

# 國立交通大學

## 資訊工程系

### 博士論文

代理簽章與前向預防式代理簽章之研究



Studies on Proxy Signatures with/without Proactive Property

研究生：陳以德

指導教授：葉義雄 教授

中華民國九十四年十二月

代理簽章與前向預防式代理簽章之研究

**Studies on Proxy Signatures with/without  
Proactive Property**

研究生：陳以德

Student : I-Te Chen

指導教授：葉義雄

Advisor : Yi-Shiung Yeh

國立交通大學  
資訊工程系  
博士論文



A Dissertation  
Submitted to Department of Computer Science  
College of Computer Science  
National Chiao Tung University  
in partial Fulfillment of the Requirements  
for the Degree of  
Doctor of Philosophy  
in  
Computer Science  
December 2005

HsinChu, Taiwan, Republic of China

中華民國九十四年十二月

# 國立交通大學

## 博碩士論文全文電子檔著作權授權書

本授權書所授權之學位論文，為本人於國立交通大學 資訊工程 系，94 學年度第 1 學期取得博士學位之論文。

論文題目：代理簽章與前向預防式代理簽章之研究  
Studies on Proxy Signatures with/without Proactive Property

指導教授：葉義雄

同意  不同意

本人茲將本著作，以非專屬、無償授權國立交通大學與台灣聯合大學系統圖書館：基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學及台灣聯合大學系統圖書館得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：

本校及台灣聯合大學系統區域網路	■中華民國 95 年 9 月 1 日公開
校外網際網路	■中華民國 96 年 9 月 1 日公開

授權人：陳以德

親筆簽名：\_\_\_\_\_

中華民國 94 年 12 月 30 日

# 國立交通大學

## 博碩士紙本論文著作權授權書

本授權書所授權之學位論文，為本人於國立交通大學資訊工程系，94學年度第1學期取得博士學位之論文。

論文題目：代理簽章與前向預防式代理簽章之研究  
Studies on Proxy Signatures with/without Proactive Property

指導教授：葉義雄

### ■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行閱覽或列印。

授權人：陳以德

親筆簽名：\_\_\_\_\_

中華民國 94 年 12 月 30 日

# 國家圖書館博碩士論文電子檔案上網授權書

ID:GT008617818

本授權書所授權之學位論文，為本人於國立交通大學資訊工程系，94學年度第1學期取得博士學位之論文。

論文題目：代理簽章與前向預防式代理簽章之研究  
Studies on Proxy Signatures with/without Proactive Property

指導教授：葉義雄



茲同意將授權人擁有著作權之上列論文全文(含摘要)，非專屬、無償授權國家圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

※ 讀者基於非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

授權人：陳以德

親筆簽名：\_\_\_\_\_

中華民國 94 年 12 月 30 日

# 代理簽章與前向預防式代理簽章之研究

學生：陳以德

指導教授：葉義雄博士

國立交通大學資訊工程系博士班

## 摘 要

網際網路的發達，使得政府及工商業界的文書往來，漸漸地由紙式文件，改為利用網際網路傳遞的電子式文件；簽章部分也由傳統的印章改為電子簽章。因應此一電子化的趨勢，世界各國紛紛制訂電子簽章法來推行電子簽章；中華民國政府也於 2001 公告自 2002 開始施行電子簽章法。

電子簽章又稱數位簽章，其發展到 1996 年，Mambo 才提出代理簽章的概念。代理簽章提供了原始簽章者，可以授權給代理簽章者代簽電子簽章的功能，是近十年來，蓬勃發展的電子簽章應用之一。許多學者也提出增進代理簽章安全性及不同的代理簽章演算方法來實現代理簽章。但這些方法被質疑能否實際應用於現實生活，所以我們除了提出架構在 Quadratic Residues 上的代理簽章外；也建議代理簽章建構在標準的簽章法，如 DSA 及 ECDSA 等；並提出了建構在 DSA/ECDSA 的代理簽章法，藉由已充分討論過安全性的標準簽章法，使代理簽章成為現實可行的簽章機制。

為了解決金鑰曝光的問題，我們在現有的代理簽章法加入前向預防式 (proactive) 的概念，而提出了 proactive secret sharing proxy signature scheme。藉由定時更新金鑰的方式，確保了某一段時間內，簽章的安全性。proactive secret sharing proxy signature scheme 的復原機制，更可以在某一代理簽章者的 share 遺失或無法使用時，由其他的代理簽章者來復原其 share。

單向雜湊函數經常配合簽章使用來增進簽章的效率，自從王小雲教授提出在  $2^{69}$  的時間複雜度內可以找到單向雜湊函數 SHA-160 的碰撞後；我們也分析 SHA-160 的訊息處理模式，發現 SHA-160 有衰減(Decay)的現象，所以我們提出兩個改進 SHA-160 的訊息處理模式安全性的方法。期望我們對單向雜湊函數與代理簽章的分析與改進，能使電子簽章能實際地運用於日常生活中。



# Studies on Proxy Signatures with/without Proactive Property

Student : I-Te Chen

Advisor : Yi-Shiung Yeh

Department of Computer Science  
College of Computer Science  
National Chiao Tung University

## Abstract

Due to the rapid progress of Internet, governments and enterprises change their paper-based documents to electronic ones, as well as hand-made signatures to digital signatures. The electronic signature relative regulations are established all over the world. Taiwan has also established the Electronic Signature Laws in 2001 and put into operation in 2002.

Mambo et al. are the first group who introduced the proxy signature scheme in 1996. The proxy signatures, with which the original signers can delegate their signing capability to the proxy signers, are the most popular application of digital signatures in the last decade. Lots of researchers proposed improvement or alternative mathematic base of proxy signatures without adopting Digital Signature Algorithm (DSA) or Elliptic Curve Digital Signature Algorithm (ECDSA); however, most of the proposed proxy signature schemes are not feasible in practice because their securities cannot be really proved. Therefore, we propose the proxy signature adopting DSA and ECDSA and firstly introduce Quadratic



Residues' concepts. Our scheme keeps not only the properties of the DSA/ECDSA but also fulfills the strong requirements of proxy signatures.

To solve key exposure problem, we adopt proactive concept into proxy signature and propose proactive secret sharing proxy signature scheme. The proactive secret sharing proxy signature scheme is based on verifiable secret sharing to against the active attacker. Consequently, the proactive secret sharing proxy signature scheme, which is a group-oriented scheme, provides the functionality of proxy signers' shares renewing and recovering.

One-way hash functions are important skills to make digital signatures efficient. Wang et al. reported their method to find a collision efficiently in SHA-160 within  $2^{69}$  hash steps in February 2005. In fact, we can still discover the decay phenomenon with the application of a message schedule's judgment when inspecting how SHA-160 generates message schedule actually. Therefore, we would like to introduce two SHA-160 corrections to enhance the security of SHA-160. In general, we hope our enhancement of SHA-160 and new proxy signature schemes could be used in practice.

## 誌 謝

感謝老師、學長、同學、學弟妹們的幫忙與支持，才使我能獲得這個博士學位；這一份榮耀將分享給所有協助、支持我的人們。首先感謝有著長者的風範的指導教授－葉義雄老師，在我碩、博士修業期間，給我學業上的指導及生活上的輔導；其學術成就及待人處世之道，均為我表率。其次感謝明信學長無私的協助，使我得以順利完成學位，是我畢生難得的益友。

感謝論文口試委員呂及人教授、孫宏民教授、黃士昆教授、詹進科教授、雷欽隆教授與蔡錫鈞教授(以上按姓氏筆劃排列)的深入指導與建議，對於我論文之教誨，也是我往後做學問隨時需警惕的原則。

感謝中華電信研究所，提供我與研究相關的工作機會，尤其是張耿豪經理、謝東明博士、王文正博士與景榮兄的提攜與照顧，在此誠心感謝。

我也將最多的榮耀分享給我父母、岳父母與妻子，感謝他們支持，使我得以安心進修，完成這個學位。僅以這一點名位，聊表無限的感激。感謝在美國的表姐及在法國的以禮幫我校稿；也感謝實驗室眾兄弟姐妹們(不論已畢業或在學中)的鼓勵與幫助，尤其是兩宇-定宇與鎮宇給我的幫忙；其餘無法一一表謝，在此一併致謝。

# Contents

博碩士論文全文電子檔著作權授權書.....	III
博碩士紙本論文著作權授權書.....	IV
國家圖書館博碩士論文電子檔案上網授權書.....	V
摘 要.....	VI
ABSTRACT .....	VIII
誌 謝.....	X
CONTENTS.....	XI
LIST OF TABLES.....	XIV
LIST OF FIGURES.....	XV
CHAPTER 1 INTRODUCTION.....	1
1.1 MOTIVATION AND RELATED WORK.....	1
1.2 ORGANIZATION OF DISSERTATION.....	6
CHAPTER 2 CRYPTOGRAPHY.....	7
2.1 CRYPTOSYSTEM.....	7
2.2 SYMMETRIC CIPHERS.....	8
2.2.1 <i>Encryption Standard in U.S.</i> .....	9
2.2.2 <i>Dynamic Extended DES</i> .....	9
2.2.3 <i>NESSIE</i> .....	16
2.3 ASYMMETRIC CIPHERS.....	16



2.3.1	<i>RSA Cryptosystem</i> .....	17
2.3.2	<i>Discrete Logarithm problem</i> .....	18
2.3.3	<i>Description of Elliptic Curves</i> .....	19
2.4	ONE WAY HASH FUNCTIONS .....	24
2.4.1	<i>Secure Hash Standard</i> .....	24
2.4.2	<i>Analyze SHA-160 in message schedule</i> .....	25
2.4.3	<i>The First Modification scheme of SHA-160 (SHA-m1)</i> .....	27
2.4.4	<i>The Second Trial of SHA-160</i> .....	29
2.4.5	<i>The Third Modification scheme of SHA-160 (SHA-m2)</i> .....	30
<b>CHAPTER 3 PRELIMINARIES .....</b>		<b>32</b>
3.1	DIGITAL SIGNATURE.....	32
3.1.1	<i>Proxy signature</i> .....	33
3.1.2	<i>Strong proxy signature</i> .....	37
3.1.3	<i>Blind signature</i> .....	38
3.1.4	<i>Lamport's One time signature</i> .....	40
3.2	SECRET SHARING .....	41
3.2.1	<i>Shamir (t, n) - threshold scheme</i> .....	41
3.2.2	<i>Verifiable Secret Sharing</i> .....	42
3.3	QUADRATIC RESIDUES .....	43
3.4	DIGITAL SIGNATURE STANDARD .....	44
3.4.1	<i>DSA</i> .....	44
3.4.2	<i>ECDSA</i> .....	47
<b>CHAPTER 4 THE PROPOSED PROXY SIGNATURES .....</b>		<b>49</b>
4.1	PROXY SIGNATURE BASED ON DIGITAL SIGNATURE ALGORITHM .....	49
4.1.1	<i>Proxy Signature Based on Digital Signature Algorithm</i> .....	50



4.1.2 Proxy Signature Based on DSA.....	51
4.1.3 Proxy Signature based on ECDSA .....	55
4.1.4 Security analysis and comparisons .....	63
4.2 PROXY SIGNATURE BASED ON QR .....	69
4.2.1 Delegation by warrant proxy signature scheme based on QR.....	69
4.2.2 Correctness analysis of DWPS <sub>QR</sub> .....	74
4.2.3 Security requirements of DWPS <sub>QR</sub> .....	77
4.2.4 Time complexity and security analysis .....	79
<b>CHAPTER 5 PROXY SIGNATURE WITH PROACTIVE PROPERTY</b>	
.....	<b>81</b>
5.1 PROACTIVE SECRET SHARING SCHEME .....	82
5.2 PROACTIVE SECRET SHARING PROXY SIGNATURE SCHEME .....	87
5.2.1 Proxy Generation .....	88
5.2.2 Proxy share update.....	89
5.2.3 Proxy signature generation .....	89
5.2.4 Proxy signature verification .....	90
5.2.5 Proxy share recovery.....	91
5.3 COMPARING TO OTHER SCHEMES.....	91
<b>CHAPTER 6 CONCLUSION .....</b>	<b>92</b>
6.1 CONCLUSION.....	92
6.2 FUTURE WORKS .....	93
<b>REFERENCES .....</b>	<b>95</b>
<b>APPENDIX A.....</b>	<b>102</b>



## List of Tables

Table 2.1 The similarity of new and original S-boxes. ....	12
Table 2.2 Extended S-boxes. ....	13
Table 2.3 Points on the Elliptic Curve $E(\mathbf{Z}_{19}^*)$ .....	20
Table 2.4 Comparison between all SHA-serial algorithms .....	24
Table 2.5 Different message block between SHA and SHA-160.....	26
Table 2.6 $w_{27}$ in SHA and in SHA-160.....	26
Table 2.7 Notations of proposed scheme.....	28
Table 2.8 Parts of experiments for choosing $\{t_1, t_2, t_3\}$ .....	28
Table 2.9 Four groups of SHA-160 on $w_t = \text{ROTL}^b(w_{t-3} \oplus w_{t-8} \oplus w_{t-14} \oplus w_{t-16})$ .....	30
Table 4.1 Points on the elliptic curve $x^3 + x + 4 \pmod{19}$ .....	60
Table 4.2 The multiples of generator $G$ .....	61
Table 4.3 Time complexity of the proxy signature based on DSA/ECDSA and DSA/ECDSA .....	68
Table 4.4 Differences among DSA/ECDSA, Mambo and proposed schemes .....	68
Table 4.5 Time complexity of Manbo's and proposed scheme.....	80
Table 5.1 Comparing of Proactive Secrete Sharing Proxy Signature	91

## List of Figures

Fig 2.1 Encryption and Decryption .....	8
Fig 2.2 128-bit extended DES .....	11
Fig 2.3 One round of 128-bit extended DES.....	11
Fig 2.4 $P$ and $Q$ are two distinct points .....	21
Fig 2.5 the addition of an elliptic curve point .....	21
Fig 2.6 Terms involved between SHA and SHA-160 .....	27
Fig 2.7 Comparison between SHA-160 and SHA-m1 .....	29
Fig 2.8 Comparison $ w_i $ between SHA-160 and SHA-m2 .....	31
Fig 3.1 Signature Signing Process.....	32
Fig 3.2 Signature Verification.....	33
Fig 4.1 Proxy signer initialization in PKI .....	52
Fig 4.2 Delegation by warrant proxy signature scheme based on $QR71$	



# Chapter 1 Introduction

Due to the rapid progress of Internet, the evidence of possessing documents is especially important in the electronic world. The digital signature is developed to replace ordinary hand-written signatures without losing the properties of signer authenticity, data integrity and non-repudiation. Proxy signature scheme is one kind of digital signature applications. In this dissertation, we survey lots of proxy signature schemes and propose several novel proxy signature schemes. On the other hand, a one-way hash function is also an important skill to make digital signature efficient. Therefore, the security of one-way hash functions is also worth discussing in this dissertation.

## 1.1 Motivation and Related Work

When original signers cannot sign a document by themselves, they might delegate their signing capability to trustworthy proxy signers. For example, when the manager of a company will leave for the vacation, she/he needs to authorize her/his secretary to sign messages on behalf of her/him. To deliver manager's private key directly to her/his secretary is dangerous, nevertheless, the traditional digital signature does not provide functionality of proxy, either.

A proxy signature scheme was introduced by Mambo et al. [MUO96] to solve the proxy problem so that the original signer could delegate her/his signing capability to proxy signer without revealing her/his secret information. However, Mambo's scheme does not provide non-repudiation property [Zha97a][Sun99]; thus several papers propose non-repudiation proxy signature scheme [Zha97a][Sun99][HWW03][LHW98][LKK01b] which means both original and proxy signers cannot deny the signatures those are created exactly by themselves.



In addition, Mambo's proxy signature scheme is not a strong proxy signature scheme because it is not a proxy-protected signature scheme in which the original signer knows and can derive the proxy key on her/his own. On the contrary, in the proxy-protected proxy signature scheme, the original signer and proxy signer create the proxy key interactively so that the proxy signer can be protected from a malicious original signer. Hence, Lee and Kim [LK99][LKK01a][LKK01b] proposed the concept of the strong proxy signature, which defined the four requirements of the proxy signature: verifiability, strong unforgeability, strong identifiability, and strong undeniability. The strong proxy signature should complete all the requirements of proxy signature.

In the first, most of proxy signatures are based on discrete logarithm problem [EIG85] including Mambo's one, so that Li, Tzeng and Hwang proposed generalization of proxy signature based on discrete logarithms [LTH03]. After that, Wu and Varadharajan proposed a proxy signature based on Chinese remainder theorem [WV99]. In 2002, Chen, Liu and Chung proposed a proxy-protected signature scheme based on elliptic curve cryptosystem [CLC02], then Hwang et al proposed generalization of proxy signature based on elliptic curves [HTT04]. Furthermore, Z. H. Shao proposed the proxy signature schemes based on factoring in 2002 [Shao02] and Qingshui Xue, Zhenfu Cao proposed "Factoring based proxy signature schemes," in 2005 [XC05]. It is desirable to design proxy signature schemes based on Quadratic Residues (QR) problem.

Fan and Lei proposed efficient blind signature scheme based on QR in 1996 [FL96] and improved their scheme in 1998 [FL98]. Therefore, by adopting Fan's signature scheme, we propose the proxy signature based on QR to provide another mathematical implement.

Unfortunately, most of the proposed proxy signature schemes prior to this date are not feasible in practice because the security of those schemes cannot be really proved without

adopting standard signature such as DSA/ECDSA. The Digital Signature Algorithm (DSA) based on ElGamal [EIG85] and Schnorr's [Sch90] signature schemes is a useful digital signature scheme and has become a U.S. Federal Information Process Standard (FIPS 186) in August, 1991; called as the Digital Signature Standard (DSS) [NIST00]. In addition, the Elliptic Curve Digital Signature Algorithm (ECDSA), a DSA reinforced by the Elliptic curve cryptosystems (ECC), was invented in 1985 [ANSI99], which was also accepted as a FIPS standard (FIPS 186-2) in 2000 [NIST00].

To conquer those disadvantages, therefore, we are the first one who propose proxy-protected signature scheme combining standard signature DSA/ECDSA, as well as the Public key infrastructure (PKI) mechanism [AF99][BPH02][CFSMW03], which are pretty well known by their security properties to reinforce the proxy signature in order to be used in practice.



In many applications, the security is assured whenever the secret key remains unrevealed; therefore, a proxy key exposure is also a serious problem for proxy signature schemes. Chang, Lin and Yeh proposed "Forward Secure Proxy Signature Scheme" in NCS 2003 to deal with the key exposure problem [CLY03]. In forward secure proxy signature scheme, the proxy signer renews her/his proxy keys and deletes the previous proxy keys periodically. Those deleted proxy keys cannot be recovered, needless to mention being revealed. In addition, many threshold proxy signature schemes are proposed in which the  $k$  out of  $n$  threshold schemes [DF89][Zha97b][KPW97][SLH99][HWW03]. However, those threshold proxy signature schemes may be insufficient to construct a long-live scheme with the proactive properties to reinforce security and the proxy share cannot be recovery either.

