is however not necessary and still leads to much better classification results. In this case the error probability was 2% on average. The resulting sum of differences of means trace is depicted in Figure 8. Key dependent frequency areas are clearly visible.

There are several similarities to the experiment using power traces as input as shown in Figure 5. Again the sum of differences trace is reflecting the spectral density of the original power traces. However the highest peaks are not located at the clock frequency, but at some multiples of it.

Figure 9 shows the performance of our approach as a function of the number of selected points of interest. Compared to the similar experiment on traces gained from power measurements (see Figure 6) a much bigger number of selected points is needed.

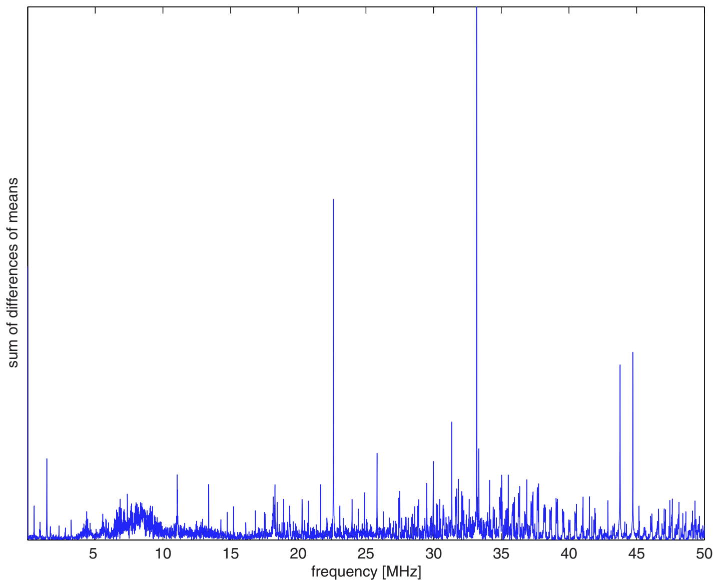


Fig. 8. Sum of differences of means trace for the preprocessed EM traces.

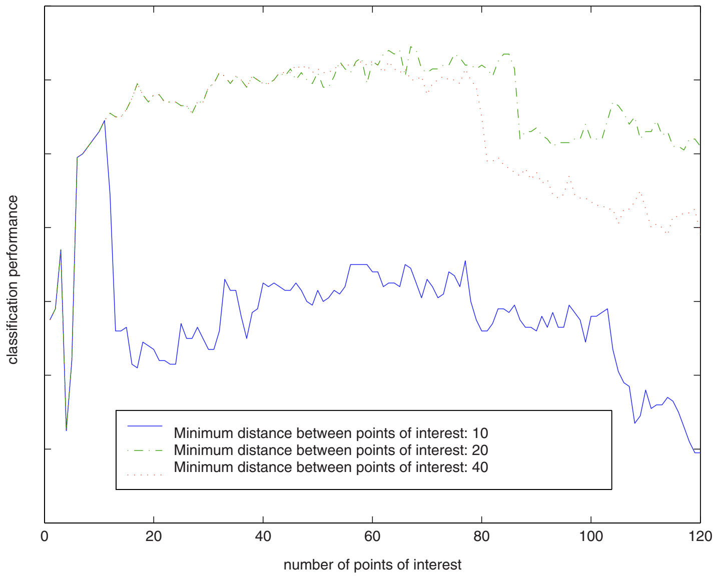


Fig. 9. Performance as a function of the number of selected points.

In contrast to the power measurements, there is a distinctive leap considering different minimal distances between selected points. This can be seen in Figure 10, where the performance of the sum of differences of means method for preprocessed EM traces is plotted as a function of the minimum distance between two points of interest. Similar to the time domain results for power

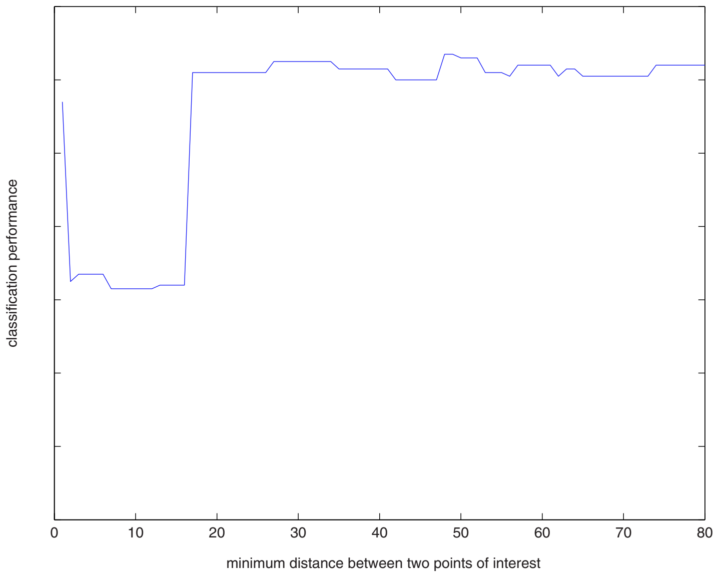


Fig. 10. Performance as a function of the minimum distance.

measurements, choosing points within a certain distance in frequency does not help classification. The slight variations of classification performance after the leap are due to the selection algorithm and no additional implications result from them. They result from choosing slightly different peaks in the same frequency areas in order to meet the minimum distance constraints.