The proactive secret sharing scheme [HJKY95], which is based on Verifiable Secret Sharing [Ped91], provides strong security for a secret sharing against the active attacker.

Consequently, the proactive secret sharing scheme is a verifiable group-oriented scheme, which provides shares renewing and recovery properties. Therefore, we adopt the concept of proactive to propose a proactive secret sharing proxy signature scheme.


A proactive secret sharing proxy signature could permit the shares of designated signers, called proxy signers, being renewed periodically without changing the secret. In particular, we apply the  $(t, n)$  threshold proxy signature scheme to allow any  $t$  or more than  $t$  signers to form a designated group from  $n$  proxy signers to sign messages on behalf of the original signer. The proxy shares of proposed scheme are periodically renewed; therefore, it will be hurtless even when the adversary obtains the proxy shares information in some period. In our proactive secret sharing proxy signature scheme; furthermore, one proxy signer can recover her/his own share from the other  $t$  proxy shares without revealing any information about the other proxy shares. Unless more than  $t$  other proxy signers cooperate and collude, the secret share algorithm is always secure.



Proxy blind signature scheme is a variant proxy signature scheme prior to this date [TLT02][SH04][LA05]. Blind signature allows a user receiving a given message signed by the original signer without revealing any information about the message itself. By using Schnorr blind signature, Tan et al. proposed two digital proxy blind signature schemes based on DLP and ECDLP in 2002 respectively [TLT02]. Moreover, Lal and Awasthi further pointed out that Tan et al.'s proxy blind signature schemes suffer from a kind of forgery attack and proposed a more efficient proxy blind signature scheme, which means Tan et al.'s schemes do not fulfill the unforgeability and unlinkability properties. Lal and Awasthi's scheme, however, does not satisfy the unlinkability property either. Therefore, Sun and Hsieh discuss the security of Tan and Lal's schemes in 2004 particularly [SH04].

Most documents are too large in size to sign digital signature; thus one-way hash

functions are important skills to make digital signature scheme efficient. SHA-160 is one of popular one-way hash functions and the security of SHA-160 is worth discussing. In 1998, F. Chabaud and A. Joux presented a method to find collisions in Secure Hash Algorithm (SHA)[NIST02] with  $2^{61}$  time complexities [CJ98]. In 2004's crypto conference and in Feb. 2005, Wang et al. [WFLY05][WY05] developed efficient methods to find collisions in MD5, as well as in SHA-160 with time complexity of  $2^{39}$  and  $2^{69}$  hash steps respectively. Furthermore, Biham and Chen [BC04] announced new analytical discoveries concerning SHA-160. Their results include a collision in a reduced-round version of SHA-160, which can be found less than 40 rounds.

Suppose the output size of one-way hash function is  $n$ -bit. According to the birthday paradox attack property [MOV96], we could expect certain collisions after trying  $2^{n/2}$  possible input values. Van Oorschot and Wiener [OW94] have explained how such a brute-force attack might be implemented.  That implies any cryptanalysis method with higher complexity than the birthday paradox attack will be regarded as inefficient. F. Chabaud and A. Joux find collision in SHA with  $2^{61}$  complexities, related to differential cryptanalysis of block ciphers [CJ98], and their method is theoretically faster than birthday paradox attack. Unfortunately, in SHA-160, their method is unable to detect collision faster than the birthday paradox attack.

In fact, we can still discover the decay phenomenon with the application of a message schedule's judgment when inspecting how SHA-160 generates message schedule actually. Furthermore, we find a reason why move SHA to SHA-160. The more nonlinear terms are involved, the more terms in message schedule process will be effective. Therefore, we would like to introduce two SHA-160 corrections to enhance the security of SHA-160. This analysis could also be used in all SHA-serials or other one-way hash functions.

## 1.2 Organization of Dissertation

We describe our motivation and related work in this chapter; and then report some fundamental cryptosystem knowledge and discuss one-way hash functions in chapter 2. In chapter 3, we describe some preliminaries of digital signature, (strong) proxy signature, Quadratic Residues, secret sharing, DSA, ECDSA, etc. In chapter 4, we propose novel proxy signatures based on QR, DSA and ECDSA respectively and analyze security of proposed proxy signature schemes. And then, we proposed a proactive secret sharing proxy signature in chapter 5 to deal with key exposure and key recovery problems. Finally, we summarize a conclusion in chapter 6 and list references respectively.



# Chapter 2 Cryptography

## 2.1 Cryptosystem

A cryptosystem can provide following properties [Sta03]:

1. **Secrecy:** It can prohibit the eavesdroppers from receiving plaintext.
2. **Authentication:** It can identify a message from its origin for the receiver and the eavesdroppers cannot disguise as someone.
3. **Integrity:** It can verify that a message has not been modified so that eavesdroppers can't replace a legal message by a false one in transmission.
4. **Non-repudiation:** It can prove the message of the sender who may falsely deny later that he had sent the message.

And a cryptosystem is composed of five basic components:

$m$  : plaintext message space.

$c$  : ciphertext message space.

$K$  : key space.

$E$  : Encryption.

$D$  : Decryption.



We show mathematic form and figure as follows:

$$E_{k_e}(m) = c \quad \text{For a given key } k \in K$$

$$D_{k_d}(c) = m \quad \text{For a given key } k \in K$$

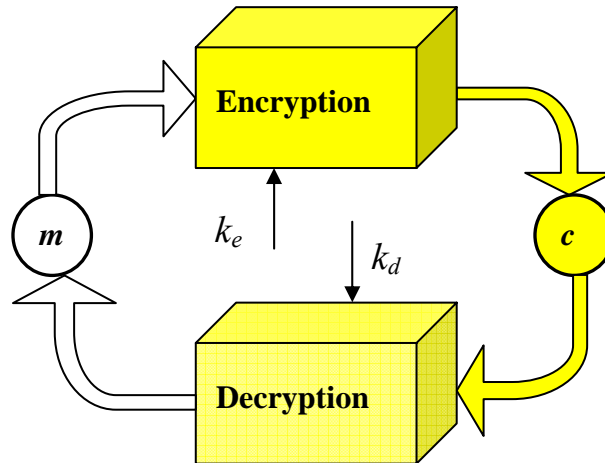


Fig 2.1 Encryption and Decryption

In Fig 2.1, cryptosystem uses keys  $k_e$  and  $k_d$  for **Encryption** and **Decryption** respectively. Simmons [Sim79] classifies the cryptosystems as symmetric (one key) and asymmetric (two keys). In symmetric cryptosystem, also called secret-key cryptosystem, the encryption key and the decryption key are the same or can be easily determined from each other. On the other hand, in asymmetric cryptosystem, also called public-key cryptosystem, the encryption key and the decryption key are different.

In Kerckhoffs's assumption [Ker83], the strength of a cryptographic system cannot rely on attacker's unawareness about the cryptosystem algorithm. A secure cryptographic system must be published and unbreakable even under the most fatal attack by the world's best cryptographers for years.

## 2.2 Symmetric Ciphers

There are two kinds of symmetric ciphers, stream ciphers and block ciphers. The stream ciphers encrypt plaintext one byte or one bit in one time span; one-time pad is one kind of stream ciphers. On the other hand, the block ciphers operate on fixed-length groups of bits to form the blocks.

## 2.2.1 Encryption Standard in U.S.

FIPS (Federal Information Processing Standard) - 46: DES (Data Encryption Standard) announced by U.S. Government in 1977 has been generally used. DES is a 64-bit block cipher with the key length of 56 bits. Unfortunately, Electronic Frontier Foundation (EFF) using a special purpose "DES cracker" machine proved DES insecure in July 1998 [EFF98]. Therefore, NIST (National Institute of Standards and Technology) announced Triple DES as FIPS 46-3 to enhance original DES. Triple DES uses three keys and three executions of the DES algorithm following an encrypt-decrypt-encrypt sequence.

Triple DES with 168 bits key and 64 bits the same block size as DES is not for long-term use [Sta03]. For reasons of both efficiency and security, a larger block size is desirable. Hence, NIST began the process of replacing DES with AES (Advanced Encryption Standard) in 1997 and Rijndael was published as AES: FIPS – 197 in 2001 [NIST01]. AES uses a 128 bits block size and its key length that can be 128, 192, and 256 bits. Four different stages are used in AES: substitute bytes (S-box), shift rows, mix columns, and add round key.



## 2.2.2 Dynamic Extended DES

The original S-boxes of DES are important design to resist differential attack. Furthermore, Yeh and Hsu proposed the extended DES [YH02], which developed eight more new S-boxes with the same cryptographic properties as original S-boxes in DES. These 16 S-boxes are used to construct the extended DES, which double the block cipher and key size. As a result, the time complexity of differential cryptanalysis of the extended



DES is 2110. We propose an intricate extended DES that includes permutation on S-boxes. By keeping the permutation information in secret, the new version of extended DES is stronger to defeat differential and linear attacks by 20922789888000 times.

### **The Extended DES**

The extended DES [YH02] has exactly the same data flow and concept as DES. The eight more S-boxes are used in the extended DES to double the block cipher and key size. Some modifications are necessary on P-box and key scheduling algorithms.

The extended DES encrypts a 128-bit data block with a 112-bit key. All data bits go through an initial permutation. The data bits then split into two 64-bit data blocks called as right and left data blocks. Two data blocks then go through 32 identical rounds, there is no swap of two data blocks in the last round. After the last round, two data blocks are combined into a 128-bit block. The result will be through the inverse initial permutation.

In each round, the right data block and 96-bit sub-key ( $R_i$  and  $K_i$  in Figure 2.2) are combined by a round function called  $F$ . The output of  $F$  is then combined with the left part data block by XOR operation. The two data blocks swap in the next round.

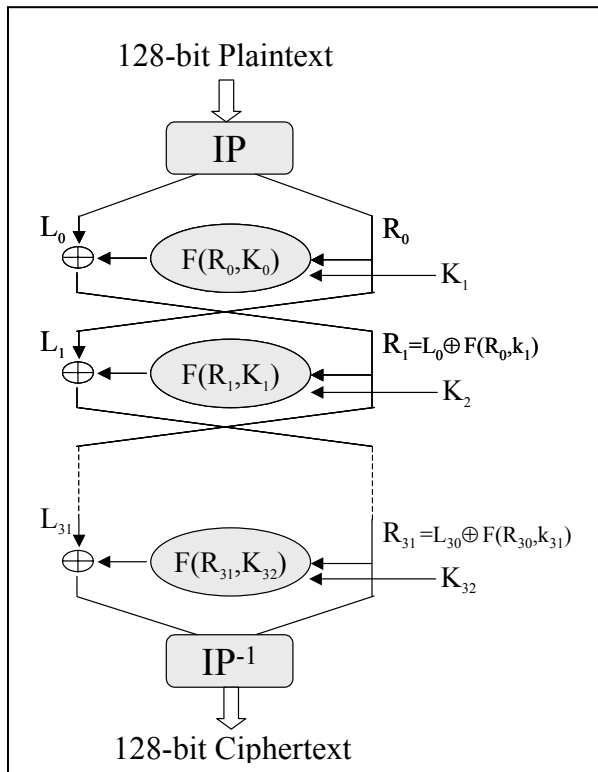


Fig 2.2 128-bit extended DES

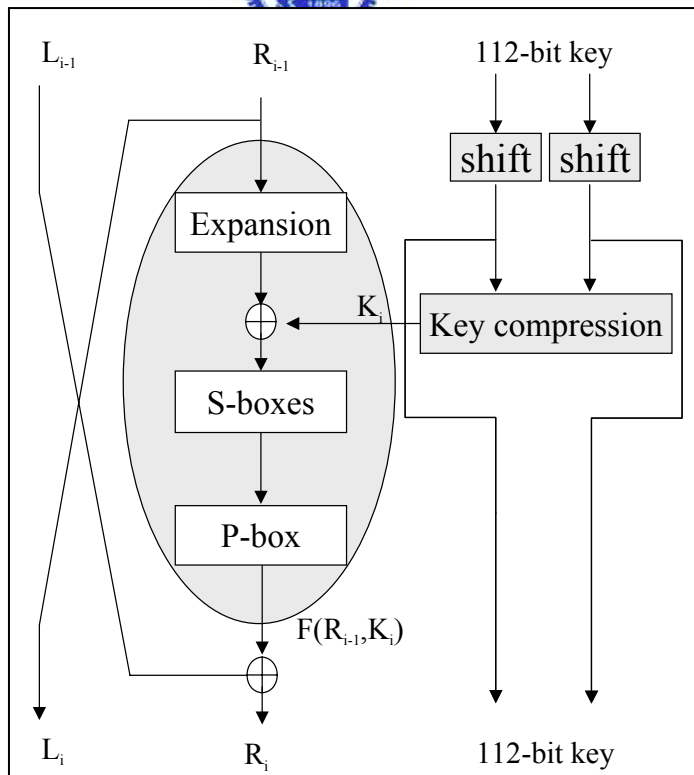


Fig 2.3 One round of 128-bit extended DES

The 64-bit right data block is expanded to 96 bits by expansion permutation after combining with the 96-bit sub-key; the 96-bit data is distributed to all 16 S-boxes as input. Each S-box has 4 output bits. Therefore, 64-bit data is used in the next step where P-box is the permutation box.

Eight more new S-boxes are proposed in following tables. Table 2.1 shows the cryptographically similarity of new S-boxes and original S-boxes. They are also semi-similar. The new S-boxes are listed in Table 2.2.

Table 2.1 The similarity of new and original S-boxes.

New design	Original	LST	B1	B2	C order	GD	ID	OD	L1	L2	L3	L4	GL	None-zero rate
S-box #9	S-box #1	20	3	3	1	9.31	32.25	46.56	18	20	22	18	78	79.4%
S-box #10	S-box #2	28	3	3	1	11.22	35.81	56.32	22	20	18	18	78	78.6%
S-box #11	S-box #3	24	3	4	1	12.65	41.70	63.62	18	22	20	18	78	79.6%
S-box #12	S-box #4	12	3	2	2	8.16	32.66	44.00	22	22	22	22	88	68.5%
S-box #13	S-box #5	20	3	2	1	9.90	35.81	55.32	22	20	18	20	80	76.5%
S-box #14	S-box #6	24	3	3	1	11.31	38.85	59.53	20	20	20	20	80	80.4%
S-box #15	S-box #7	24	3	3	1	12.17	43.45	65.18	18	22	14	20	74	77.2%
S-box #16	S-box #8	20	3	2	1	10.95	38.71	56.21	22	20	20	22	84	77.1%

- LST: Linear structure tolerance.
- B1: First order 0-1 balance tolerance.
- B2: Second order 0-1 balance tolerance.
- C order: Maximum order of completeness.
- GD: Global SAC-map distance.
- ID: Input SAC-map distance
- OD: Output SAC-map distance
- Li: Nonlinearity of output bit i.
- GL: Global nonlinearity
- None-zero rate: Percentage of none zero entry in the DDT map.

Table 2.2 Extended S-boxes.

3 0 9 7 15 12 6 11 14 13 2 1 5 10 8 4 0 3 5 8 9 15 12 6 13 10 11 7 14 4 2 1 15 5 12 2 0 11 9 14 4 3 1 8 10 6 7 13 9 15 0 5 10 6 3 8 2 12 13 11 4 1 14 7 <b>S-box #9</b>	1 10 15 12 8 3 6 5 13 4 0 7 14 9 11 2 4 7 10 0 15 9 1 12 8 14 3 13 5 2 6 11 2 5 4 10 7 12 9 3 11 8 14 1 13 6 0 15 7 0 9 3 4 15 10 6 2 13 5 14 11 8 12 1 <b>S-box #10</b>
15 4 12 1 5 10 2 13 3 8 6 11 0 7 9 14 6 13 15 2 8 4 5 11 0 7 9 12 3 10 14 1 4 13 15 10 2 1 8 6 14 3 0 5 11 12 7 9 13 3 1 4 11 14 2 8 7 10 12 15 0 5 9 6 <b>S-box #11</b>	10 7 15 12 4 2 1 11 0 13 5 3 9 14 6 8 6 13 12 0 1 7 11 14 3 8 9 15 10 4 5 2 4 1 2 11 15 12 8 6 7 10 14 5 0 9 13 3 1 11 7 14 12 0 2 5 13 6 4 9 3 10 8 15 <b>S-box #12</b>
4 7 1 12 14 11 8 2 13 10 6 9 0 5 3 15 13 0 2 7 4 14 1 11 3 12 5 10 15 9 8 6 10 1 12 11 9 2 7 14 6 13 15 4 5 8 0 3 7 11 9 4 2 1 14 13 0 6 10 3 12 15 5 8 <b>S-box #13</b>	2 14 15 0 12 11 9 5 4 13 8 3 1 6 7 10 12 5 9 10 7 0 2 15 3 6 14 13 8 11 4 1 12 2 3 14 15 4 10 9 11 1 5 8 6 13 0 7 1 15 12 5 10 9 7 2 6 8 0 14 3 4 13 11 <b>S-box #14</b>
13 2 4 7 3 12 8 1 0 15 14 9 5 10 11 6 3 8 14 13 9 2 5 11 15 4 0 10 12 7 6 1 2 11 8 13 15 0 4 14 12 5 1 6 10 3 7 9 1 13 6 1 8 2 11 14 5 10 9 12 3 7 4 0 15 1 <b>S-box #14</b>	12 2 10 7 1 4 15 8 11 14 0 9 13 3 6 5 2 1 9 4 7 14 12 11 13 8 3 15 10 5 0 6 11 15 8 4 13 2 7 14 0 5 6 3 10 9 12 13 6 1 8 2 5 14 4 7 10 12 15 9 3 0 <b>S-box #16</b>

### Permuted S-boxes

Extended DES has 16 fixed S-boxes, each of them is a mapping from  $\{0, \dots, 63\}$  to  $\{0, \dots, 15\}$ , or formulated as  $S: [0\dots 63] \rightarrow [0\dots 15]$ , used in a settled order. Unfortunately, this usage is convenient for cryptanalysis. To remedy the situation, more complicated use of S-boxes should be effectuated.

The change is to rearrange the order of S-boxes in the succeeding round. In detail, a permutation mappings  $p: [1\dots 16] \rightarrow [1\dots 16]$  is used to construct the new order. The  $i^{\text{th}}$  S-box in the  $j^{\text{th}}$  round will be equal to the  $p(i)^{\text{th}}$  S-box in the  $(j-1)^{\text{th}}$  round. For example, the S-boxes sequence in the former round is  $S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12} S_{13} S_{14} S_{15}$

$S_{16}$  and given the permutation as (3,9,16,2,11,7,10,8,1,12,4,14,6,13,5,15); in the next round, the S-boxes sequence then becomes  $S_3 S_9 S_{16} S_2 S_{11} S_7 S_{10} S_8 S_1 S_{12} S_4 S_{14} S_6 S_5 S_{15}$ .

By keeping the permutation information in secret, the exact usage of S-boxes is not explicit. This increases the difficulty of cryptanalysis.