5 Conclusion

Starting from the pioneering work of [CRR03], where several important questions concerning the implementation of template attacks are left open, we proposed several improvements.

First, we outlined a way to perform a template attack in more detail: In order to find optimal parameters for the attack like number and position of points of interest, a first step named trial classification was introduced. Using this trial classification, we ensured the practicality of the attack by introducing a efficient algorithm that replaces principal component analysis. We increased the speed of selecting points of interest by a factor of 1000 without impairing the classification performance.

Additionally, we introduced the preprocessing step. A transformation of traces from the time domain into the frequency domain is suggested. This improved classification results and lowered runtime and storage requirements.

Finally, the power of our proposals is illustrated on a concrete example. Due to much ambient noise, classification is not possible at all without preprocessing in this example. Using our proposed preprocessing technique combined with trial classifications however proved to be very valuable and resulted in practical classification probabilities.

References

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Proceedings of CHES 2002*, volume 2535 of *LNCS*, pages 29–45. Springer, 2002.

[ARR03] Dakshi Agrawal, Josyula R. Rao, and Pankaj Rohatgi. Multi-channel Attacks. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Proceedings of CHES 2003*, volume 2779 of *LNCS*, pages 2–16. Springer, 2003.

[Ble98] Daniel Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, number 1462 in LNCS, pages 1–12. Springer, 1998.

[BNSQ03] Lilian Bohy, Michael Neve, David Samyde, and Jean-Jacques Quisquater. Principal and Independent Component Analysis for Crypto-systems with Hardware Unmasked Units. In *Proceedings of e-Smart 2003*, 2003.

[BP85] CS S. Burrus and Thomas W. Parks. *DFT/FFT and Convolution Algorithms and Implementation*. John Wiley & Sons, 1985.

[BS99] Eli Biham and Adi Shamir. Power Analysis of the Key Scheduling of the AES Candidates. In *Second Advanced Encryption Standard (AES) Candidate Conference*, Rome, Italy, 1999.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Proceedings of CHES 2002*, volume 2535 of *LNCS*, pages 13–28. Springer, 2003.

[FP99] Paul N. Fahn and Peter K. Pearson. IPA: A New Class of Power Attacks. In Çetin Kaya Koç and Christof Paar, editors, *Proceedings of CHES'99*, volume 1717 of *LNCS*, pages 173–186. Springer, 1999.

[Jol02] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[KPR03] Vlastimil Klima, Ondrej Pokorny, and Tomas Rosa. Attacking RSA-Based Sessions in SSL/TLS. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Proceedings of CHES 2003*, volume 2779 of *LNCS*, pages 426–440. Springer, 2003.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Isabelle Attali and Thomas P. Jensen, editors, *Proceedings of Smart Card Programming and Security*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.

[Sch96] Bruce Schneier. *Applied Cryptography, 2nd Edition*. John Wiley & Sons, 1996.

[WT01] Colin D. Walter and Susan Thompson. Distinguishing Exponent Digits by Observing Modular Subtractions. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *LNCS*, pages 192–207. Springer, 2001.

A Attack Implementation and Measurement Setup

For our practical experiments on the template attack, we used RC4. The algorithm was implemented on an ATMEL AT89S8252. This 8 bit microcontroller is operated on a clock frequency of 11.059MHz. For data acquisition we used the digital oscilloscope Lecroy LC584AM, having a resolution of 8 bit. In most of our experiments we used a sampling rate of 100 MSamples/s.

RC4 specifies a whole family of algorithms whose differences lie in the used word-size n; typically the word-size is 8. The algorithm consists of two parts which are executed sequentially:

- An initialization phase or KSA (Key Scheduling Algorithm) with 2^n rounds
- An output phase or PRGA (Pseudo Random Generation Algorithm) outputting one word per round.

Both parts access an internal table of size $n * 2^n$ bits. In order to keep our implementation of RC4 completely inside the internal RAM, the word size was reduced to 7 bit. Read/write access to external RAM resources would have leaked too much side-channel information and would therefore have simplified the attack unrealistically. For our template attack on RC4, we just considered the KSA.

Figure 11 depicts our measurement setup. It shows our evaluation board with the ATMEL microcontroller, the EM probe used for our EM measurements and the differential probe used for our power measurements.

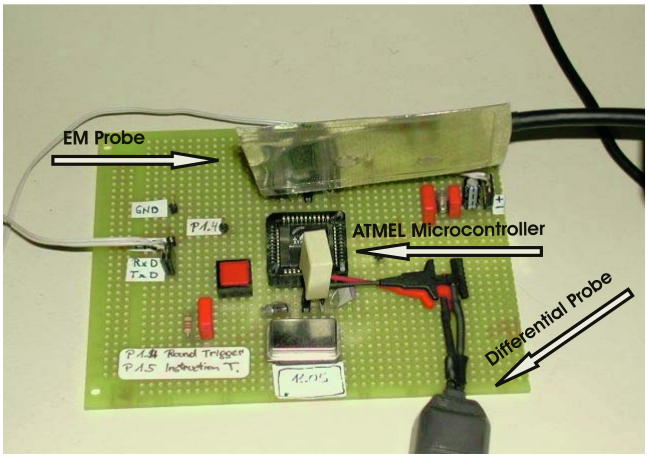


Fig. 11. Measurement setup.

Evaluation and Improvement of the Tempest Fonts

Hidema Tanaka, Osamu Takizawa, and Akihiro Yamamura

National Institute of Information and Communications Technology,
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
{hidema,taki,aki}@nict.go.jp

Abstract. Information leakage via electromagnetic emanation, usually known as Tempest, has been recognized as a threat and countermeasures have been proposed. In particular, Kuhn and Anderson developed a protective measure for Tempest called the *Tempest fonts*. Through experiments, we have investigated and evaluated the effectiveness of the Tempest fonts. Furthermore, we propose a new measure based on a similar approach to prevent successful Tempest attacks. While Kuhn and Anderson use Fourier transformation as a low-pass filter, our approach uses a Gaussian filter in addition to Fourier transformation. Our experimental results show that this approach is more effective.

1 Introduction

The unintentional emanation of physical energy is a major threat to privacy. Adversaries can eavesdrop on sensitive information via electromagnetic emanation from computers or their peripherals. Tempest refers to the techniques, investigations, and studies of compromising emanations and their application to eavesdropping, as well as to the information leakage through emanations. Tempest has been a concern regarding computer security in military and governmental institutions for a long time; however, much of the information gathered on Tempest technologies has not been publicly disclosed.

Several Tempest test receivers are now available to non governmental institutions, and a few researchers have published details of their experiments [1, 2, 4, 8, 10]. These papers have verified that Tempest is a potential problem and warn that it is a serious issue. Kuhn and Anderson [6, 8] have proposed a countermeasure; the use of a filtered fonts with spectral characteristics. They claim that their fonts, called the *Tempest fonts*, significantly reduces the effective range of eavesdropping at a negligible cost in image quality and prevents adversaries obtaining on-screen information.

In this paper, we report on our experiments on the reconstruction of images containing text written in the Tempest fonts and verify the effectiveness in a particular situation. We also show that the fonts do not provide sufficient security in certain attack models where the adversary uses sophisticated equipment. Furthermore, we propose and discuss an alternative to the Tempest fonts. Kuhn and Anderson use only Fourier transformation as a low-pass filter. We use a

Gaussian filter in addition to Fourier transformation. The reader is referred to [5] for more detail on Fourier transformation and Gaussian filters.

In the following sections, we discuss our experiments, compare the two approaches, and show that our method is more effective in certain cases.

2 Tempest Fonts

Kuhn and Anderson [6, 8] performed experiments on recovering PC screen images using an ESL model 400 Tempest monitoring receiver and a dipole antenna. They developed the Tempest fonts to protect privacy from the Tempest threat, and their Tempest fonts package can be downloaded at [7]. The fonts contained in the package is a filtered and anti-aliased version of the Courier font.

Kuhn and Anderson claim that the Tempest fonts provide adequate security in a certain situation at less cost than preparing perfectly shielded devices and peripherals.

3 Outline of Experiments

3.1 Equipment

In our experiments, we used an FSET22 receiver and FrameControl ver. 4.24 image processing software. The FSET22 specifications are shown in Table 1. FrameControl can process the signal from the FSET22 at 256 frames/3 s. We used an Anritsu MP666A logarithm periodic antenna (20 $\sim$ 2000 MHz), an Anritsu MA2601B/C near magnetic field probe (5 $\sim$ 1000 MHz) and a TOKIN EIP-100 injection probe (80 KHz $\sim$ 30 MHz).

The effectiveness of image processing is especially important. Our image processing software could create an averaged image from up to 256 frames. The software we used has almost the same capabilities as Adobe Photoshop, and it works in real time. Note that our equipment is not classified and can be purchased from a commercial firm.

3.2 Target Machines

The targets in our experiments were an IBM ThinkPad S30 notebook PC (denoted as IBM), a SONY VAIO PCG-V505 notebook PC (denoted as VAIO), and a NANAO FlexScan 77F 21-inch CRT (denoted as CRT). The CRT was connected to the VGA connector of the VAIO. We tried to reconstruct the display image of each target, whose screen displayed the cur-13x24m-a.gif character table (Fig. 1).