### Substitution Words Access

The whole S-boxes data can be filled into a table that forms as a two-dimension, 16x64, matrix. Without loss of generality, let the table be  $M[1...16, 1...64]$  and the initial S-boxes sequence be  $S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12} S_{13} S_{14} S_{15} S_{16}$ . The  $k^{\text{th}}$  word (4-bit) of  $S_i$  is placed in  $M[i, k]$ . While applying an S-boxes permutation  $p$ , the S-boxes sequence of first encrypting round will be  $S_{p(1)} S_{p(2)} S_{p(3)} S_{p(4)} S_{p(5)} S_{p(6)} S_{p(7)} S_{p(8)} S_{p(9)} S_{p(10)} S_{p(11)} S_{p(12)} S_{p(13)} S_{p(14)} S_{p(15)} S_{p(16)}$ ; that is, the  $k^{\text{th}}$  word of the  $i^{\text{th}}$  S-box is placed in  $M[p(i), k]$  now. Generally, the S-boxes sequence of the  $j^{\text{th}}$  round is:

$$S_{p^j(1)} S_{p^j(2)} S_{p^j(3)} S_{p^j(4)} S_{p^j(5)} S_{p^j(6)} S_{p^j(7)} S_{p^j(8)} S_{p^j(9)} S_{p^j(10)} S_{p^j(11)} S_{p^j(12)} S_{p^j(13)} S_{p^j(14)} S_{p^j(15)} S_{p^j(16)},$$

where  $p^j(i)$  denotes to execute the mapping  $p$  with  $j$  times such as  $p(p(\dots p(p(i))\dots))$ . It is obviously that the  $k^{\text{th}}$  word of the  $i^{\text{th}}$  S-box of the  $j^{\text{th}}$  round is placed in  $M[p^j(i), k]$ .


According to the above derivation, we know that a word in an S-box can still be easily read from the S-boxes table while including the S-boxes permutation. The increasing calculations are just some mapping operations and never exceed 16 times of nested mapping because of the restriction of 16 rounds in extended-DES. Therefore, the new algorithm is considered as the same efficient as extended-DES. While decrypting, the same 16 S-boxes sequences in encryption are used but in reverse order. This does not increase the computing time complexity.

## Permutation Materials

The adopted S-boxes permutation should be kept in secret. It can be added up some other secret information like an independent key to the system. This will increase the quantity of secret information; system will be more secure in this viewpoint. On the other hand, there turns out more secret data to be managed which may raise the burden for users.

Alternatively, the S-boxes permutation can be also derived from the key. For example, we can choose the smallest integer between A and B, which is larger than the key value and relatively prime to 16 as the multiplier. The  $i^{\text{th}}$  value of the permutation function  $p$ , will be  $p(i)=(A+i*B \bmod 16)+1$ .

## Security Analysis on Dynamic Extended DES

Both differential and linear attacks  need to know the exact usage of S-boxes. If we can keep the permutation in secret, it will be difficult for the adversary to apply the two attacks. The attack may guess the permutation with rare  $\frac{1}{20922789888000}$  probability because 16 S-boxes can derive  $16! = 20922789888000$  different permutations. It is computational inefficiency to guess the right permutation.

Furthermore, if higher security is required, the permutations used in each round can be different. That is, using 16 different permutations to construct the initial S-boxes sequence, and applying them in different rounds. The probability to guess the right permutation reduces to  $\frac{1}{20922789888000^{16}} \cong \frac{1}{1.34869 \times 10^{214}}$  to be computational impossible.

Dynamic Extended DES permutes the S-boxes order in the succeeding round; as a result, the usage of S-boxes becomes more confused. This change can enhance extended DES to resist differential and linear attacks. In addition, this method can be also used in

any other S-boxes. However, the permutation information should be always kept in secret; otherwise, not only the confusion effect will no more exists, but also become even favorable for the cryptanalysis.

### 2.2.3 NESSIE

New European Schemes for Signature, Integrity, and Encryption (NESSIE) project has launched out the next generation of cryptographic algorithms in 2000 [Nessie04]. The NESSIE portfolio of cryptographic primitives has been announced in February 2003. In block cipher scheme, MISTY1 (64-bit), AES (128-bit), Camellia (128-bit), SHACAL-2 (256-bit) are recommended algorithms. MISTY1 is similar to the block cipher KASUMI, which has been scrutinized prior to its adoption as a 3GPP standard, so many analyses for KASUMI would be also applicable to MISTY1. AES is FIPS – 197 announced by U.S. NIST; and Camellia has many similarities to the AES. SHACAL-2 is based on a one-way hash function upon SHA [NIST02] used in encryption mode. The strength of SHACAL-2 is inheritance from the extensive analysis that has been made on SHA. Although RC6 is also a secure block cipher, the NESSIE felt unable to consider RC6 [RRSY98] owing to ongoing serious Intellectual Property Rights issues.



## 2.3 Asymmetric Ciphers

Diffie Whitfield and Hellman introduced asymmetric ciphers in 1976 [DH76]. Asymmetric ciphers rely on one key for encryption and a different but related key for decryption; nevertheless, it is computationally infeasible to determine the decryption key given only the knowledge of the algorithm and encryption key. For example, asymmetric ciphers cryptosystem encrypting the sender's messages by using recipient's "public" key

and the recipient's "private" key can decrypt the messages. RSA [RSA78], ElGamal [ElG85] and Elliptic curve [Men93][ANSI99][Han04] are most popular asymmetric cryptosystems that we describe as follows:

### 2.3.1 RSA Cryptosystem

Rivest, Shamir, and Adleman developed the RSA algorithm in 1977 [RSA78]; the letters RSA are the initials of their surnames. The RSA scheme makes use of factoring problem to generate key pairs described as follows:

1. Let  $p$  and  $q$  are large primes such that  $p \neq q$  and  $n = pq$ .
2. Compute the Euler's totient function  $\phi(n) = (p-1)(q-1)$ .
3. Choose a integer  $e$ , where  $1 < e < \phi(n)$  and  $e$  is coprime to  $\phi(n)$  i.e.  $\gcd(e, \phi(n))=1$ .
4. Compute  $d$  such that  $ed = 1 \pmod{\phi(n)}$ .
5. The public key is  $(e, n)$  and the private key is  $(d, n)$ .



Euler's theorem shows that  $a^{\phi(n)} \pmod n = 1$ ; thus to encrypt message  $m$ , we could compute  $m^e \pmod n = c$  to obtain ciphertext  $c$ . And to decrypt ciphertext  $c$ , we could compute as follows:

$$\begin{aligned}
 & c^d \pmod n \\
 &= (m^e \pmod n) \pmod n \\
 &= m^{ed} \pmod n \\
 &= m^{(k\phi(n)+1)} \pmod n \\
 &= m^1 \pmod n \\
 &= m
 \end{aligned}$$



We use artificially small parameters here; but we can also use OpenSSL [OpenSSL] to generate and examine key pairs. For example, let  $p = 101$ ,  $q = 53$ ,  $n = 101 \cdot 53 = 5353$ , the message  $m = 4657$ , and choose  $e = 743$  for public key. Via Euclidean algorithm, we could compute the private key  $d = 7$  so that the public and the private key are  $(743, 5353)$  and  $(7, 5353)$  respectively. The encryption function is

$$E_{743}(4657) = 4657^{743} \bmod 5353 = 1003$$

and the decryption function is:

$$D_7(1003) = 1003^7 \bmod 5353 = 4657$$

Both of these computations can be done efficiently using the square-and-multiply algorithm for modular exponentiation. RSA is much slower than symmetric cryptosystems. In practice, sender typically encrypts a secret message with a symmetric algorithm, encrypts the symmetric key with RSA, and transmits both the RSA-encrypted symmetric key and the symmetrically- encrypted message to receiver.



### 2.3.2 Discrete Logarithm problem

Discrete logarithms are defined in group theory, which is a collection of elements together with a binary operation. A primitive root  $g \in \mathbf{Z}_p^*$ , a number  $x$  under multiplication modulo the prime  $p$ , and  $g^x$  denoting the element obtained by multiplying  $g$  itself by  $x$  times; by Fermat's little theorem, we know that for a primitive root  $g \in \mathbf{Z}_p^*$ ,  $g^p = g \bmod p$ , and the set of group is:

$$\{g, g^2, g^3, \dots, g^{p-1}\} = \{1, \dots, p-1\}.$$

The discrete logarithm problem is as follows: given a primitive root  $g$  in  $\mathbf{Z}_p^*$  and another element  $y \in \mathbf{Z}_p^*$ , finding an integer  $x$  such that  $g^x = y \pmod p$ . For example, the solution to the problem  $3^x = 15 \pmod{17}$  is 6, because  $3^6 = 729 = 15 \pmod{17}$ . If in  $\mathbf{Z}_n^*$ , where  $n$  is not a prime number, by Euler's theorem,  $\alpha^{\phi(n)} \pmod n = 1$ , the set of group will be:

$$\{\alpha, \alpha^2, \alpha^3, \dots, \alpha^{\phi(n)}\}.$$

The ElGamal cryptosystem [ElG85] is based on the discrete logarithm problem. For a generator (primitive root)  $g \in \mathbf{Z}_p^*$  of order  $p$ , Alice chooses a random  $x$  from  $\{0, \dots, p-1\}$  and computes  $y = g^x$ . The values  $p$ ,  $g$ , and  $y$  are the public keys and  $x$  is a private key of Alice. To encrypt a message  $m$  to Alice, we show as follows:

1. Convert  $m$  to into an element of  $\mathbf{Z}_p^*$
2. Choose a random  $k$  from  $\{0, \dots, p-1\}$
3. Calculate  $c_1 = g^k \pmod p$  and  $c_2 = my^k \pmod p$
4. Send the ciphertext  $(c_1, c_2)$  to Alice



Then, Alice can decrypt ciphertext by computing  $c_2 (c_1^x)^{-1}$ .

$$c_2 (c_1^x)^{-1} = my^k (g^{kx})^{-1} = mg^{xk} (g^{kx})^{-1} = m \pmod p.$$

### 2.3.3 Description of Elliptic Curves

In general, elliptic curves take the form:  $y^2 + axy + by = x^3 + cx^2 + dx + e$  where  $a, b, c, d$ , and  $e$  are the real numbers satisfying to some conditions [Han04]. There are two finite fields  $\mathbf{Z}_p^*$  and  $\mathbf{Z}_{2^n}^*$ . The elliptic curve  $E$  over  $\mathbf{Z}_p^*$  and  $\mathbf{Z}_{2^n}^*$  are defined as definition 2.1 and 2.2 respectively:

**Definition 2.1:** Let  $a, b \in \mathbf{Z}_p^*$  be constants such that  $4a^3+27b^2 \neq 0$ . An elliptic curve is the set  $E$  of solutions  $(x, y) \in \mathbf{Z}_p^*$ , to the equation:

$$y^2 = x^3 + ax + b \tag{2.1}$$

together with a special point  $O$  called the point at infinity.

**Definition 2.2:** Let  $a, b \in \mathbf{Z}_{2^n}^*$  be constants such that  $b \neq 0$ . An elliptic curve is the set  $E$  of solutions  $(x, y) \in \mathbf{Z}_{2^n}^*$ , to the equation:

$$y^2 + xy = x^3 + ax^2 + b \tag{2.2}$$

together with a special point  $O$  called the point at infinity.

We concentrate on elliptic curves over finite fields  $\mathbf{Z}_p^*$ . An example of elliptic curve  $E$  over  $\mathbf{Z}_p^*$  as following:



Let  $p = 19$  and consider the elliptic curve  $E: y^2 = x^3 + x + 4$  defined over  $\mathbf{Z}_{19}^*$ . In this case,  $a = 1$  and  $b = 4$ . We have  $4*1^3+27*4^2 \pmod{19} = 18 \neq 0$ , which satisfies the condition for an elliptic group mod 19. The order of points in  $E(\mathbf{Z}_{19}^*)$  is also 19 and all the points and  $O$  are list as following:

Table 2.3 Points on the Elliptic Curve  $E(\mathbf{Z}_{19}^*)$

(0, 2)	( 6,13)	(11, 4)
(0,17)	( 8, 7)	(11,15)
(1, 5)	( 8,12)	(14, 8)
(1,14)	(9, 1)	(14,11)
(5, 1)	(9,18)	$O$
(5,18)	(10, 8)	
(6, 6)	(10,11)	

The addition and multiplication operation in ECC are counterpart of modular multiplication and exponentiation in RSA, respectively. Let  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$  be two points on an elliptic curve  $E$ . Then,  $P + Q = R$ , we show it as Fig 2.4 and Fig 2.5 geometrically.

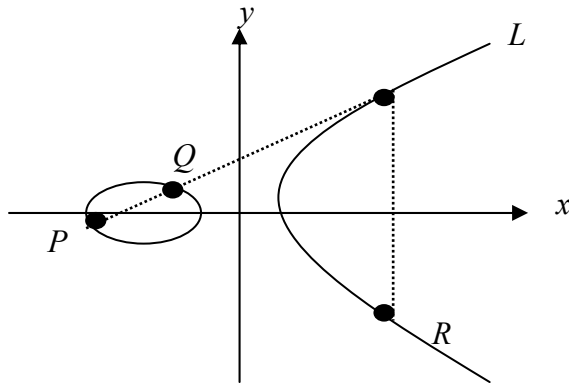


Fig 2.4  $P$  and  $Q$  are two distinct points

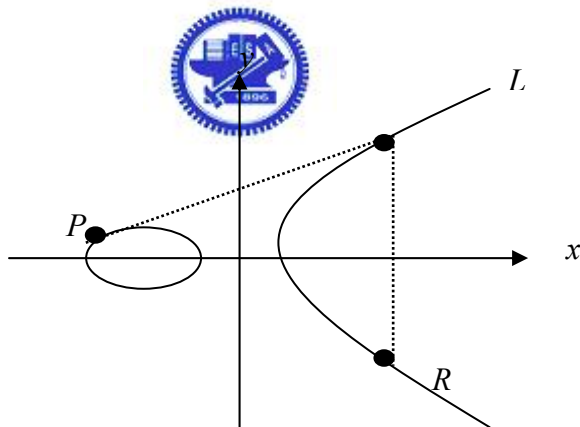


Fig 2.5 the addition of an elliptic curve point

First at all, we have to find the slope of  $\overline{PQ}$ , where  $P \neq Q$  or the tangent line of  $P$ , where  $P = Q$ . We show as following:

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q, \text{ where } \lambda \text{ is slope of line } \overline{PQ}. \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q, \text{ where } \lambda \text{ is tangent line of } P. \end{cases} \quad (2.3)$$

The equation of line  $L$  is  $y = \lambda x + v$ . The  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$  is on  $L$  so that:  $y_1 = \lambda x_1 + v$  and  $y_2 = \lambda x_2 + v$ . We substitute  $y = \lambda x + v$  into the equation (2.1), getting the following:

$$\begin{aligned} (\lambda x + v)^2 &= x^3 + ax + b \\ x^3 - \lambda^2 x^2 + (a - 2\lambda v)x + b - v^2 &= 0 \end{aligned} \quad (2.4)$$

$x_1$  and  $x_2$  are two roots of equation (2.4), which are real. As the result, the third root, said  $x_3$ , must also be real.

$$\begin{aligned} (x - x_1)(x - x_2)(x - x_3) \\ = x^3 - (x_1 + x_2 + x_3)x^2 + (x_1x_2 + x_2x_3 + x_1x_3)x - x_1x_2x_3 &= 0 \end{aligned} \quad (2.5)$$

Comparing equation (2.4) and (2.5), we know that  $\lambda^2 = x_1 + x_2 + x_3$ . Hence,

$$x_3 = \lambda^2 - x_1 - x_2$$

The slope  $\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-y_3 - y_1}{x_3 - x_1}$ , hence:

$$y_3 = \lambda(x_1 - x_3) - y_1$$



The rules for the sum of two points and the double of one point, we summarize as follows: for all  $P, Q \in E(\mathbf{Z}_p^*)$  [Han04]:

1.  $P + O = P$
2. If  $P = (x, y)$ , then the point  $(x, -y)$  denoted as  $-P$  and  $P + (-P) = O$
3. Let  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$ , where  $P \neq Q$ , then  $P + Q = (x_3, y_3)$  where

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1, \text{ where the slope } \lambda \text{ shows as equation (2.2).} \end{cases}$$

4. Let  $n$  be the smallest integer such that  $nP = O$ , then  $n$  is the order of  $P$  over  $E$ .

Elliptic curve cryptography (ECC) [Kob87] is a public-key cryptography based on the elliptic curve discrete logarithm problem (ECDLP) proposed by Neal Koblitz and Victor Miller in 1985. Given an elliptic curve  $E$ , over a Galois field  $\text{GF}(q)$ , the operation “+” is defined as above paragraph and the operation “\*” defined as  $Z \times E(q) \rightarrow E(q)$  where  $E(q)$  is rational points form  $(x, y)$ , and both  $x$  and  $y$  are in  $\text{GF}(q)$ . If  $P$  is some point in  $E(q)$ , then we define:

$$2*P = P + P,$$

$$3*P = 2*P + P = P + P + P, \text{ and so on.}$$

The ECDLP is then to determine integer  $k$  in  $k*P = Q$ , where  $P$  and  $Q$  are the given points. For a specific base point  $G$  is selected and published for use with the curve  $E(q)$ , Alice chooses a private key  $k$  as random integer and then the value  $P = k*G$  is published as the public key. To encrypt a message  $m$  to Alice, we show as follows:



1. Convert message  $m$  (where  $0 \leq m < \frac{p}{2^k}$ ) to into an element  $P_m$  of  $E(q)$ 
  - We append  $k$  bits at the end of the message
  - Compute  $x = 2^k m + i$ , for  $i = 0, 1, \dots$ , until  $(\frac{x^3 + ax + b}{p}) = 1$ .
2. Choose a random integer  $r$
3. Calculate ciphertext  $C_m = \{rG, P_m + rP\}$
4. Send the ciphertext  $C_m$  to Alice

Alice can decrypt ciphertext by multiplying the first point in the pair of Alice's secret key and subtracts the result from the second point:

$$\begin{aligned} & P_m + rP - k(rG) \\ &= P_m + r(kG) - k(rG) \\ &= P_m. \end{aligned}$$

## 2.4 One way hash functions

A one-way hash function,  $h(m)$ , operates on an arbitrary-length message  $m$ , and returns a fixed-length hash value, called digest. One-way hash functions are widely deployed in electronic mail, electronic funds transfer, software distribution, data storage, and other applications, which require the assurance of data integrity.

### 2.4.1 Secure Hash Standard

SHA, one kind of popular one-way hash functions, was originally applied to DSA (Digital Signature Algorithm), issued by the NIST and published as a federal information processing standard (FIPS PUB 180) in 1993; a revised version was issued as FIPS PUB 180-1 in 1995 [NIST95] and is generally referred to as SHA-160. SHA and SHA-160 operate on an arbitrary-length message as input; and then output a 160-bit digest.