3.3 Attack Scenarios and Experimental Design

We performed three experiments corresponding to attacks using a near magnetic field probe, an antenna, or an injection probe as follows.

Table 1. FSET22 Specifications.

Frequency range	100 Hz $\sim$ 22 GHz
Frequency resolution	0.1 Hz
Bandwidth	10 Hz $\sim$ 500 MHz in steps of 1/2/5
Average noise level	< -142 dBm (1 MHz)

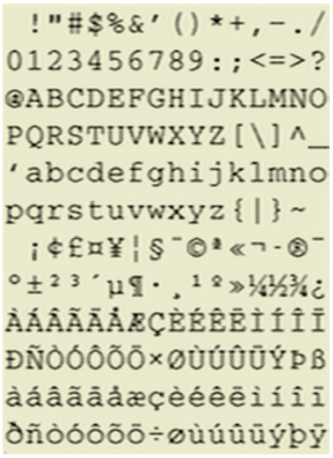


Fig. 1. Tempest fonts (cur-13x24m-a.gif).

1. eavesdroppers embed a near magnetic field probe in the vicinity of the target,
2. eavesdroppers try to catch emanation outside of the room in which the target machine is located,
3. eavesdroppers try to receive signals transmitted by the power supply line.

The results of each experiment are described in Sections 4.1, 4.2, and 4.3, respectively.

3.4 Tempest Procedure

Knowing the synchronous frequency of a target significantly helps an eavesdropper reconstruct the display image via electromagnetic emanation. An eavesdropper who does not know the frequency can easily guess it from the standard parameters of the Video Electronics Standards Association (VESA) [12]. In this work, we measured the synchronous frequency of each target in the experiments using a near magnetic field probe (see Section 4.1). The experimental procedure was as follows.

[Step 1] Search for the source location of the electromagnetic wave emission.
[Step 2] Adjust the parameters for the received frequency.
[Step 3] Adjust the parameters for the synchronous frequency.
[Step 4] Apply image processing.

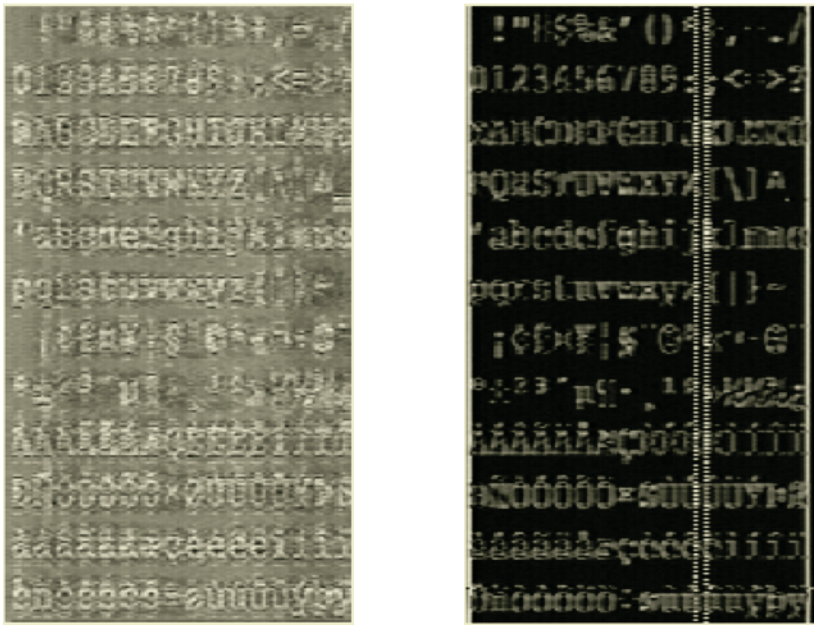


Fig. 2. Reconstructed images: (left) IBM, (right) CRT.

We repeated these steps until we got a clear reconstruction of the target display. Measurement of the synchronous frequency using a probe is very important in Tempest. Our instruments were accurate to within six figures below a decimal point with regard to both the horizontal and vertical frequencies of the VGA signal. Since the synchronous frequency is a unique value for each PC, if we know the correct parameters for the synchronous frequency, we do not have to carry out Step 3. We show both the horizontal and vertical synchronous frequencies for each of our targets in Table 2. Since the CRT was connected to the VAIO VGA connector, the synchronous frequencies of the CRT and VAIO were identical.

4 Experiments

4.1 Experimental Results: Near Magnetic Field Probe

In this experiment, we placed the near magnetic field probe very close to the targets (the IBM, VAIO, and CRT). Reconstructed images (for the IBM and CRT) obtained by averaging 128 frames are shown in Fig. 2. The reconstructed image for the VAIO was almost the same as that for the IBM, so we do not show it in Fig. 2. The parameter values that provided the best results for the IBM and CRT are listed in Table 3. We can distinguish many characters in the Tempest fonts from the images in Fig. 2; this led us to conclude that we can obtain the semantics of text (natural language) written in the Tempest fonts.