FIPS 180-2 [NIST02] is announced by NIST on May 30, 2001. FIPS 180-2 is a strengthened version of the SHA-160, which offers four secure hash algorithms including SHA-160, SHA-224, SHA-256, SHA-384, and SHA-512. Table 2.4 presents the basic properties of FIPS 180-2.

Table 2.4 Comparison between all SHA-serial algorithms

Algorithm	Message Size	Block Size	Word size	Message Digest Size	Security*
SHA-160	$<2^{64}$	512	32	160	80
SHA-224	$<2^{64}$	512	32	224	112
SHA-256	$<2^{64}$	512	32	256	128
SHA-384	$<2^{128}$	1024	64	384	192
SHA-512	$<2^{128}$	1024	64	512	256

Note: \*In this context, “security” refers to the fact that a birthday attack on a message digest of size  $n$  produces a collision with a work factor of approximately  $2^{n/2}$ .

The input of SHA and SHA-160 is processed in a 512-bit message block [NIST95]. First at all, for a 512-bit block, append padding and length after the message. The message block is transformed from 16 32-bit words ( $m_0, m_1, \dots, m_{15}$ ) to 80 32-bit words ( $w_0, w_1, \dots, w_{79}$ ) by the following algorithm: The difference between SHA and SHA-160 is that SHA-160 rotates 1 bit left: ROTL<sup>1</sup>.

$$w_t = m_t, \quad 0 \leq t \leq 15$$

$$w_t = \text{ROTL}^1(w_{t-3} \oplus w_{t-8} \oplus w_{t-14} \oplus w_{t-16}), \quad 16 \leq t \leq 79$$

Each of the 80 steps of the processing one 512-bit block form as:

$$A, B, C, D, E \leftarrow [\text{ROTL}^5(A) + f_t(B, C, D) + E + w_t + k_t], A, \text{ROTL}^{30}(B), C, D$$

Where  $f_t$  defines in belowing section and the logical operators (AND, OR, NOT, XOR) are represented by the symbols ( $\wedge, \vee, \neg, \oplus$ ).  $k_t$  are constants, please refer to [NIST02].

$$f_t(x, y, z) = \text{Ch}(x, y, z) = (x \wedge y) \vee (\neg x \wedge z), \quad 0 \leq t \leq 19$$

$$f_t(x, y, z) = \text{Parity}(x, y, z) = x \oplus y \oplus z, \quad 20 \leq t \leq 39$$

$$f_t(x, y, z) = \text{Maj}(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z), \quad 40 \leq t \leq 59$$

$$f_t(x, y, z) = \text{Parity}(x, y, z) = x \oplus y \oplus z, \quad 60 \leq t \leq 79$$

The output of each round becomes initial value of next round until finish whole blocks. The final output is the concatenation of  $A, B, C, D, E$ .

## 2.4.2 Analyze SHA-160 in message schedule

We examine the changes from SHA to SHA-160 and discover the decay phenomenon with the application of a message schedule's judgment when inspecting how SHA-160 generates message schedule actually.



**One reason from SHA to SHA-160:** Firstly, we define notation  $x^n = \text{ROTL}^{n \bmod 32}(x)$ .

The message schedule of  $w_t$  of SHA-160 and SHA shall be prepared respectively as follows:

Table 2.5 Different message block between SHA and SHA-160

SHA	SHA-160
$w_t = m_t, 0 \leq t \leq 15$	$w_t = m_t, 0 \leq t \leq 15$
$w_t = w_{t-3} \oplus w_{t-8} \oplus w_{t-14} \oplus w_{t-16}, 16 \leq t \leq 79$	$w_t = \text{ROTL}^1(w_{t-3} \oplus w_{t-8} \oplus w_{t-14} \oplus w_{t-16}), 16 \leq t \leq 79$

The other reason why  $\text{ROTL}^1$  function can upgrade the security level is the increase of involved terms of  $m_t$ . For example, when comparing  $w_{27}$  in both SHA and SHA-160 (shown as follows), there are only 6 terms involved in SHA compared with 14 terms involved in SHA-160.

Table 2.6  $w_{27}$  in SHA and in SHA-160

SHA involved 6 terms	$w_{27} = m_2 \oplus m_5 \oplus m_4 \oplus m_7 \oplus m_8 \oplus m_{15}$
SHA-160 involved 14 terms	$w_{27} = m_2^4 \oplus m_3^2 \oplus m_4^4 \oplus (m_5^2 \oplus m_5^3) \oplus m_7^3 \oplus m_8^2 \oplus (m_{10}^2 \oplus m_{10}^4) \oplus (m_{11}^1 \oplus m_{11}^2) \oplus (m_{13}^1 \oplus m_{13}^3) \oplus m_{15}^4$

$w_{27}$  becomes independent of  $m_5$  in the end even though  $m_5$  has been involved twice in SHA. But in SHA-160,  $m_5$  is involved under ROTL function thus  $m_5^2$  and  $m_5^3$  will not be eliminated. Belowing is a figure comparing the number of terms involved in message schedules of both SHA and SHA-160. X-axis presents the index, and y-axis presents the number of terms.

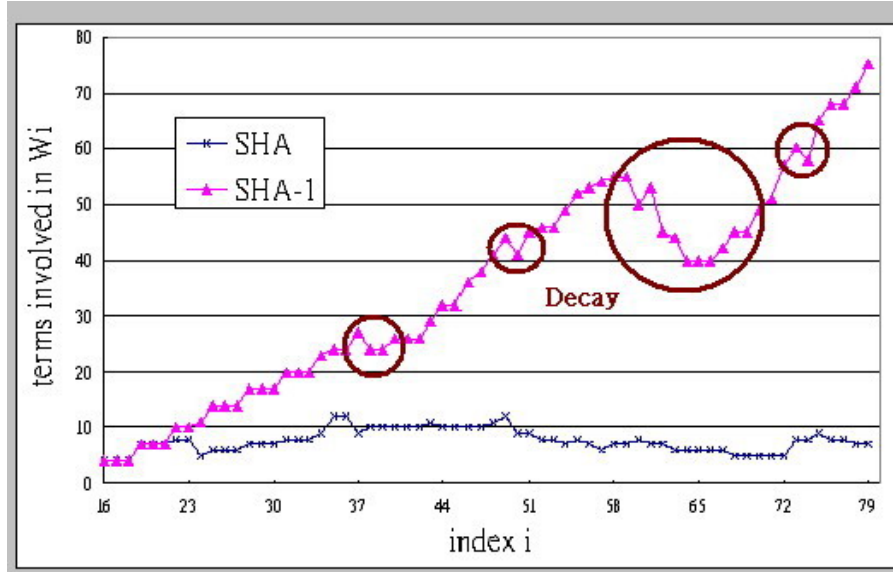


Fig 2.6 Terms involved between SHA and SHA-160

Not only the paper “Differential collisions in SHA-0” shows the security level of SHA-160 is greatly higher than SHA [BC04], we also shows the terms involved in SHA-160 is much more than in SHA in Fig 2.6. Furthermore, we find the decay phenomenon in message schedule, which points out the existence of some inefficient calculations in SHA-160. If the inefficient calculations could be modified such that the decay phenomenon postpones, much more terms will be involved in later  $w_t$ . Therefore, we would like to introduce two SHA-160 corrections to enhance the security of SHA-160.

### 2.4.3 The First Modification scheme of SHA-160 (SHA-m1)

Firstly, we re-write the original recursive equation into a general form:

$$\begin{aligned}
 w_t &= m_t & , 0 \leq t \leq 15 \\
 w_t &= \text{ROTL}^1(w_{t-1} \oplus w_{t-2} \oplus w_{t-3} \oplus w_{t-4}) & , 16 \leq t \leq 79
 \end{aligned}$$

And, we define some notations with convenience and generality. Let  $m^{(i)}$  be an input block,  $i = 0, \dots, 15$ ; and  $w_j, j = 0, \dots, 79$  be the message words.

Table 2.7 Notations of proposed scheme

$\text{ROTL}^b$	Left rotation of $b$ bits
$\text{ROTR}^b$	Right rotation of $b$ bits
$m_i^b$	Left rotation of $b$ bits on $m_i, i = 0, \dots, 15$
$w_i^b$	Left rotation of $b$ bits on $w_i, i = 0, \dots, 79$
$m_j^{b_1 \dots b_{p_j}}$	$= m_j^{b_1} \oplus m_j^{b_2} \oplus \dots \oplus m_j^{b_{p_j}}, j = 0, \dots, 15$
$w_j^{b_1 \dots b_{q_j}}$	$= w_j^{b_1} \oplus w_j^{b_2} \oplus \dots \oplus w_j^{b_{q_j}}, j = 0, \dots, 79$
$\parallel$	Concatenation
$ X $	Number of $X$

As a result, in the original SHA-160 algorithm,  $(t_1, t_2, t_3, t_4)$  equals to  $(3,8,14,16)$  according to following basic constraints:

- a.  $1 \leq t_1 \leq t_2 \leq t_3 \leq t_4 = 16$
- b.  $\text{gcd}(t_1, t_2, t_3) = 1$



There are  $C \binom{15}{3} = 455$  possibilities to assign  $(t_1, t_2, t_3)$ , where  $1 \leq t_1 < t_2 < t_3 \leq 15$ .

We list parts of experiments in Table 2.8 and the comparison between SHA-m1 and SHA-160 in Fig 2.7. We list whole experiments of assign  $(t_1, t_2, t_3)$  in appendix A. According to our experiments, the best choice is  $(t_1, t_2, t_3) = \{1, 2, 11\}$ .

Table 2.8 Parts of experiments for choosing  $\{t_1, t_2, t_3\}$

$t_1$	$t_2$	$t_3$	Total terms	Maximum number of involved terms in $w_t$	Average terms involved of all $w_t$
...					
1	2	10	7279	175	113.4844
1	2	11	8670	212	135.2188
1	2	12	7189	182	112.0781
...					

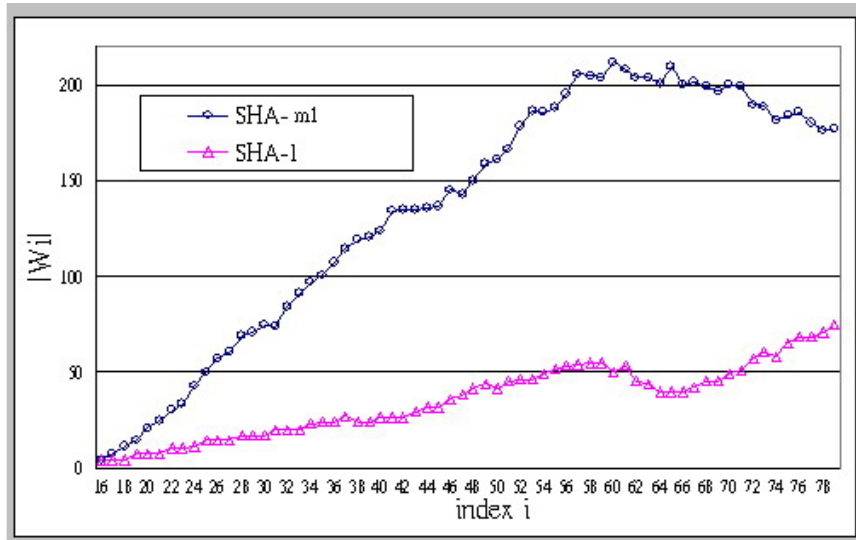


Fig 2.7 Comparison between SHA-160 and SHA-m1

SHA-m1 algorithm costs as much time as SHA-160; however, the terms involved in SHA-m1 are significantly more than in SHA-160 as shown in Fig 2.7; as well as the decay phenomenon postpones.



#### 2.4.4 The Second Trial of SHA-160

Another viewpoint to modify SHA-160 is based on the ROTL<sup>1</sup> function. We re-write the original equation and summarize 3 conclusions as follows:

$$w_t = m_t, \quad 0 \leq t \leq 15$$

$$w_t = \text{ROTL}^b(w_{t-3} \oplus w_{t-8} \oplus w_{t-14} \oplus w_{t-16}), \quad 16 \leq t \leq 79$$

1. ROTL<sup>b</sup> and ROTL<sup>32-b</sup> cause the same effect;
2. The smaller gcd(32, b) is, the more involved terms will be; and
3. ROTL<sup>n</sup> and ROTL<sup>m</sup> will cause the same effect if gcd(n,32)=gcd(m,32).

Table 2.9 Four groups of SHA-160 on  $w_t = \text{ROTL}^b(w_{t-3} \oplus w_{t-8} \oplus w_{t-14} \oplus w_{t-16})$

	$\text{gcd}(b,32)$	variations	Total Terms
$b = \{1,2,3,5,6,7,9,10,11,13,14,15,17,18,19,21,22,23,25,26,27,29,30,31\}$	$\{1,2\}$	$\{31,15\}$	2271
$b = \{4,12,20,28\}$	4	7	1733
$b = \{8,24\}$	8	3	1265
$b = 16$	16	1	725

We classify four groups as listed in Table 2.9. The original SHA-160 is one of the 24 experiments with the most terms involved. The same experiments on SHA-m1 are classified into five groups by the largest common divisor of 32 and the variable  $b$ . As a result, rotating one bit is the best choice already both in SHA-160 and SHA-m1.



### 2.4.5 The Third Modification scheme of SHA-160 (SHA-m2)

We re-write the  $w_t$  in another form:

$$w_t = m_t, \quad 0 \leq t \leq 15$$

$$w_t = (w_{t-3})^{b_1} \oplus (w_{t-8})^{b_2} \oplus (w_{t-14})^{b_3} \oplus (w_{t-16})^{b_4}, \quad 16 \leq t \leq 79 \text{ where } 0 \leq b_1, b_2, b_3, b_4 \leq 31.$$

Based on the results in second trial, we make one supposition that “The largest number of 'Terms involved in  $w_t$ ' will appear when  $b_1, b_2, b_3$  and  $b_4$  are all odds”. Hence, the time complexity to determine  $b_1, b_2, b_3$ , and  $b_4$  is reduced from  $32^4$  to  $16^4$ . We conclude two results:

1. The maximal number of ‘Terms involved in  $w_t$ ’ founded in 1280 experiments is 2509; one of them is  $\{b_1, b_2, b_3, b_4\} = \{1,3,9,3\}$ .
2. The minimum number of ‘Terms involved in  $w_t$ ’ founded in 256 experiments is 1023; one of them is  $\{b_1, b_2, b_3, b_4\} = \{1,1,3,7\}$ .

We develop SHA-m2 by using one of the best choice  $\{b_1, b_2, b_3, b_4\}=\{1,3,9,3\}$  and show the comparison between SHA-160 and SHA-m2 as follows:

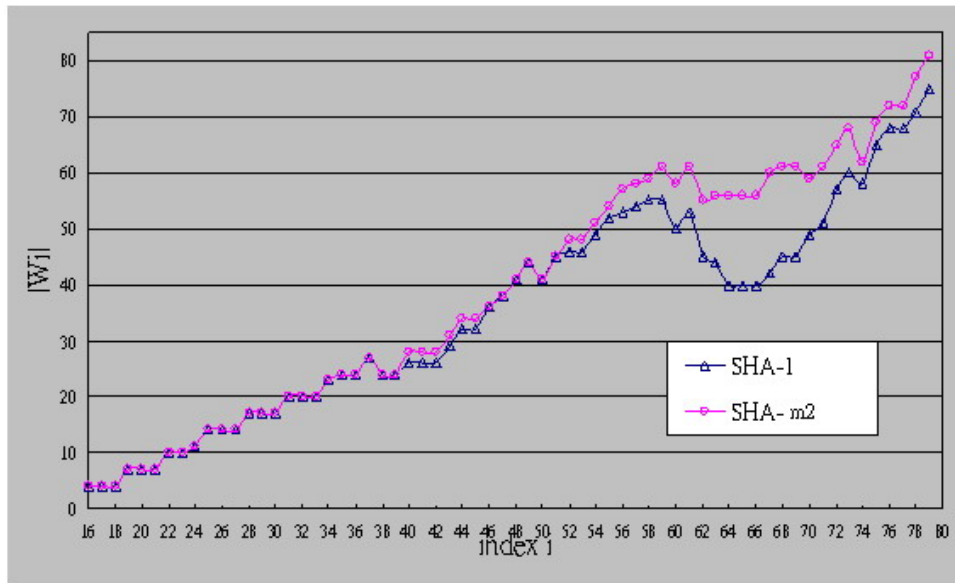


Fig 2.8 Comparison  $|w_i|$  between SHA-160 and SHA-m2



In order to increase the ‘Terms involved in  $w_i$ ’, we develop two algorithms SHA-m1 and SHA-m2 by modifying recursive equations and the number of shift-rotated-bit of SHA-160. The more nonlinear terms are involved, the more terms of  $f_i$  and

$$a = \text{ROTL}^5(a) + f_i(b,c,d) + e + K_t + w_t [3]$$

will be effective. Because the increase of the nonlinear terms really helps to enhance the security level of original SHA-160, this analysis could also be used in all SHA-serials or other one-way hash functions. Basing on our result, we can further develop the more secure one-way hash function such as SHA-1024 or SHA-2048.

## Chapter 3 Preliminaries

Cryptographic primitives are widely used in network security. We briefly describe, in this chapter, some necessary cryptographic primitives including digital signature, (strong) proxy signature, proactive secret sharing, and one-way hash functions. Based on those basic primitives, we can further enhance and improve those original primitives.

### 3.1 Digital Signature

The purpose of a digital signature, which is created to replace the hand-written signature in the electronic world, is to bind its identity with a piece of message. Digital signature, which is fundamental in authentication, authorization, and non-repudiation, protects two parties exchanging messages from the interception of any third party. We show the signature signing process as follows.

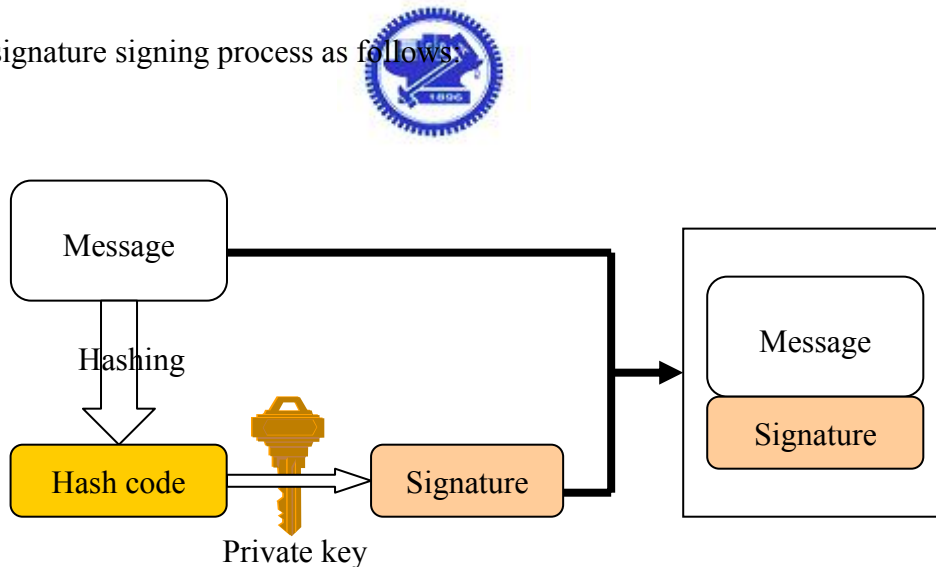


Fig 3.1 Signature Signing Process

Anyone can verify signature via sender's public key and compare the relationship between the signature (decrypted hash code) and hash code of message. The purpose of hash code is to increase the signature signing efficiency and we show the verification process as follows.

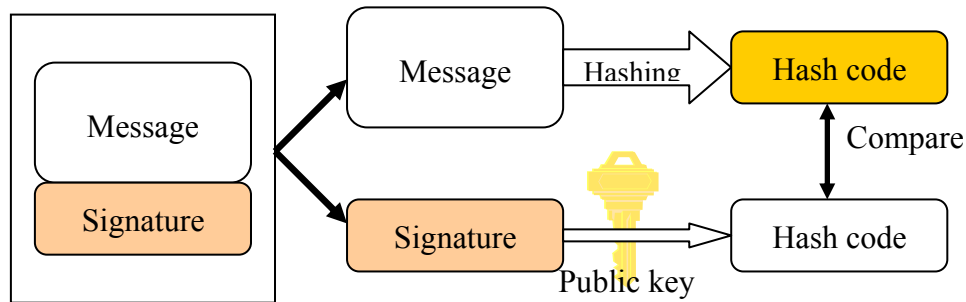


Fig 3.2 Signature Verification

Bruce Schneier identifies the characteristics of a good digital signature in his book “Applied Cryptography” [Sch96] as follows:

1. The signature is unforgeable.
2. The signature is authentic.
3. The signature is not reusable.
4. The signed document is unalterable.
5. The signature cannot be repudiated.



Diffie and Hellman invented the concept of public key cryptography in 1976 [DH76]. There are two kinds of most popular public key cryptosystems; one is the RSA signature scheme [RSA78], which was the first method by encrypting the entire message or the hash code of message with the sender's private key. The other is based on the discrete logarithm problem [EIG85]. Afterward, many researches have developed alternative digital signature techniques.

### 3.1.1 Proxy signature

In the proxy signature scheme, the original signer (such as boss) delegates her/his signing capability to the proxy signer (such as secretary), and the proxy signer creates a digital signature on behalf of the original signer. Proxy signature schemes resemble digital signature schemes except that they involve a proxy key generation, a proxy key



verification and a proxy signature-signing phase. In the proxy signature scheme, the original signer does not create a signing key by herself/himself alone. Instead, both the original signer and the proxy signer collaboratively generate the signing key.

Mambo et al. was first one that introduced the proxy signature scheme in 1996 [MUO96]. According to Mambo's scheme, there are three types of proxy signatures: full delegation, partial delegation, and delegation by warrant.

- **Full delegation:** In full delegation, the original signer gives hers/his private key to the proxy signer. In this case, the proxy signature created by the proxy signer is indistinguishable from the signature created by the original signer. This type is barely used for security issue.
- **Partial delegation:** In partial delegation, a proxy secret is derived from the original signer's private key; and the proxy secret is given to the proxy signer in a secure way. However, the processes from the original signer's private key to the proxy secret should be unilateral for security requirement.
- **Delegated by warrant:** When delegated by warrant, the proxy signer is authorized trustworthily to act on behalf of the original signer under certain conditions, such as a valid proxy signer and within the duration of delegation, etc.

### **Brief Description of Mambo's scheme**

We briefly describe Mambo's scheme. The participants are an original signer, a proxy signer and a verifier. The parameters,  $(p, q, g)$ , are public and are defined as follows.

- $p$  : a large prime number.
- $q$  : a prime divisor of  $p-1$ .
- $g$  : an element of  $Z_p^*$  with order  $q$ .

The basic protocol of Mambo's scheme uses the following algorithms:

### Proxy secret generation

The original signer selects a random number  $x$  as the private key, where  $1 \leq x < q$ . Also, the corresponding public key is  $y = g^x \bmod p$ . Then, the original signer publishes  $(p, q, g, y)$ .

### Proxy key generation

The original signer executes following steps to generate proxy key and forwards proxy key to proxy signer.

Step 1. Select a random number  $k_A \in Z_q^*$ .

Step 2. Compute  $r_A = g^{k_A} \bmod p$ .

Step 3. Set  $s_A = (x + k_A r_A) \bmod q$ .



Step 4. Forward  $(r_A, s_A)$  to the proxy signer in a secure manner.

Upon receiving the pair  $(r_A, s_A)$ , the proxy signer verifies validity of  $(r_A, s_A)$  by checking  $g^{s_A} \stackrel{?}{=} y \cdot r_A^{r_A} \bmod p$ . If the equality holds, then accepts the pair  $(r_A, s_A)$  and does the following steps; otherwise, rejects the pair. Thus, the proxy private key is  $s_A$ .

### Proxy signature signing

The proxy signer can sign a message  $m$  on behalf of the original signer by creating a signature with the proxy key  $s_A$ . The proxy signature is  $S(s_A, m)$ .

## Proxy signature verification

To verify the proxy signature  $S(s_A, m)$ , a verifier first replaces the proxy key  $s_A$  by  $y$  and  $r_A$  where  $g^{s_A} = y \cdot r_A^{r_A} \pmod p$ , and then checks  $V(y, r_A, S(s_A, m), m) = True$ .

First, the original signer creates a proxy secret  $s_A$  using her/his private key and forwards the proxy secret to a designated signer, called the proxy signer. Next, the proxy signer verifies validity of the proxy key pair  $(r_A, s_A)$  and then signs a message,  $m$ , and creates a signature  $S(s_A, m)$  using the proxy key  $s_A$ . Finally, a verifier verifies the validation of the proxy signature by checking  $V(y, r_A, S(s_A, m), m) = True$ .

Mambo's proxy signature fulfill following the requirements:

- (i) Verifiability: From a proxy signature, a verifier can be convinced that the original signer agrees on signing the message.
- (ii) Unforgeability: The designated proxy signer can create a valid proxy signature on behalf of the original signer.



Unfortunately, Mambo's proxy signature is not a proxy-protected signature scheme in which the original signer knows how to derive the proxy key on her/his own. On the contrary, in the proxy-protected proxy signature scheme, the original signer and proxy signer creates the proxy key interactively so that the original signer cannot derive the proxy key alone. Hence, Lee and Kim [LK99][LKK01a][LKK01b] proposed the concept of the strong proxy signature.

### 3.1.2 Strong proxy signature

Lee and Kim indicated that the strong proxy signature must fulfill following requirements [LK99]:

- (i) **Verifiability**: as mention above.
- (ii) **Strong unforgeability**: Only the designated proxy signer can create a valid proxy signature on behalf of the original signer. Any other people who are not designated as a proxy signer, the original signer included, cannot create a valid proxy signature.
- (iii) **Strong identifiability**: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.
- (iv) **Strong undeniability**: Once a proxy signer creates a valid proxy signature for an original signer, the proxy signer cannot repudiate his signature creation against anyone. This requirement is also called non-repudiation.

We describe the strong proxy signature scheme proposed by Lee *et al* [LK99], which is also based on discrete logarithm, as follows:

#### Proxy secret generation

An original signer selects a random number  $k_A$  and computes both  $r_A \equiv g^{k_A} \pmod{p}$  and  $s_A \equiv x_A h(m_w, r_A) + k_A \pmod{p-1}$ . Where  $p$ ,  $q$ , and  $g$  follow the definition as in section 3.1.1. The message warrant  $m_w$  indicates the relationship between the original signer and the proxy signer such as the identity of each protocol participant, the duration of delegation, and the usage of proxy key, etc. Then, the original signer sends  $(r_A, s_A, m_w)$  to a proxy signer 'P' in a secure manner.

### Proxy secret verification and proxy key generation

The proxy signer accepts the delegation if and only if  $g^{s_A} \equiv r_A y_A^{h(m_w, r_A)} \pmod{p}$ . Then, the proxy signer uses  $s_A$  to generate proxy key  $x_p \equiv s_A + x_B \pmod{q}$  and the implicitly public key  $y_p \equiv g^{x_p} \equiv y_A^{h(m_w, r_A)} y_B r_A \pmod{p}$ .

### Proxy signature signing

The proxy signer can use the proxy key  $x_p$  to create a signature  $m_{\text{sign}}$  on behalf of the original signer. Therefore, a valid proxy signature is  $(m, m_{\text{sign}}, m_w, y_A, y_B, r_A)$ .

### Verification of the proxy signature

Firstly, a verifier computes the proxy public key  $y_p' \equiv y_A^{h(m_w, r_A)} y_B r_A$  with parameters  $(m_w, y_A, y_B, r_A)$ ; and then accepts proxy signature if  $V(m, m_{\text{sign}}, y_p') \stackrel{?}{=} \text{true}$ .

Change in his dissertation point out that prevention of misuse is also an important requirement of proxy signature scheme [Chang05]; we describe as following:

- (v) **Prevention of misuse:** it is confident that proxy key should be used only for creating proxy signature conforming to delegation information. The proxy key pair cannot be used for other purposes.

### 3.1.3 Blind signature

Blind signature schemes, first introduced by Chaum [Cha83], are another digital signature form which allow a person to get a message signed without revealing any information about the message. In on-line vote, we would like to vote anonymously such

that no one knows whom we vote for. Similarly, in e-commerce environment, we would like to spend electronic cash under bank legitimation but prevent revealing our privacy to bank. Hence, the blind signature schemes with untraceability (also called unlinkability) are widely used in on-line vote and electronic cash applications.

Chaum demonstrated the implementation of this concept by using RSA signature. For example, Alice would like to have message  $m$  to be signed by Bob, but she does not want Bob to know any information about  $m$ . Let  $(n, e)$  be Bob's public key and  $(n, d)$  be his private key. Alice selects a random number  $r$  such that  $\text{gcd}(r, n) = 1$ , and sends  $x = r^e m \text{ mod } n$  to Bob. The random number  $r$  is “blinded” by the value  $x$ ; hence Bob can derive no useful information from  $x$ . Then, Bob returns the signed value  $t = x^d \text{ mod } n$  to Alice and Alice “un-blinds” the signed value  $t$  by computing  $s = r^{-1} t \text{ mod } n$  according to following equations:



$$\begin{aligned}
 s &= r^{-1} t \text{ mod } n \\
 &= r^{-1} x^d \text{ mod } n \\
 &= r^{-1} (r^e m)^d \text{ mod } n \\
 &= r^{-1} r^e m^d \text{ mod } n \\
 &= m^d \text{ mod } n
 \end{aligned}$$

Because of untraceability, the blind signature may be used for crimes such as blackmail or money laundry. Therefore, Stadler et al. proposed the fair blind signature scheme, which joined by a trustworthy third party to prevent signer’s forge attack and to trace doubtful message delivery in 1995 [SPC95]. Further work on blind signatures has been carried out in recent years such as efficient blind signature scheme based on QR [FL96] and proxy blind signature [TLT02][SH04][LA05] etc.

### 3.1.4 Lamport's One time signature

Lamport's one-time signature scheme contains three algorithms: key generation, signature signing and verification [Lam79]. Let  $h:Y \rightarrow Z$  denote a one-way hash function.

#### Key generation

1. Let  $y_{i,j} \in Y$  be randomly chosen, where  $1 \leq i \leq n$ ,  $j = 1, 0$ , and  $n$  is the length of message.
2. Compute  $z_{i,j} = h(y_{i,j})$ ,  $1 \leq i \leq n$  and  $j = 1, 0$ .
3. The key  $K$  consists of the  $2n$  private key  $y$ 's and the  $2n$  public key  $z$ 's shown as follows:

$$K = (y_{i,j}, z_{i,j} : 1 \leq i \leq n \text{ and } j = 1, 0)$$

$$y_{i,j} = \begin{pmatrix} y_{1,0} & y_{2,0} & \dots & y_{n,0} \\ y_{1,1} & y_{2,1} & \dots & y_{n,1} \end{pmatrix} \begin{img alt="University of Science and Technology of China logo" data-bbox="455 470 545 528"} = \begin{pmatrix} z_{1,0} & z_{2,0} & \dots & z_{n,0} \\ z_{1,1} & z_{2,1} & \dots & z_{n,1} \end{pmatrix}$$

#### Signature Signing

To sign a  $n$ -bit message  $m = m_1, \dots, m_n$ , the corresponding items of the message  $m_1, \dots, m_n$  are  $y_{1,m_1}, \dots, y_{n,m_n} : sig_K(m_1, \dots, m_n) = \{y_{1,m_1}, \dots, y_{n,m_n}\}$ .

For example, we want to sign a message  $m = 01 \dots 1$ . The signature is:

$$sig(m_1, \dots, m_n) = \begin{pmatrix} [y_{1,0}] & y_{2,0} & \dots & y_{n,0} \\ y_{1,1} & [y_{2,1}] & \dots & [y_{n,1}] \end{pmatrix} = (y_{1,0} \quad y_{2,1} \quad \dots \quad y_{n,1})$$

#### Verification

To verify signature  $(y_{1,m_1}, \dots, y_{n,m_n})$  on message  $m$ ,  $Ver_K(m_1, \dots, m_n, y_{1,m_1}, \dots, y_{n,m_n}) = \text{true}$  if and only if  $h(y_{i,m_i}) = z_{i,m_i}$  holds for  $1 \leq i \leq n$ .

The Lamport's one-time signature scheme needs large storage for signature and

public/private key pairs. The  $n$ -bit length message needs  $n$  signature items and  $2n$  items for both public and private keys (each of the item requires one hash value). For example, using SHA-160 as a one-way hash function, every one bit of message needs 160 bits signature, 2 public and 2 private items respectively; hence, for signing  $n$ -bit length message, Lamport's one-time signature scheme requires  $(n+2n+2n)*160$  bits storage.

## 3.2 Secret Sharing

The idea of secret sharing was invented independently by Adi Shamir [Sha79] and George Blakley [Bla79]. The secret sharing scheme is a method for distributing a secret among a group of participants, each of them takes a share of the secret. The secret can be only reconstructed when all (said  $n$ ) the shares or parts (said  $t$ , where  $t < n$ ) of the shares combined together; individual share will be useless.



### 3.2.1 Shamir ( $t, n$ ) - threshold scheme

Let  $t, n$  be positive integers, where  $t \leq n$ . A  $(t, n)$  - threshold scheme is a method of sharing a key among a set of  $n$  participants so that no less than  $t$  participants can reconstruct the key value. We describe Shamir  $(t, n)$  - threshold scheme in  $\mathbb{Z}_p^*$  as follows:

1. D (the dealer) chooses  $n$  distinct, nonzero elements in  $\mathbb{Z}_p^*$ , which are public and denoted as  $x_i$ , where  $1 \leq i \leq n$ . Then D gives the values  $x_i$  to  $P_i$ .
2. Suppose D wants to share a key  $K \in \mathbb{Z}_p^*$ . D secretly chooses (independently at random)  $t-1$  elements of  $\mathbb{Z}_p^*$ , said  $a_1, \dots, a_{t-1}$ .
3. D computes  $y_i = f(x_i)$ , where  $f(x) = K + \sum_{j=1}^{t-1} a_j x^j \pmod p$ , for  $1 \leq i \leq n$ ,
4. D gives the share  $y_i$  to  $P_i$ , for  $1 \leq i \leq n$ ,



Take  $K=13$  for example. To make four shares of the (3, 4)-threshold scheme, D chooses  $f(x) = x^2+2x+13$  and  $p = 17$ ; then the four shares are:  $s_1 = f(1) \bmod 17 \equiv 16$ ,  $s_2 = f(2) \bmod 17 \equiv 4$ ,  $s_3 = f(3) \bmod 17 \equiv 11$ ,  $s_4 = f(4) \bmod 17 \equiv 3$ . If we hold three participants  $s_1, s_2$  and  $s_3$ , then we can reconstruct the value of  $K$  by the following linear equations:

$$\begin{aligned} a \cdot 1^2 + b \cdot 1^1 + K \cdot 1^0 &= s_1 = 16, \\ a \cdot 2^2 + b \cdot 2^1 + K \cdot 2^0 &= s_2 = 4, \quad \text{and obtain } (a, b, K) = (1, 2, 13). \\ a \cdot 3^2 + b \cdot 3^1 + K \cdot 3^0 &= s_3 = 11, \end{aligned}$$

### 3.2.2 Verifiable Secret Sharing

Shamir secret sharing detects and tolerates Byzantine faults in a certain number of participants, but does not detect or tolerate errors on the part of the dealer. Fortunately, T. P. Pedersen proposed the Verifiable Secret Sharing (VSS) schemes in 1991 [Ped91] against Byzantine faults in both the dealer and the participants. Moreover, in the VSS scheme, the participants can generate the secret together without dealer.



A dealer may send incorrect shares to some or all of the participants, and the participants may submit incorrect shares during the reconstruction protocol. Therefore, in VSS scheme, let  $p$  be a large prime,  $q$  be a prime factor of  $p-1$ , and  $g$  be a generator of order  $q$  in  $\mathbf{Z}_p^*$ . Each participant  $P_i$ , where  $1 \leq i \leq n$ , generate a random polynomial  $f_i(x)$  of degree  $t$  over  $\mathbf{Z}_p^*$ . The constant coefficient of  $f_i(x)$  is  $P_i$ 's secret.

$$f_i(x) = a_{i,0} + \sum_{j=1}^{t-1} a_{i,j} x^j \pmod{q}$$

$P_i$  sends  $f_i(j)$  to  $P_j$ , where  $j = 1, \dots, n; i \neq j$ ) via the secure channel and publish the verification values  $\{g^{a_{i,0}}, g^{a_{i,1}} \dots g^{a_{i,t-1}}\}$ . Then, participant  $P_j$  verifies validity of its received share  $f_j(i)$  by

$$g^{f_j(i)} = \prod_{k=0}^{k=t} (g^{a_{j,k}})^{i^k} \pmod{p}$$

If the verification fails,  $P_j$  asks  $P_i$  to publish  $f_i(j)$ .  $P_i$  is disqualified if  $P_i$  does not posts an consistent  $f_i(j)$ .

### 3.3 Quadratic Residues

Fan and Lei first proposed efficient blind signature scheme based on QR in 1996 [FL96]. Our proxy signature based on QR is derived from Fan's signature scheme. Therefore, we describe several important QR mathematical properties as follows [Ros05]:

**Definition 3.1:** Let  $n$  be a positive integer. The integer  $y$  is a quadratic residue of  $n$  (denoted  $QR_n$ ) if  $\gcd(y, n)=1$  and the congruence  $x^2 = y \pmod{n}$  exists a solution. Otherwise,  $y$  is a quadratic nonresidue of  $n$ .