Table 2. Synchronous frequency of each target.

	Horizontal frequency [MHz]	Vertical frequency [KHz]
IBM	48.530874	60.211965
VAIO	48.338321	59.973150
CRT	48.338321	59.973150

Table 3. Parameter values used to reconstruct images using the near magnetic field probe.

	Frequency [MHz]	Bandwidth [MHz]
IBM	461.2	20.0
CRT	57.4	20.0

4.2 Experimental Results: Antenna

A Tempest attack using an antenna is considered a more realistic attack than one using a near magnetic field probe. In our experiment, the antenna was placed 4 m from the target. Since we could not successfully reconstruct the IBM image, we show only the VAIO and CRT results in Fig. 3. The parameters in this case are shown in Table 4.

4.3 Experimental Results: Injection Probe

Tempest attacks using injection probes are more realistic still than Tempest attacks using an antenna. An injection probe looks like a clothespin and can be clipped onto a cable to receive electromagnetic emanation. The attenuation of the emanation of an electromagnetic wave depends on the distance from the target, and is very low for conduction emanation from a power supply line compared with direct emanation from the target to air. In this experiment, we set the probe 30 cm from the target and on an extended cable at over 30 m from the target. In both cases we succeeded in obtaining images from the CRT, and there was little difference between the reconstructed images. That is, we could read the Tempest font characters from the CRT (Table 5 and Fig. 4) as we could in the experiment using a near magnetic field probe (Section 4.1). On the other hand, we could not reconstruct the images at all from the IBM and VAIO. We believe that the reconstruction failed in these cases because of the AC adapter.

4.4 Conclusions Based on Experiment Results

We verified that character images written in the Tempest fonts are much more difficult to recover than those written in a normal font, such as a 10-point terminal font, when an attacker uses an obsolete Tempest receiver. On the other hand, the improvements made in Tempest receivers have greatly reduced the effectiveness of the Tempest fonts. We could clearly reconstruct character images by receiving electromagnetic emanation from the target on which the Tempest fonts was displayed in our experiments. Therefore, while the Tempest fonts can

Table 4. Parameter values used to reconstruct images using an antenna.

	Frequency [MHz]	Bandwidth [MHz]
VAIO	844.8	20.0
CRT	973.2	20.0

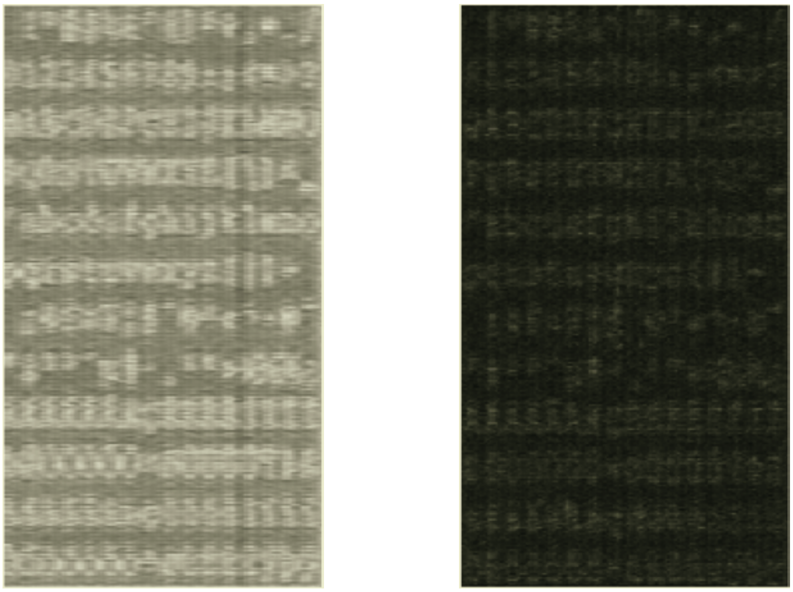


Fig. 3. Reconstructed image of Tempest font averaged from 128 frames obtained using an antenna at a distance of 4 m: (left) VAIO, (right) CRT.

prevent eavesdroppers from recovering character images under certain situations, such as when an adversary has only obsolete Tempest receivers, it is no longer an effective measure in more realistic environments.

Table 6 summarizes our experiment results. In Table 6, "TF" stands for the Tempest font, "Normal" stands for a 10-point terminal font (normal font), "readable" means we could recognize 80% of the characters, "nearly readable" means we could recognize 50% of the characters, and "non-readable" means that we could not recognize any characters. Here, our conclusions regarding readability are based on a subjective evaluation.

We found that reconstructing the Tempest fonts was easier than reconstructing the normal font when we used an antenna. Since the normal font consists of narrow lines, the high frequency spectrum generated from these lines creates sharp spikes on carrier waves. Such sharp spikes on electromagnetic waves are easily attenuated after emanating through air, making it difficult for us to distinguish the sharp spikes from air noise. Therefore, it was more difficult to reconstruct images written in the normal font in this case.