It is infeasible to compute the square root  $x$  when the exact factorization of  $n$  is unknown. In addition, the *Legendre* symbol  $\left(\frac{y}{p}\right)$  and *Jacobin* symbol  $\left[\frac{y}{n}\right]$  are useful to show whether an integer  $y$  is a quadratic residue. We describe as follows:

**Definition 3.2:** Let  $p$  be an odd prime and  $y$  be an integer. The *Legendre symbol*  $\left(\frac{y}{p}\right)$

defines as:

1.  $\left(\frac{y}{p}\right) = 0$  if  $p|y$ ,
2.  $\left(\frac{y}{p}\right) = 1$  if  $y \in QR_n$ , and
3.  $\left(\frac{y}{p}\right) = -1$  if  $y \in$  quadratic nonresidue mod  $n$ .

**Definition 3.3:** Let  $n \geq 3$  and  $n$  be an odd integer with prime factorizations  $n = \prod_{i=1}^k p_i^{e_i}$ .

Then the *Jacobin symbol*  $\left[ \frac{y}{n} \right]$  is defined as  $\prod_{i=1}^k \left[ \frac{y}{p_i} \right]^{e_i}$

If  $n$  is prime, then the *Jacobin symbol* is reduced to the *Legendre symbol*.

**Definition 3.4:** A natural number  $n$  is a *Blum integer* if  $n = pq$  where  $p$  and  $q$  are prime numbers that are congruent to 3 mod 4.

If  $n$  is *Blum integer*, each quadratic residue has exactly four square roots, one of which is also a square. For example, one square root of 139 mod 437 is 24; the other three are 185, 252, and 413 [Sch96]. In addition, it is computationally infeasible to solve the root of quadratic residue without knowing any information of  $n$ .



## 3.4 Digital signature standard

### 3.4.1 DSA

DSA has become FIPS 186 in August 1991; also called DSS. DSA is a variant of the Schnorr [Sch90] and ElGamal [EIG85] signature algorithms. The algorithm of DSA uses the following parameters [NIST00] and publishes the first three parameters:  $p$ ,  $q$ , and  $g$ :

1.  $p =$  a prime modulus, where  $2^{L-1} < p < 2^L$  for  $512 \leq L \leq 1024$  and  $L$  is a multiple of 64.
2.  $q =$  a prime divisor of  $p - 1$ , where  $2^{159} < q < 2^{160}$ .
3.  $g = a^{(p-1)/q} \bmod p$ , where  $a$  is any integer with  $1 < a < p - 1$  such that  $a^{(p-1)/q} \bmod p > 1$  ( $g$  has order  $q \bmod p$ ).
4.  $x =$  a randomly or pseudo-randomly generated integer with  $0 < x < q$ , denoted as

private key.

5.  $y = g^x \bmod p$ , denoted as public key.
6.  $k$  = a randomly or pseudo-randomly generated integer with  $0 < k < q$ .

The algorithm of DSA also uses a one-way hash function,  $h(m)$ , SHA-160 as described in section 2.4. To sign a message  $m$ :

1. Alice selects a random number,  $k$ , less than  $q$ .
2. Alice generates

$$r = (g^k \bmod p) \bmod q$$

$$s = (k^{-1}(h(m) + xr)) \bmod q$$

where  $r$  and  $s$  are her signature sent to Bob.

3. Bob verifies the signature by computing:

$$w = s^{-1} \bmod q$$

$$u_1 = (h(m) * w) \bmod q$$

$$u_2 = (rw) \bmod q$$

$$v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$$

If  $v = r$ , then Bob accepts the signature.



### DSA Correctness Analysis

We start DSA correctness analysis with a lemma 3.1 to show that  $g^q \bmod p = 1$ .

**Lemma 3.1:** Let  $p$  and  $q$  be primes so that  $q$  divides  $p - 1$ ,  $h$  a positive integer less than  $p$ ,

and  $g = a^{(p-1)/q} \bmod p$ . Then  $g^q \bmod p = 1$ , and if  $m \bmod q = n \bmod q$ , then

$$g^m \bmod p = g^n \bmod p.$$

*Proof:*

$$g^q \bmod p$$

$$= (a^{(p-1)/q} \bmod p)^q \bmod p$$

$$= a^{(p-1)} \bmod p$$

$$= 1$$

by Fermat's Little Theorem. Let  $m \bmod q = n \bmod q$ , i.e.,  $m = n + kq$  for some integer  $k$ . Then

$$g^m \bmod p = g^{n+kq} \bmod p$$

$$= (g^n g^{kq}) \bmod p$$

$$= ((g^n \bmod p) (g^q \bmod p)^k) \bmod p$$

$$= g^n \bmod p$$

$$\text{since } g^q \bmod p = 1.$$

□

**Theorem 3.1:** If  $m' = m$ ,  $r' = r$ , and  $s' = s$  in the signature verification, then  $v = r'$ .

*Proof:*

$$w = s'^{-1} \bmod q = s^{-1} \bmod q$$



$$u_1 = (h(m') * w) \bmod q = (h(m) * w) \bmod q$$

$$u_2 = (r'w) \bmod q = (rw) \bmod q$$

Now  $y = g^x \bmod p$ , so that by the *lemma 3.1*,

$$v = (g^{u_1} * y^{u_2} \bmod p) \bmod q$$

$$= (g^{h(m)*w} * y^{rw} \bmod p) \bmod q$$

$$= (g^{h(m)*w} * g^{xrw} \bmod p) \bmod q$$

$$= (g^{(h(m)+xr)*w} \bmod p) \bmod q.$$

Also

$$s = (k^{-1}(h(m)+xr)) \bmod q.$$

Hence

$$w = (k(h(m)+xr)^{-1}) \bmod q$$

$$(h(m)+xr)^*w \bmod q = k \bmod q.$$

Thus by the above lemma,

$$\begin{aligned} v &= (g^{(h(m)+xr)^*w} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q \\ &= r \\ &= r'. \end{aligned}$$

□

By Theorem 3.1, a verifier can check the valid of signature correctly.

### 3.4.2 ECDSA

ECDSA is counterpart of DSA and operates on elliptic curve group  $E(\mathbf{Z}_p^*)$ . ECDSA was invented in 1985 and was accepted as FIPS 186-2 [NIST00] IEEE standards in 2000. It was also accepted as an ISO standard in 1998. We describe key generation, signature, and verification for ECDSA as follows:



#### ECDSA key Generation

1. Selects an elliptic curve  $E$  over  $\mathbf{Z}_p^*$ .
2. Select a point  $P \in E(\mathbf{Z}_p^*)$  where order is also prime  $q$ .
3. Select a statistically unique and unpredictable integer  $d$  in the interval  $[1, q-1]$ .
4. Compute  $Q = dP$ .
5. The public key is  $(E, P, q, Q)$ ; the private key is  $d$ .

#### ECDSA Signature Generation

1. Select a statistically unique and unpredictable integer  $k$  in the interval  $[1, q-1]$ .
2. Compute  $kP = (x_1, y_1)$  and  $r \equiv x_1 \bmod q$ . If  $r = 0$ , then go to step 1.
3. Compute  $k^{-1} \bmod q$ .

4. Compute  $s = k^{-1}[h(m)+dr] \bmod q$ .
5. If  $s = 0$ , then go to step 1.
6. The signature for the message  $m$  is the pair of integers  $(r, s)$ .

### ECDSA Signature Verification

1. Obtain an authentic copy of Alice's public key  $(E, P, q, Q)$ .
2. Verify that  $r$  and  $s$  are integers in the interval  $[1, q-1]$ .
3. Compute  $w \equiv s^{-1} \bmod q$  and  $h(m)$ .
4. Compute  $u_1 \equiv h(m)w \bmod q$  and  $u_2 \equiv rw \bmod q$ .
5. Compute  $u_1P + u_2Q = (x_0, y_0)$  and  $v \equiv x_0 \bmod q$ .
6. Accept the signature if and only if  $v = r$ .



**Theorem 3.2:** If the signature of ECDSA is valid, then  $v = r$ .

*Proof:*

$$s = k^{-1}[h(m)+dr] \bmod q, \text{ hence } k = s^{-1}[h(m)+dr] \bmod q$$

$$(x_1, y_1) = kP = s^{-1}[h(m)+dr]P$$

$$(x_0, y_0) = u_1P + u_2Q = h(m)wP + rwdP$$

$$= [h(m)s^{-1} + rs^{-1}d]P$$

$$= s^{-1}[h(m)+dr]P = (x_1, y_1)$$

$$\text{Therefore, } v = x_0 \bmod q$$

$$= x_1 \bmod q$$

$$= r$$

□

By theorem 3.2, we verify the signature of ECDSA through the equation  $v = r$ .

## Chapter 4 The Proposed Proxy Signatures

### 4.1 Proxy Signature based on Digital Signature Algorithm

DSA and ECDSA are pretty well known by their security properties so that they have been chosen as standard signature schemes. However, they both lack functionality of proxy. Most of the proxy signature schemes, which have been proposed prior to this date, are not based on standard signature such as DSA or ECDSA and have been considered infeasible because of their obvious security weaknesses.

In this section, we carefully modify the DSA/ECDSA to be a proxy-protected proxy signature scheme to fulfill the strong proxy signature requirements. Although proxy-protected proxy signature scheme becomes more time-consuming for creating the proxy key interactively between the original signer and proxy signer, the proxy-protected scheme ensures that the original signer cannot derive the proxy key on her/his own; therefore, the proxy signer will not be betrayed.

Actually, most proposed proxy signature schemes cannot be proven sufficiently strong, secure, and unbreakable in order to against some unknown intentional attacks; in addition, they are not based on standard signature. In fact, all that the proposed proxy signature schemes can do till now is to demonstrate the scheme's power against some existing attacks; however, it occurs often that there will be always a new attack invented exactly against these schemes [LC03].

To conquer those disadvantages; therefore, we propose a proxy-protected signature scheme combining with standard signature DSA/ECDSA [NIST00] which are pretty well



known by their security properties to reinforce the proxy signature. Combining DSA/ECDSA, proxy signature and PKI mechanism, this work could be more useful in practice.

#### 4.1.1 Proxy Signature Based on Digital Signature Algorithm

The SHA-serials [NIST02] are used in our scheme and the participants of our scheme include an original signer ‘Alice’, a proxy signer ‘Bob’, and a verifier. Suppose that a Certificate Authority (CA) certifies Alice and Bob enrolls proxy key into the PKI when a proxy key is created with the original signer interactively. The useful notations we list as follows:

*Alice* An original signer



*Bob* A proxy signer

$p$  A prime number, where  $2^{L-1} < p < 2^L$  for  $512 \leq L \leq 1024$  and  $L$  is a multiple of 64

$q$  A prime divisor of  $p - 1$  in DSA, A prime number, where  $2^{159} < q < 2^{160}$  and is the order of points over  $E$  in ECDSA

$g$   $a^{(p-1)/q} \bmod p$ , where  $a$  is an integer with  $1 < a < p - 1$  such that  $a^{(p-1)/q} \bmod p > 1$

$x$  A pseudo-randomly generated integer with  $0 < x < q-1$ , denoted as private key.

$y$   $g^x \bmod p$ , denoted as public key in DSA

$k$  A randomly or pseudo-randomly generated integer with  $0 < k < q$ .

$E$  An elliptic curve defined over  $F_p$

$G$  A point over  $E$  having prime order  $q$

$Q$  A public key with  $Q = xG$  over  $E$

$h()$  A one-way hash function, SHA-160[NIST02]

There are four algorithms in proposed schemes shown as follows:

1. Proxy generation and delivery
2. Proxy verification and proxy key generation
3. Signing by proxy signer
4. Verification of Proxy signature

In addition, there are two approaches to implement the proxy signature based on digital signature algorithms DSA and ECDSA respectively. First at all, we describe the proxy signature based on DSA in next section. Besides, we use X.509v3 certificate extension [RSA00] to indicate the relationship between an original signer and the proxy signer by proxy parameters, and the PKI mechanism can avoid man-in-middle attack [MOV96].



#### 4.1.2 Proxy Signature Based on DSA

At initialization step, the CA or Registration Authority (RA) verifies the relationship of the delegation. The four algorithms we show as follows:

##### **Proxy generation and delivery**

1. Bob selects a random  $\sigma \in Z_q^*$ , where  $\gcd(\sigma, p-1) = 1$  and computes  $g' = g^\sigma \bmod p$ .  
Then, Bob sends  $g'$  to Alice.
2. After receiving  $g'$ , Alice selects a random  $k_A \in Z_q^*$ , computes, publishes  $r_A = g'^{k_A} \bmod p$ , and sets  $e = h(g'^{k_A} \bmod p) \bmod q$  and  $s_A = (xe + k_A) \bmod q$ . Then, Alice sends  $(r_A, s_A)$  to Bob. The pair  $(r_A, s_A)$  is a delegation proxy certificate for proving that Alice delegates her signing capacity to Bob.

### Proxy verification and proxy key generation

1. On receiving  $(r_A, s_A)$ , Bob computes  $e' = h(r_A^\sigma \bmod p) \bmod q$  and verifies the validity by checking if  $r_A = (g^{s_A} y^{-e'} \bmod p) \bmod q$ .
2. If the equation  $r_A = (g^{s_A} y^{-e'} \bmod p) \bmod q$  holds, Bob sets  $s_B = s_A \sigma^{-1} \bmod q$  as a proxy key, sets  $(s_B, g^{s_B} \bmod p)$  as public key pairs and sends the certificate request [RSA00] to the RA.
3. According to certificate policy, RA identifies Bob and then forwards the certificate request to the CA for signing proxy certificate. The process of proxy certificate generating is shown in Fig 4.1.

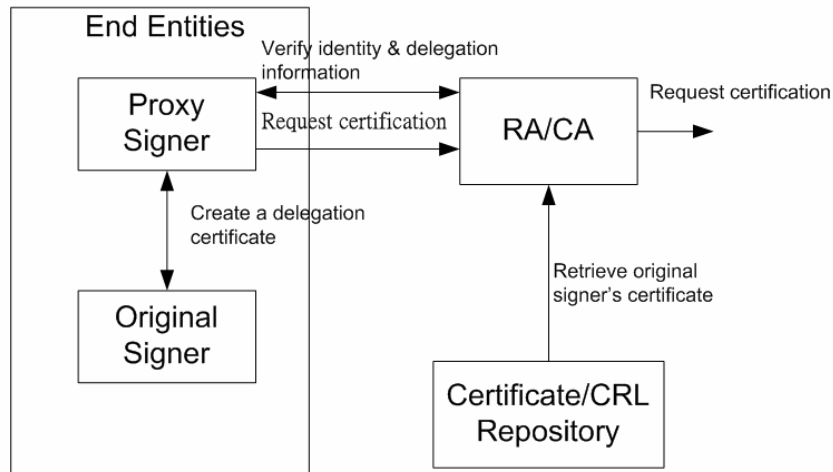


Fig 4.1 Proxy signer initialization in PKI

### Signing by proxy signer

To sign a message  $m$ , Bob should do the following steps:

1. Select a random  $k \in Z_q^*$ .
2. Compute  $r = (g^{tk} \bmod p) \bmod q$
3. Set  $s = k^{-1}(h(m) + s_B r) \bmod q$ .
4. If  $s = 0$  then re-select another random number  $k$  and run again.

The proxy signature is the tuple  $(g', e', r, s)$ .

## Verification of Proxy signature

To verify the proxy signature  $(g', e', r, s)$  on message  $m$ , a verifier should:

1. Query repository and check if the certificate of proxy key is valid.
2. Get and verify valid of  $r_A$ .
3. Verify that  $1 \leq r \leq q$  and  $1 \leq s \leq q$ ; if not reject the signature.
4. Compute  $w = s^{-1} \bmod q$ .
5. Compute  $u_1 = w \cdot h(m) \bmod q$ ,  $u_2 = rw \bmod q$ , and  $u_3 = e'u_2 \bmod q$ .
6. Compute  $v = (g^{u_1} r_A^{u_2} y^{u_3} \bmod p) \bmod q$ .

Accept the signature if  $v = r$ .

We consider that proxy signature based on DSA scheme can be deployed in both the DSA with proxy signature capability and the conventional DSA. This scheme can be a conventional DSA if taking the parameters  $g' = g$ ,  $r_A = 1$  and  $e' = 1$ . Therefore, this scheme is generalized DSA and can also be used in conventional DSA.



### 4.1.2.1 Correctness of proxy signature scheme based on DSA

In this section, we will prove the correctness of the proxy signature scheme based on DSA scheme.

**Theorem 4.1:** If the delegation certificate  $(r_A, s_A)$  is valid, it will pass the verification  $r_A = g^{s_A} y^{-e'} \bmod p$ .

Proof:

Firstly, we proof that  $e = e'$

$$e = h(g^{r_A} \bmod p) \bmod q \text{ and } e' = h(r_A^\sigma \bmod p) \bmod q$$

$$\therefore g^{r_A} \bmod p = (g^\sigma)^{r_A} \bmod p = (g^{k_A})^\sigma \bmod p = r_A^\sigma \bmod p$$

Therefore,  $e = e'$ .

$$\therefore s_A = (xe + k_A) \bmod q;$$

Substitute  $e'$  for  $e$ , then we obtain

$$s_A = (xe' + k_A) \bmod q.$$

Rearrange the above equation as

$$k_A = (s_A - xe') \bmod q.$$

Raise both sides by  $g$ ; by *lemma 3.1* we know that  $g^q \bmod p = 1$ .

$$g^{k_A} = g^{(s_A - xe')} \bmod p,$$

$$r_A = (g^{s_A} \cdot g^{-xe'}) \bmod p (\because r_A = g^{k_A} \bmod p)$$

$$r_A = (g^{s_A} y^{-e'}) \bmod p (\because y = g^x \bmod p)$$

Thus,  $r_A = (g^{s_A} y^{-e'}) \bmod p$  as required. □



**Theorem 4.2:** If the proxy signer generates the proxy signature correctly, it will pass the proxy signature verification.

Proof:

We have a valid proxy signature  $s = k^{-1}(h(m) + s_B r) \bmod q$ .

Rearrange the signature as

$$k = s^{-1} (h(m) + s_B r) \bmod q$$

$$k = s^{-1} (h(m) + s_A \sigma^{-1} r) \bmod q. (\because s_B = s_A \sigma^{-1} \bmod q)$$

$$k = s^{-1} [h(m) + (xe + k_A) \sigma^{-1} r] \bmod q. (\because s_A = (xe + k_A) \bmod q)$$

Raise both sides by  $g'$

$$g'^k = (g'^{s^{-1}h(m)} g'^{k_A \sigma^{-1} r s^{-1}} g'^{x e \sigma^{-1} r s^{-1}} \bmod p) \bmod q.$$

Substitute following notations respectively:

$$g'^k = r, \quad g'^{k_A \sigma^{-1}} = g^{k_A} = r_A \quad \text{and} \quad g'^{x \sigma^{-1}} = g^x = y (\because g' = g^\sigma \bmod p)$$

$$r = (g^{s^{-1}h(m)} r_A^{rs^{-1}} y^{ers^{-1}} \bmod p) \bmod q.$$

Let  $w = s^{-1} \bmod q$ ,  $u_1 = w \cdot h(m) \bmod q$ ,  $u_2 = rw \bmod q$ , and  $u_3 = e'u_2 \bmod q$ .