Table 5. Parameter values used to reconstruct images using an injection probe.

	Frequency [MHz]	Bandwidth [MHz]
CRT	23.8	20.0

Table 6. Summary of experiments.

	Near magnetic field probe		Antenna		Injection probe	
	TF	Normal	TF	Normal	TF	Normal
VAIO	readable	readable	readable	non-readable	non-readable	non-readable
IBM	readable	readable	non-readable	non-readable	non-readable	non-readable
CRT	readable	readable	nearly readable	non-readable	readable	readable

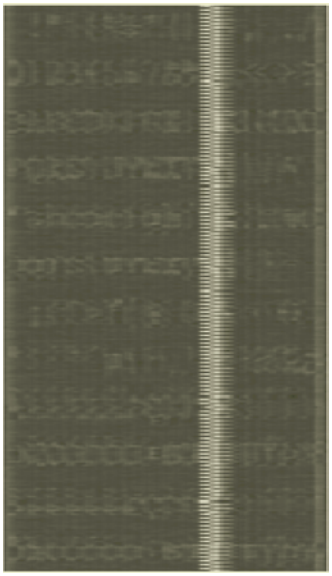


Fig. 4. Reconstructed image of Tempest font averaged from 128 frames obtained using an injection probe. The probe was set about 30 m from the CRT.

4.5 The Plausibility of Attacks

Radio frequency emanation decreases in proportion to the square of the distance from the target, and other radio frequencies from electric devices also interfere with radio frequency emanation. Therefore, we believe that an attack made by attempting to receive the radio frequency at a distance is unlikely to succeed.

On the other hand, power supply lines may extend into the building structure and eavesdroppers can use these lines to receive signals from anywhere in the same building. Hence, attacks by receiving the radio frequency through power supply lines is the most plausible threat.

From Table 6 and Figs. 2 $\sim$ 4, we see that the reality of an attack scenario (as explained in Section 3.3) is inversely proportional to the quality of the re-

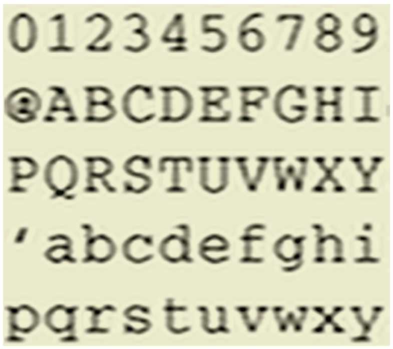


Fig. 5. Enlarged image of Tempest fonts.

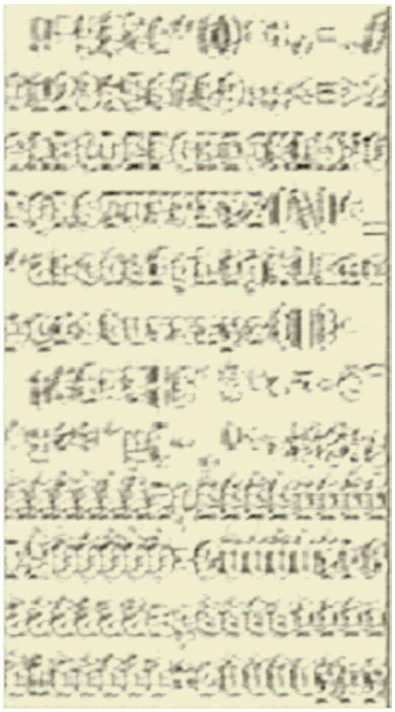


Fig. 6. Another processed reconstructed image of Tempest fonts. (We used the same image as in the left figure of Fig. 2.)

constructed images. An attack using near magnetic field probes is the most dangerous in terms of the reconstructed image quality, so countermeasures against Tempest attacks should be evaluated in terms of their effectiveness against attacks using near magnetic field probes. In Section 5, we evaluate an improvement made to the Tempest fonts in experiments where we used only near magnetic field probes.

5 Improvement of the Tempest Font

5.1 Basic Idea

The Tempest fonts is produced by applying Fourier transformation to the source font image and removing the top 30% of the horizontal frequency spectrum [7].

The high frequency spectrum creates large spikes in the carrier waves during the digital-analog conversion process. Such large spikes are easily distinguished from air noise. Therefore, the high frequency spectrum is valuable information that eavesdroppers can use to reconstruct the target display image. Because the high frequency spectrum provides eavesdroppers with a significant clue, removing it is an effective Tempest countermeasure. This is the basic idea underlying the Tempest fonts developed by Kuhn and Anderson.

5.2 Our Perspective

Through our experiments, we found another aspect of the Tempest fonts that makes recovery of the display image difficult. We explain our observation and this effect of the Tempest fonts in the following.

Figure 5 is an enlarged image of the Tempest fonts. We can recognize block noise (dither) around the characters in this figure. We applied another filter ("black/white reverse" and "edge emphasis": a function of FrameControl) to the reconstructed image of the Tempest fonts and emphasized the area around the character edges. The resulting image of the Tempest fonts can be perfectly reconstructed from the block noise. The block noise disrupts character forms that are made of curves, but not the character forms made of straight lines. Since the Tempest fonts produces block noise, it makes it easy to read characters made of straight lines but hard to read characters made of curves.

Although Kuhn and Anderson show the result that an eavesdropper completely failed to reconstruct images in [8], we were able to sufficiently reconstruct images from which we could read characters by reconstructing block noise. However, since the characters reconstructed from Tempest font are hard to recognize, we conclude that the Tempest font is an effective means of protecting privacy in certain situations.

5.3 Our Improvement

Since the generation of block noise reduces the resistance to Tempest, we propose a method to decrease block noise. Our fonts is produced from the Tempest fonts by applying a Gaussian filter. (See [5] for detailed information on Gaussian filters.) A Gaussian filter basically makes images flat and smooth. It does not lead to a high frequency spectrum because it causes a high correlation between adjacent pixels.

We had to find the best parameter values for the Gaussian filter because excessive values would substantially lower the display visibility. As parameters, we used a radius of 3.0 pixels and a threshold of 25.0 pixels. Figure 7 shows

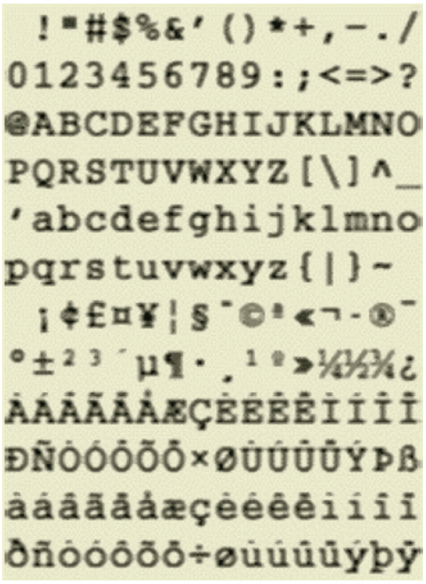


Fig. 7. Our filtered fonts.

our filtered fonts. Compared to the Tempest fonts (Fig. 1), our filtered fonts is grayer and characters like $\ll$ are harder to read.

5.4 Comparison and Discussion

We compare reconstructed images of the Tempest fonts and our filtered fonts (IBM, using a near magnetic field probe) in Fig. 8. Our fonts is clearly harder to read than the Tempest fonts. To confirm this subjective observation, though, we carried out the following experiment. We ask eight testees to look at Fig. 6 and identify as many characters as they could. The testees correctly identified almost 80% of all the characters in the Tempest font; in contrast, they made a correct identification of close to 0% for our filtered font. Furthermore, the number of attempted answers for our filtered font was 83% of that for the Tempest fonts. This experiment thus objectively confirmed our subjective evaluation.

Our experiments verified the effectiveness of the Tempest fonts in a certain situation (when an eavesdropper is using an obsolete receiver); however, the Tempest fonts is not a sufficient countermeasure against an eavesdropper using more sophisticated tools.

The characters of our filtered fonts, though, are dim, ambiguous, and weak in contrast. Thus, fine characters and symbols are hard to read. This makes our filtered fonts less user-friendly, especially for the elderly. Improved readability is needed for practical use.

We show an enlarged image of our filtered fonts in Fig. 9. From this image, we found that noise dots are uniformly scattered all over the background. Such noise dots are reconstructed in the image reconstruction process. These lower the

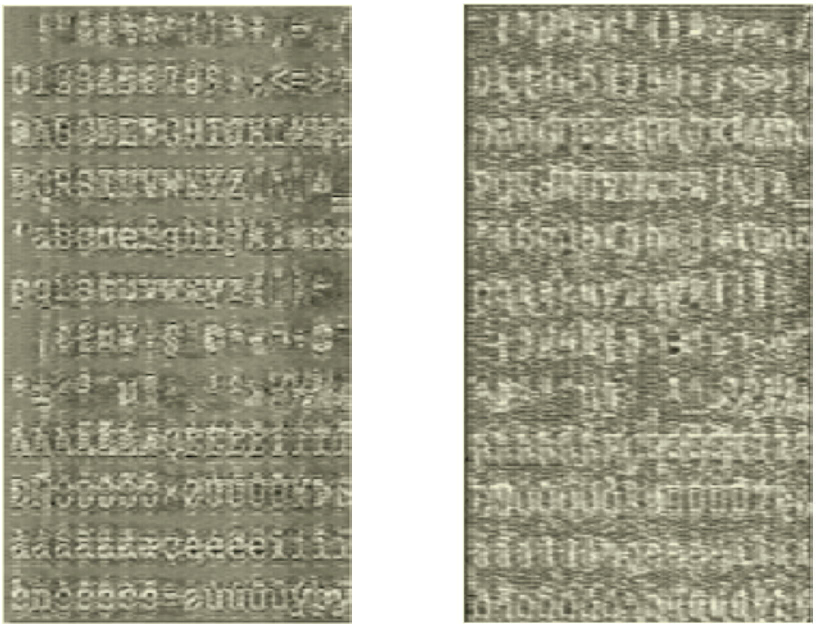


Fig. 8. Best reconstructed image of (left) Tempest fonts and (right) our filtered fonts. (IBM, near-magnetic-field probe).

readability of our filtered fonts in the reconstructed image, making it impossible to obtain a clear reconstructed image.