We yield the equation:

$$r = (g^{u_1} r_A^{u_2} y^{u_3} \bmod p) \bmod q \text{ as required.} \quad \square$$

A verifier has to use both the original signer's public key and proxy key certificate to verify the proxy signature. Since the proxy key is created interactively between original signer and proxy signer, a verifier can be aware of the agreement upon signing the message from the original signer. This property obeys the definition of *verifiability*; and by theorem 4.1 and 4.2, a verifier can check the valid of proxy signature.



### 4.1.3 Proxy Signature based on ECDSA

ECDSA, a DSA based on the ECC, was invented in 1985 and accepted as FIPS 186-2 in 2000 [NIST00]. In this section, we introduce the proxy-protected signature based on ECDSA, which is a variant ECDSA with properties of strong proxy signature. An elliptic curve  $E$  modulo a prime  $p$  denotes as a public-key cryptography. The operation of elliptic curve could be referred to [IEEE05]. We describe the protocol of proxy-protected ECDSA as follows. First we let Alice have private key  $x$  and public key  $Q = xG$  certificated by a certificate authority. Bob is a designated proxy signer.

#### **Proxy generation and delivery**

*Bob:* Select a random number,  $k_o$  ( $1 < k_o < q$ ).

Compute  $G' = k_o G \bmod q$ .

*Bob* → *Alice*  $G'$ .

*Alice*: Select a random number,  $k_A$  ( $1 < k_A < q$ )

Compute and publish  $R_A = k_A G$ .

Set  $(x_1, y_1) = k_A G'$ .

Compute  $e = x_1 \bmod q$  and set  $s_A = (xe + k_A) \bmod q$ .

If  $x_1 = 0$ , then re-select  $k_A$  and run again.

*Alice* → *Bob*  $(R_A, s_A)$ .

### **Proxy verification and proxy key generation**

*Bob*: Let  $(x_2, y_2) = k_o R_A$ , and set  $e' = x_2 \bmod q$ .

Accept the delegation if and only if  $R_A = s_A G - e' Q$ .

Once Bob accepts the delegation, he will compute  $s_B = s_A k_o^{-1} \bmod q$  as a proxy key; and will send the certificate request [RSA00] to the RA. According to certificate policy, RA identifies Bob and then forwards the certificate request to the CA for signing proxy certificate.

### **Signing by the proxy signer**

*Bob*: Select a random number  $k$  ( $1 < k < q$ ).

Compute  $(x_3, y_3) = k G'$ .

Set  $r = x_3$ .

Compute  $s = k^{-1}(h(m) + s_B r) \bmod q$ .

If  $r = 0$  or  $s = 0$  then re-select another random number  $k$  and run again.

The proxy signature for the message  $m$  is  $(G', e', r, s)$ .

### Verification of the proxy signature

Carol: Get and verify  $R_A$ .

Verify that  $r$  and  $s$  are integers in interval  $[1, q-1]$ .

Compute  $w = s^{-1} \bmod q$ .

Compute  $u_1 = h(m)w \bmod q$ .

Compute  $u_2 = rw \bmod q$ .

Compute  $u_3 = e'u_2 \bmod q$ .

Compute  $X = (x_3', y_3') = u_1G' + u_2R_A + u_3Q$ .

If  $X = O$ , then reject the signature, else accept the signature if and only if

$$x_3' = x_3 = r.$$

The proxy-protected ECDSA could be also deployed in ECDSA by taking parameters  $G' = G$ ,  $R_A = 0$  and  $e' = 1$ . Furthermore, the proxy-protected ECDSA also maintains the properties of strong proxy signature [LK99][LKK01a][LKK01b].



#### 4.1.3.1 Correctness of proxy signature scheme based on ECDSA

In this section, we will prove the correctness of the proxy signature scheme based on ECDSA scheme.

**Theorem 4.3:** If the delegation certificate  $(R_A, s_A)$  is valid, then  $R_A = s_A G - e'Q$ , where

$$R_A = k_A G, s_A = (xe + k_A) \bmod q, (x_2, y_2) = k_o R_A, \text{ and } e' = x_2 \bmod q.$$

Proof:

Firstly, we proof that  $e = e'$

$$(x_1, y_1) = k_A G' = k_A k_o G = k_o R_A G = k_o R_A = (x_2, y_2);$$



Hence,  $e = x_1 \bmod q$

$$= x_2 \bmod q$$

$$= e'$$

$\therefore s_A = (xe + k_A) \bmod q$ ;

Substitute  $e'$  for  $e$  in above equation, then we obtain

$$s_A = (xe' + k_A) \bmod q.$$

Rearrange the above equation as

$$k_A = (s_A - xe') \bmod q.$$

Multiple  $G$  on both sides

$$R_A = k_A G$$

$$= (s_A - xe') G$$

$$= s_A G - e' x G$$

$$= s_A G - e' Q$$



□

**Theorem 4.4:** If the proxy signer generates the proxy signature correctly, it will pass the proxy signature verification.

Proof:

We have a valid proxy signature  $s = k^{-1}(h(m) + s_B r) \bmod q$ .

Rearrange the signature as

$$k = s^{-1} (h(m) + s_B r) \bmod q$$

$$k = s^{-1} (h(m) + s_A k_0^{-1} r) \bmod q. (\because s_B = s_A k_0^{-1} \bmod q)$$

$$k = s^{-1} [h(m) + (xe + k_A) k_0^{-1} r] \bmod q. (\because s_A = (xe + k_A) \bmod q)$$

Multiple  $G'$  on both sides

$$k G' = s^{-1} G' [h(m) + (xe + k_A) k_0^{-1} r]$$

$$\begin{aligned}
&= s^{-1}G'h(m) + s^{-1}xeG' k_0^{-1}r + s^{-1}k_A G' k_0^{-1}r \\
&= s^{-1}h(m)G' + s^{-1}xeGr + s^{-1}k_A Gr (\because G' k_0^{-1} = G) \\
&= u_1G' + u_2xeG + u_2k_A G (\because u_1 = h(m)w = h(m)s^{-1}; u_2 = rw = rs^{-1}) \\
&= u_1G' + u_2xe'G + u_2R_A (\because e = e'; R_A = k_A G) \\
&= u_1G' + u_2R_A + u_3Q (\because u_3 = e'u_2G; Q = xG) \quad \square
\end{aligned}$$

A verifier has to use both the original signer's public key and proxy key certificate to verify the proxy signature. This proof show that the proxy signature scheme based on ECDSA fulfills *verifiability* property; and by theorem 4.3 and 4.4, a verifier can check the valid of proxy signature.

#### 4.1.3.2 Proxy Signature based on ECDSA example demonstration



In some reports concerning security estimation, the elliptic curve based on cryptosystem will be secure till the year 2020. It has been suggested that one should take  $p \approx 2^{160}$ . In this section we work through a tiny example to illustrate the computations in the proxy-protected ECDSA.

Let  $E$  be the elliptic curve  $y^2 = x^3 + x + 4$  over  $\mathbf{Z}_{19}^*$ . The parameter  $q$  is the number of points in  $E$ , also called order of  $E$  over  $\mathbf{Z}_{19}^*$ . We first compute  $x^3 + x + 4 \pmod{19}$  for  $x \in \mathbf{Z}_{19}^*$ , and then try to solve the above equation for  $y$ ; and set  $z = x^3 + x + 4 \pmod{19}$  and test if  $z$  is a quadratic residue (QR), by Euler's criterion. If the modulo prime  $p = 3 \pmod{4}$ , we could yield the square roots of a quadratic residue  $z$  as following formula:

$$\pm z^{(19+1)/4} \pmod{19} = \pm z^5 \pmod{19}.$$

The results of the computing are listed in Table 4.1.

Table 4.1 Points on the elliptic curve  $x^3 + x + 4 \pmod{19}$

$x$	$z=x^3 + x + 4 \pmod{19}$	$y' = \pm z^5 \pmod{19}$	$(y')^2$	$y$	Is QR?
0	4	17,2	4	17,2	√
1	6	5,14	6	5,14	√
2	14	10,9	5		
3	15	2,17	4		
4	15	2,17	4		
5	1	1,18	1	1,18	√
6	17	6,13	17	6,13	√
7	12	8,11	7		
8	11	7,12	11	7,12	√
9	1	1,18	1	1,18	√
10	7	11,8	7	11,8	√
11	16	4,15	16	4,15	√
12	15	2,17	4		
13	10	3,16	9		
14	7	11,8	7	11,8	√
15	12	8,11	7		
16	12	8,11	7		
17	13	14,5	6		
18	2	13,6	17		

Because  $G$  is a generator, we can take the generator  $G = (1,5)$ ; and compute the remaining multiples of  $G$  by applying the addition operation on  $E$ .

The addition operation on  $E$  is defined as follows:

Suppose  $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$  are the points on  $E$ . If  $x_2 = x_1$  and  $y_2 = -y_1$ , then  $p_1 + p_2 = O$  where  $O$  is a special point, called point at infinity; otherwise

$P_1 + P_2 = (x_3, y_3)$ , where

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1,$$

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1}, & \text{if } P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1}, & \text{if } P = Q \end{cases}, \text{ and}$$

Therefore, the next multiple is  $2G = G + G$ ,  $3G = 2G + G$ , and so on. The results of these computations are tabulated in Table 4.2.

Table 4.2 The multiples of generator  $G$

$G = (1,5)$	$2G = (5,1)$	$3G = (14,8)$	$4G = (9, 18)$	$5G = (6,13)$
$6G = (10,11)$	$7G = (0,2)$	$8G = (8,12)$	$9G = (11,4)$	$10G = (11,15)$
$11G = (8,7)$	$12G = (0,17)$	$13G = (10,8)$	$14G = (6,6)$	$15G = (9,1)$
$16G = (14,11)$	$17G = (5,18)$	$18G = (1,14)$	$19G = O$	

Suppose that Alice's private key  $x$  is 3, so the public key is  $Q = 3G = (14, 8)$ .

### Proxy generation and delivery

Bob : Select a random number,  $k_o$ , said 5;

and compute  $G' = k_o G = 5G = (6, 13)$



Bob → Alice  $G'$

Alice: Select a random number,  $k_A$ , said 4;

and compute  $R_A = k_A G = 4G = (9, 18)$ ;

set  $k_A G' = 4(6, 13) = (1, 5) = (x_1, y_1)$ .

Suppose that  $e = x_1 = 1$ . Alice computes

$$s_A = (xe + k_A) \bmod q = (3 \cdot 1 + 4) \bmod 19 = 7 \text{ and}$$

forward  $(R_A, s_A) = [(9, 18), 7]$  to Bob.

### Proxy verification and proxy key generation

Let  $(x_2, y_2) = k_o R_A = 5(9, 18) = (1, 5)$ , and  $e' = x_2 = 1$ .

$$\begin{aligned} & s_A G - e' Q \\ &= 7 \cdot (1, 5) - 1 \cdot (14, 8) \\ &= (9, 18) \end{aligned}$$

$$= R_A$$

Then, Bob accepts the delegation because  $R_A = s_A G - e'Q$ .

The proxy key is:

$$s_B = s_A k_o^{-1} \bmod q = 7 \cdot 5^{-1} \bmod 19 = 7 \cdot 4 \bmod 19 = 9,$$

and we omit the process of enrolls proxy key into the PKI mechanism.

### Signing by the proxy signer

Suppose that the message is  $m$ ,  $h(m) = 8$  and  $k = 13$ . To sign the message, Bob computes

$$(x_3, y_3) = kG' = 13 \cdot (6, 13) = (8, 12),$$

sets  $r = x_3 = 8$  and creates proxy signature,

$$s = k^{-1}(h(m) + s_B r) \bmod q = 13^{-1}(8 + 9 \cdot 8) \bmod 19 = 3 \cdot 80 \bmod 19 = 12.$$

The proxy signature is  $(G', e', r, s) = [(6, 13), 1, 8, 12]$ .



### Verification of the proxy signature

The verifier does the following processes:

Get and verify  $R_A = (9, 18)$ .

$$w = s^{-1} \bmod q = 12^{-1} \bmod 19 = 8,$$

$$u_1 = h(m)w \bmod q = 8 \cdot 8 \bmod 19 = 7,$$

$$u_2 = rw \bmod q = 8 \cdot 8 \bmod 19 = 7,$$

$$u_3 = e'u_2 \bmod q = 1 \cdot 7 \bmod 19 = 7,$$

and

$$\begin{aligned} X &= (x_3', y_3') = u_1 G' + u_2 R_A + u_3 Q \bmod q \\ &= 7 \cdot (6, 13) + 7 \cdot (9, 18) + 7 \cdot (14, 8) \bmod 19 \\ &= (35 + 28 + 21)G \\ &= (8, 12). \end{aligned}$$

The verifier accepts the signature, because  $x_3' = 8 = x_3$ . This example adequately shows the proxy-protected ECDSA can be used in practice. Nevertheless, the security of the proxy-protected ECDSA is as secure as the standard signature ECDSA that we discuss the security of proxy-protected DSA/ECDSA in next session.

#### 4.1.4 Security analysis and comparisons

The security of the proposed scheme is based on the difficulty of breaking a one-way hash function as well as the hardness of three discrete logarithm problems. One of the discrete logarithms is in  $\mathbf{Z}_q^*$  where the powerful index-calculus methods applied; the second one is in the cyclic subgroup of order  $q$  [MOV96]; and the third one is elliptic curve discrete logarithm problem. In this section, we discuss several possible attacks against the security of proposed schemes.



##### **Attack Scenario 1:**

##### **I. Proxy signature based on DSA part:**

If an attacker might forge the proxy signature on the message  $m$  by selecting a random number  $k$ ; and computing  $r = g^{tk} \pmod p$ ; the attacker needs proxy key  $s_B = \sigma^{-1}(xe + k_A) \pmod q$ ,  $k$  to forge signature  $s = k^{-1}(h(m) + s_B r) \pmod q$ . It is computationally infeasible to determine  $s$  without both  $s_B$  and correct  $k$  under the assumption of the discrete logarithm problem [MOV96]. In addition, the probability of successful conjecture of both  $s_B$  and correct  $k$  is  $1/q$ , which is negligible when  $q$  is large enough. Furthermore, the attacker does not have proxy certificate to pass verification.

##### **II. Proxy signature based on ECDSA part:**

If an attacker might forge the proxy signature on the message  $m$  by selecting a random number  $k$ ; and computing  $(x_3, y_3) = kG'$  and setting  $r = x_3$ ; the attacker needs proxy key  $s_B = s_A k_0^{-1} \bmod q$ ,  $k$  to forge signature  $s = k^{-1}(h(m) + s_B r) \bmod q$ . It is computationally infeasible to determine  $s$  without both  $s_B$  and correct  $k$  under the assumption of the elliptic curve problem [IEEE05]. In addition, the probability of  $s_B$  and correct  $k$  is  $1/q$ , which is negligible when  $q$  is large enough.

### Attack Scenario 2:

Suppose that another malicious signer impersonates the authorized proxy signer to create a proxy key interactively with an original signer (man-in middle attack) by selecting another random  $\sigma$  (or  $k_o$  in ECDSA). To prevent this attack, we require only the certificate of original/proxy signer's public keys by any kind of authority mechanism such as PKI mechanism. With the verification of public keys' certificate, the verifier will reject all unauthorized proxy keys generated by the fake proxy signer.

### Attack Scenario 3:

#### I. Proxy signature based on DSA part:

If a dishonest original signer attempts to forge the proxy key, the proxy signer could use a blind factor  $\sigma$  to blind  $g' = g^\sigma \bmod p$  so that the original signer needs to solve  $\sigma$  from  $g' = g^\sigma \bmod p$ . It is difficult to determine  $\sigma$  according to the hardness of the discrete logarithm problem [MOV96].

#### II. Proxy signature based on ECDSA part:

If a dishonest original signer attempts to forge the proxy key, the proxy signer could use a blind factor  $k_o$  to blind  $G' = k_o G \bmod q$  so that the original signer needs to solve  $k_o$  from  $G' = k_o G \bmod q$ . It is difficult to determine  $k_o$  according to the hardness of the elliptic curve problem [IEEE05].

Under so-called ‘*proxy-protected*’ security property restriction, an original signer should not be able to derive the authorized proxy signer’s proxy key; otherwise a verifier could not distinguish exactly whether the original signer or the proxy signer creates the proxy signature.

#### Attack Scenario 4:

##### I. Proxy signature based on DSA part:



**Theorem 4.5:** If a malicious proxy signer attempts to impersonate an original signer to create a delegation certificate, then the malicious proxy signer can derive the secret key of original signer.

Proof:

On the other hand, if a malicious proxy signer attempts to impersonate an original signer to create a delegation certificate, the malicious proxy signer selects a random number  $k_A$  and computes  $r_A = g^{k_A} \bmod p$ , and  $e' = h(g^{k_A}) \bmod p$ .

If a malicious proxy signer can create a delegation certificate, she/he must know  $s_A$  to pass  $r_A = (g^{s_A} y^{-e'} \bmod p) \bmod q$  in the proxy verification phase and the proxy key

$$s_B = s_A \sigma^{-1} \bmod q \text{ is also derive from } s_A.$$

$s_A = (xe + k_A) \bmod q$ ; hence the malicious proxy signer can derive:

$$x = e^{-1}(s_A - k_A) \bmod q.$$

□



From theorem 4.5, if the malicious proxy signer can create a delegation certificate under just knowing original signer's public key  $y$ ,  $g'$  and  $r_A$ , then she/he can derive the secret key  $x$  of original signer. That is a contradiction to the criteria of discrete logarithm problem. Therefore, the proxy signer only can get  $s_A$  from original signer; and if a verifier gets a valid signature, then the verifier can be convinced that the original signer delegates her/his authority to the proxy signer.

## II. Proxy signature based on ECDSA part:

**Theorem 4.6:** If a malicious proxy signer attempts to impersonate an original signer to create a delegation certificate, then the malicious proxy signer can derive the secret key of original signer.

Proof:

If a malicious proxy signer can create a delegation certificate, she/he may randomly select  $k_A$  ( $1 < k_A < q$ ) such that  $R_A = k_A G$ . Besides, she/he must know  $s_A$  to pass  $R_A = s_A G - e'Q$  in the proxy verification phase and the proxy key  $s_B = s_A k_o^{-1} \bmod q$  is also derive from  $s_A$ .

$s_A = (xe + k_A) \bmod q$ ; hence the malicious proxy signer can derive:

$$x = e^{-1}(s_A - k_A) \bmod q. \quad \square$$

From theorem 4.6, if the malicious proxy signer can create a delegation certificate under just knowing original signer's public key  $Q$ ,  $G'$  and  $R_A$ , then she/he can derive the secret key  $x$  of original signer. That is a contradiction to the criteria of elliptic curve cryptosystem. On the other hand, the proxy signer only can get  $s_A$  from original signer. If a verifier gets a valid signature, then the verifier can be convinced that the original signer delegates her/his authority to the proxy signer.