6 Conclusion

Hardware-based countermeasures against Tempest, such as shields, perform well but are inflexible and expensive. Software-based countermeasures, such as the Tempest fonts, allow the emanation of electromagnetic waves, but prevent eavesdroppers obtaining screen information via the received waves. Furthermore, these countermeasures are flexible and inexpensive. Therefore, software-based countermeasures are particularly promising.

The Tempest fonts was proposed as an inexpensive and easy to implement countermeasure against Tempest threats. Unfortunately, our results indicate that the Tempest fonts does not provide sufficient security in an attack scenario where the adversary possesses up-to-date Tempest receivers. Current Tempest receivers are unclassified and can be purchased by anybody. Removing 30% of the high-frequency spectrum is no longer an adequate security measure against eavesdroppers with such sophisticated receivers. In this paper, though, we have shown that the effectiveness of the Tempest fonts can be improved by applying a Gaussian filter.

It is important, however, to consider the trade-off between effectiveness against Tempest attacks and the font visibility. Also, the parameter values that

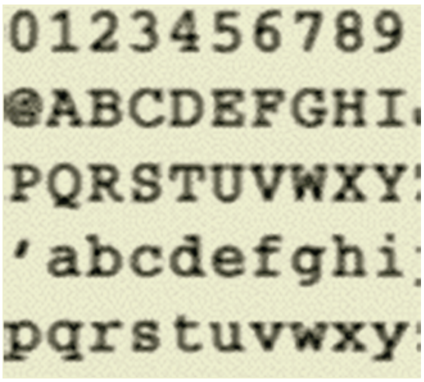


Fig. 9. Enlarged image of our filtered fonts.

should be used in the Fourier transformation and Gaussian filter differ depending on the type and size of the source font and the pertinent environment. Our future work will be aimed at solving these problems.

References

1. D. Agrawal, B. Archambeault, S. Chari and J.R. Rao, Advances in Side-Channel Cryptanalysis Electromagnetic Analysis and Template Attacks, RSA Laboratories Cryptobytes, Vol. 6, No. 1,Spring 2003, pp.20–32.
2. D. Agrawal, B. Archambeault, J.R. Rao and P. Rohatgi, The EM side channel(s), Cryptographic Hardware and Embedded Systems (CHES 2002), Lecture Notes in Computer Science, Vol. 2523, Springer-Verlag, 2002, pp.29–45.
3. S. Chari, J.R. Rao and P. Rohatgi, Tempest attacks, Cryptographic Hardware and Embedded Systems (CHES 2002), Lecture Notes in Computer Science, Vol. 2523, Springer-Verlag, 2002, pp.13–28.
4. K. Gandolfi, C. Mourtel and F. Oliver, Electromagnetic analysis: Concrete results, Cryptographic Hardware and Embedded Systems (CHES 2001), Lecture Notes in Computer Science, Vol. 2162, Springer-Verlag, 2001, pp.251–261.
5. R. Gonzalez and R. Woods, Digital Image Processing, Addison-Wesley Publishing Company, 1992.
6. M. Kuhn, Electromagnetic Eavesdropping Risks of Flat-Panel Displays, presented at the 4th Workshop on Privacy Enhancing Technologies (PET2004), 26-28 May 2004. Presentation file is available at `http://www.petworkshop.org/2004/program.html`
7. M. Kuhn, Filtered-Tempest font, available at `http://www.c1.cam.ac.uk/~mgk25/st-font.zip`
8. M.G. Kuhn and R.J. Anderson, Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations, Information Hiding 1998 (IH'98), Lecture Notes in Computer Science, Vol. 1525, Springer-Verlag, 1998, pp.124–142.
9. J. Loughry and D.A. Umphress, Information leakage from optical emanations, ACM Transactions on Information and System Security Vol.5 No.3, 2002, pp.262–289.

10. J-J. Quisquater and D. Samyde, Electromagnetic analysis (EMA): measures and countermeasures for smart cards, Smart cards programming and security (e-Smart 2001), Lecture Notes in Computer Science, Vol. 2140, Springer-Verlag, 2001, pp.200–210.
11. P. Smulders, The threat of information theft by reception of electromagnetic radiation from RS-232 cables, Computer and Security 9, 1990.
12. Video Electronics Standards Association `http://www.vesa.org/`
13. W. van Eck, Electromagnetic radiation from video display units: An eavesdropping risk?, Computers and Security 4, 1985.

Author Index