After the proxy signer Bob receiving a delegate certificate  $(r_A, s_A)$  (or  $(R_A, s_A)$  in ECDSA) correctly from the original signer Alice, he cannot forge another delegate certificate to create a proxy key because it is difficult to find another  $r_A$  (or  $R_A$  in ECDSA) for creating a valid delegation certificate. On the other hand, Alice can neither forge the proxy key because the generator is blinded by a factor  $\sigma$  (or  $k_o$  in ECDSA), which is only known by Bob. Thus, only the authorized proxy signer can create the valid proxy key, which means the proposed scheme confirms the properties of *strong unforgeability* [LK99][LKK01a][LKK01b] and *proxy-protected*.

In the proxy signature based on DSA scheme, the size of  $q$  is 160 bits and the size of  $p$  is between 512 and 1024 bits. For the security reason, a 512-bit prime merely provides marginal security such that at least 786 bits is recommended. Suppose  $p$  is a 768-bit integer and one modular exponentiation takes on 240 modular multiplications [MOV96]. In Table 4.3, we compare the time complexity between the proxy signature based on DSA/ECDSA scheme and the DSA/ECDAS. The major portion of time complexity is modular multiplications and modular inverses, thus we neglect the time complexity of one-way hash function and modular additions. In the proxy signature based on DSA/ECDSA scheme, the time complexity of the proxy signature is similar to the DSA/ECDSA, while the time complexity of the proxy signature verification requires only one modular exponentiation instead of two modular multiplications for the DSA.

Table 4.3 Time complexity of the proxy signature based on DSA/ECDSA and DSA/ECDSA

Schemes	Key Generation	Proxy Generation	Proxy Verification	Signature	Verification
DSA	$240T_{mm}$			$242T_{mm} + T_{inv}$	$483T_{mm} + T_{inv}$
ECDSA	$T_m$			$T_m + 2T_{mm} + 2T_{inv}$	$2T_m + 2T_{mm} + T_{inv}$
Proxy Signature based on DSA	$240T_{mm}$	$721T_{mm}$	$962 T_{mm}$	$242T_{mm} + T_{inv}$	$725T_{mm} + T_{inv}$
Proxy Signature based on ECDSA	$T_m$	$2T_m + 2T_{mm}$	$3T_m + T_{mm} + T_{inv}$	$T_m + 2T_{mm} + T_{inv}$	$3T_m + 3T_{mm} + T_{inv}$

Note:  $T_m$ : The number of multiplication.  
 $T_{mm}$ : The number of modular multiplication.  
 $T_{inv}$ : The number of modular inverse with 160-bit.

The proposed schemes are modified from conventional DSA/ECDSA and the conventional DSA/ECDSA can be reduced to our proposed scheme in polynomial time. Furthermore, no other scheme based on standard signature DSA/ECDSA, so we show the differences among DSA/ECDSA, Mambo's proxy signature scheme and the proposed schemes in Table 4.4.



Table 4.4 Differences among DSA/ECDSA, Mambo and proposed schemes

	Based on Signature	Proxy functionality	Combining with PKI	Standard Signature	Proxy-protected
DSA	ElGamal and Schnorr	No	No	√	No
ECDSA	Elliptic Curve	No	No	√	No
Mambo's Scheme	ElGamal	√	No	No	No
Proposed scheme based on DSA	DSA	√	√	Generalized Standard	√
Proposed scheme based on ECDSA	ECDSA	√	√	Generalized Standard	√

## 4.2 Proxy Signature Based on QR

The proxy signature scheme based on QR scheme is more efficient than other schemes based on discrete logarithms or factoring. Moreover, the proposed scheme involves relatively few multiplications; therefore, the proposed scheme is ideal for low power and low computing device such as mobile phones, IC cards, sensor network nodes, and so on. The delegation by warrant proxy signature scheme based on QR,  $DWPS_{QR}$  comprise four phases; we describe as following:

- (1) initial phase,
- (2) proxy phase,
- (3) proxy-signature phase, and
- (4) verification phase.



### 4.2.1 Delegation by warrant proxy signature scheme based on QR

SA is lead in the  $DWPS_{QR}$  scheme. The SA holds the secret and public system keys, which can grant the delegation capability to an original signer and the signing capability to a proxy signer, respectively. Therefore, the SA prevents the misuse of unqualified proxy signers and improves the warrant mechanism used for negotiations between the original and proxy signers. Additionally, the SA takes responsibility for publishing the public keys of both original and proxy signers.

During the initial phase, the SA, an original signer and a proxy signer generate the secret and public system key pair interactively. Meanwhile, both the original and proxy signers create the parameters required for signature authentication. Subsequently, the

original signer signs a warrant information  $m_w$  in the proxy phase. The symbol  $h(\cdot)$  used in the proxy phase denotes a one-way hash function; and the symbol  $m_w$  indicates a proxy signature restriction such as the proxy valid period. When original signer delivers the system key to the proxy signer, the proxy signer will identify the original signer and verify the system key.

Within the proxy-signing phase, the proxy signer signs the document  $m$  and returns the proxy signature to the applicant. The verifier determines whether the proxy signature is valid during the final phase. Fig 4.2 illustrates the whole phases, and the following sections presents the details of the DWPS<sub>QR</sub> scheme.



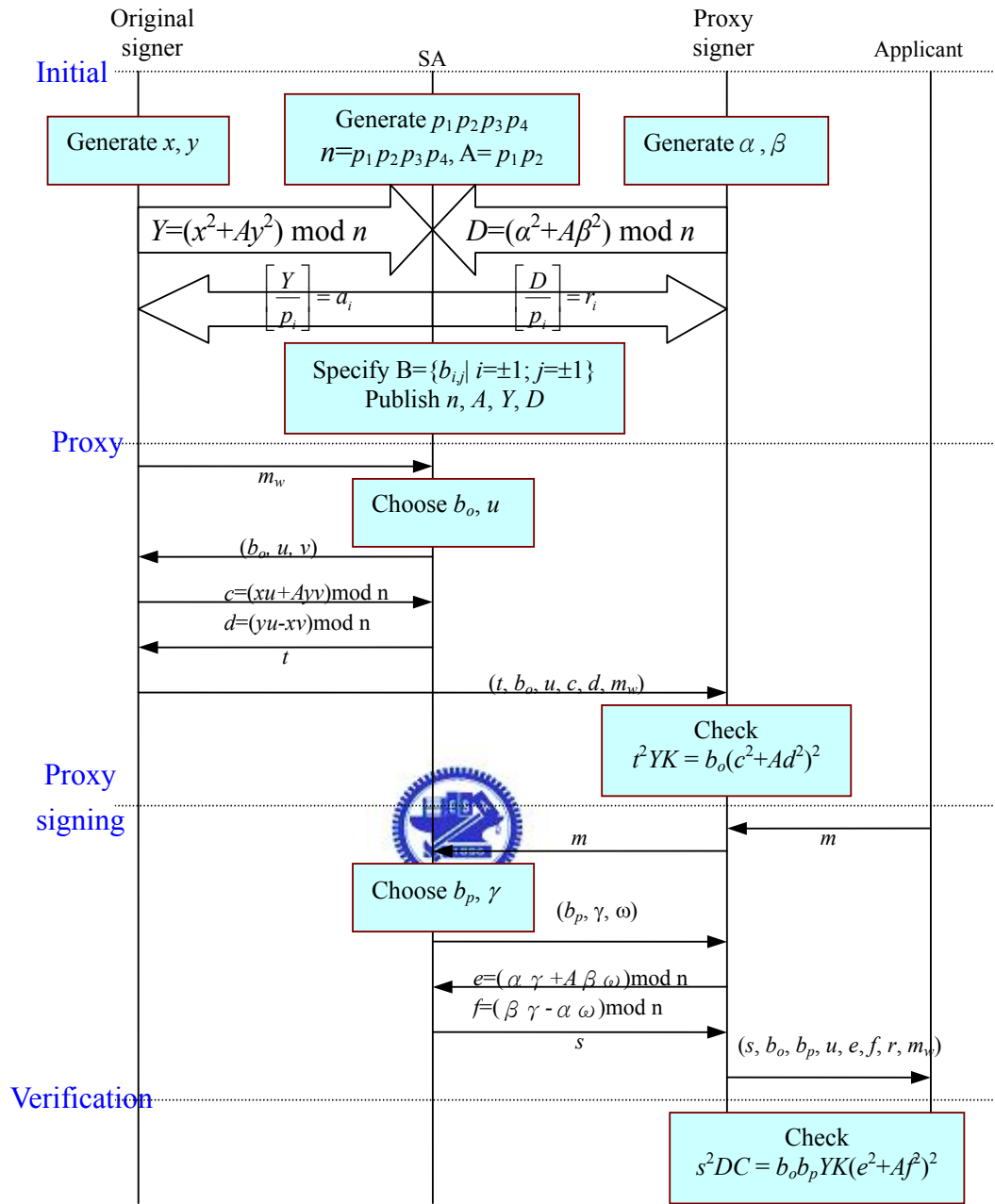


Fig 4.2 Delegation by warrant proxy signature scheme based on QR

### Initial phase

Step 1: The SA selects the large prime numbers  $p_i$  with  $p_i \equiv_4 3$  where  $i = 1, 2, 3, 4$  and lets

$$n = \prod_{i=1}^4 p_i . \quad \text{Thereafter, the SA sets } A = p_1 p_2 \text{ and assigns } (n, A) \text{ and } (p_1, p_2, p_3, p_4)$$

as the system public and private keys respectively.

Step 2: The SA specifies four elements as  $B=\{b_{ij} | i=\pm 1; j=\pm 1\}$  in  $\mathbf{Z}_n^*$

$$\text{so that } \left[ \frac{b_{i,j}}{p_1} \right] = i \text{ and } \left[ \frac{b_{i,j}}{p_2} \right] = j.$$

Step 3: The original signer selects  $x$  and  $y$  in  $\mathbf{Z}_n^*$  as original private keys and sends public key  $Y = (x^2 + Ay^2) \bmod n$  to the SA thereafter. Relatively, the proxy signer selects  $\alpha$  and  $\beta$  in  $\mathbf{Z}_n^*$  as proxy private keys and sends public key  $D = (\alpha^2 + A\beta^2) \bmod n$  to the SA.

Step 4: The SA sends  $\left[ \frac{Y}{p_i} \right] = a_i$  and  $\left[ \frac{D}{p_i} \right] = r_i, i=1, 2, 3, 4$  to the original signer and the proxy signer respectively.

Step 5: The SA publishes the public key of system, as well as those for original signer and proxy signer.



### **Proxy phase**

Step 1: The original signer sends warrant message  $m_w$  to the SA.

Step 2: The SA selects a proper integer  $b_o \in B$  such that

$$\left\{ \begin{array}{l} \left[ \frac{b_o}{p_1} \right] = a_1 \\ \left[ \frac{b_o}{p_2} \right] = a_2 \end{array} \right.$$

Step 3: The SA chooses  $u \in \mathbf{Z}_n^*$  such that  $v = h(u * m_w)$  in  $\mathbf{Z}_n^*$  and

$$\left\{ \begin{array}{l} \left[ \frac{b_o K}{p_3} \right] = a_3 \\ \left[ \frac{b_o K}{p_4} \right] = a_4 \end{array} \right. , \text{ where } K = (u^2 + Av^2) \bmod n$$

Step 4: The SA sends  $(b_o, u, v)$  to the original signer. Thereafter the original signer lets

$$c = (xu + Ayv) \bmod n$$

$$d = (yu - xv) \bmod n$$

Step 5: The original signer sends  $(c, d)$  to the SA. The SA uses system private key  $p_1, p_2, p_3$  and  $p_4$  to solve the square root  $t$  of  $b_o(c^2 + Ad^2)$  in  $O(\log n)$  time complexity [Per86].

Step 6: The SA sends square root  $t$  to the original signer.

Step 7: The original signer sends  $(t, b_o, u, c, d, m_w)$  to the proxy signer. After receiving  $(t, b_o, u, c, d, m_w)$ , the proxy signer examines whether  $t^2 YK \equiv_n b_o(c^2 + Ad^2)^2$  or not. In addition, the proxy signer can compute  $K = (u^2 + Av^2) \bmod n$  with  $v = h(u * m_w)$  herself/himself and retrieve the original signer's public key  $Y$  from the SA.

### Proxy-signing phase



Step 1: After receiving message  $m$  from an applicant, the proxy signer sends  $m$  to the SA.

Step 2: The SA selects a proper integer  $b_p \in B$  such that

$$\begin{cases} \left[ \frac{b_p}{p_1} \right] = r_1 \\ \left[ \frac{b_p}{p_2} \right] = r_2 \end{cases}$$

Step 3: The SA chooses  $\gamma \in \mathbf{Z}_n^*$  such that  $\omega = h(\gamma * m)$  in  $\mathbf{Z}_n^*$  and

$$\begin{cases} \left[ \frac{b_p C}{p_3} \right] = r_3 \\ \left[ \frac{b_p C}{p_4} \right] = r_4 \end{cases} \quad \text{where } C = (\gamma^2 + A\omega^2) \bmod n$$

Step 4: The SA sends  $(b_p, \gamma, \omega)$  to the proxy signer.



Step 5: The proxy signer lets

$$e = (\alpha\gamma + A\beta\omega) \bmod n$$

$$f = (\beta\gamma - \alpha\omega) \bmod n$$

Step 6: The proxy signer sends  $(e, f)$  to the SA. The SA uses the system private key  $p_1, p_2, p_3$  and  $p_4$  to solve the square root  $s$  of  $b_p t^2 DC$  with  $O(\log n)$  time complexity.

Step 7: The SA sends square root  $s$  to proxy signer.

Step 8: The proxy signer sends the proxy signature  $(s, b_o, b_p, u, e, f, \gamma, m_w)$  of message  $m$  back to the applicant.

### Verification phase

The verifier checks whether  $s^2 DC \stackrel{?}{\equiv}_n b_o b_p YK (e^2 + Af^2)^2$  to examine the validity of proxy signature  $(s, b_o, b_p, u, e, f, \gamma, m_w)$ .

The verifier can retrieve  $C = (\gamma^2 + A\omega^2)$  and  $K = (u^2 + Av^2)$  automatically. Furthermore,  $D = (\alpha^2 + A\beta^2) \bmod n$  and  $Y = (x^2 + Ay^2) \bmod n$  are the public keys of the proxy signer and original signer respectively.

### 4.2.2 Correctness analysis of DWPS<sub>QR</sub>

**Lemma 4.1:**  $c^2 + Ad^2 \equiv_n YK$ , where  $c \equiv_n xu + Ayv$ ,  $d \equiv_n yu - xv$ ,  $Y = (x^2 + Ay^2) \bmod n$  and

$$K = (u^2 + Av^2) \bmod n$$

Proof:

Place  $c \equiv_n xu + Ayv$  and  $d \equiv_n yu - xv$  onto  $c^2 + Ad^2$ , then we can compute:

$$\begin{aligned}
& c^2 + Ad^2 \\
& \equiv_n (xu + Ayv)^2 + A(yu - xv)^2 \\
& \equiv_n (xu)^2 + (Ayv)^2 + 2Axyuv + A(yu)^2 + A(xv)^2 - 2Axyuv \\
& \equiv_n (xu)^2 + (Ayv)^2 + A(yu)^2 + A(xv)^2 \\
& \equiv_n (x^2 + Ay^2)(u^2 + Av^2) \\
& \equiv_n YK. \quad \square
\end{aligned}$$

**Lemma 4.2:**  $\left[ \frac{b_o YK}{p_i} \right] = 1$ , where  $i=1, 2, 3, 4$ .

Proof:

From the proxy phase, the SA sets  $\left[ \frac{Y}{p_i} \right] = a_i$ , where  $i=1, 2, 3, 4$ . Also, the SA selects proper  $b_o$  and  $u$ , such that

$$\left\{ \begin{aligned} \left[ \frac{b_o}{p_1} \right] &= a_1 \\ \left[ \frac{b_o}{p_2} \right] &= a_2 \end{aligned} \right\}, \text{ and } \left\{ \begin{aligned} \left[ \frac{b_o K}{p_3} \right] &= a_3 \\ \left[ \frac{b_o K}{p_4} \right] &= a_4 \end{aligned} \right\}$$

By the *Jacobi symbol*,

$$\begin{aligned}
& \left[ \frac{b_o K}{p_1} \right] \\
& = \left[ \frac{b_o (u^2 + Av^2)}{p_1} \right] \\
& = \left[ \frac{b_o}{p_1} \right] \left[ \frac{(u^2 + Av^2)}{p_1} \right]
\end{aligned}$$

$$= \begin{bmatrix} b_o \\ p_1 \end{bmatrix} \begin{bmatrix} u^2 \\ p_1 \end{bmatrix} (\because A = p_1 p_2)$$

$$= \begin{bmatrix} b_o \\ p_1 \end{bmatrix}$$

$$= a_1$$

$$\text{Similarly, } \begin{bmatrix} b_o K \\ p_2 \end{bmatrix} = a_2.$$

$$\text{Hence, } \begin{bmatrix} b_o K \\ p_i \end{bmatrix} = a_i, \text{ where } i=1, 2, 3, 4.$$

$$\begin{bmatrix} b_o YK \\ p_i \end{bmatrix}$$

$$= \begin{bmatrix} Y \\ p_i \end{bmatrix} \begin{bmatrix} b_o K \\ p_i \end{bmatrix}$$

$$= (a_i)^2$$

$$= 1.$$



$$\text{Therefore, } \begin{bmatrix} b_o YK \\ p_i \end{bmatrix} = 1, \text{ where } i=1, 2, 3, 4. \quad \square$$

$$\begin{bmatrix} b_o YK \\ p_i \end{bmatrix} = 1, \text{ where } i=1, 2, 3, 4; \text{ therefore, } t^2 \equiv_n b_o (c^2 + A d^2) \text{ belongs to } QR_n.$$

**Theorem 4.7:** The proxy signer can verify the validity of delegation  $(t, b_o, u, c, d, m_w)$  by checking whether  $t^2 YK \equiv_n b_o (c^2 + A d^2)^2$ , where  $K = (u^2 + A v^2) \pmod n$  and  $v = h(u * m_w)$ .

Proof:

Lemma 4.2 show that  $t^2 \equiv_n b_o (c^2 + A d^2)$  belongs to  $QR_n$ ; and based on proposed

protocol, if  $t$  is valid, then  $t^2 \equiv_n b_o(c^2 + Ad^2)$ .

Form lemma 4.1, we know that  $c^2 + Ad^2 \equiv_n YK$ ; hence:

$$\begin{aligned} t^2 YK &\equiv_n b_o(c^2 + Ad^2)(c^2 + Ad^2) \\ &\equiv_n b_o(c^2 + Ad^2)^2. \end{aligned}$$

□

### 4.2.3 Security requirements of DWPS<sub>QR</sub>

**Lemma 4.3:**  $e^2 + Af^2 \equiv_n DC$ , where  $e \equiv_n (\alpha\gamma + A\beta\omega)$ ,  $f \equiv_n (\beta\gamma - \alpha\omega)$ ,  $C = (\gamma^2 + A\omega^2) \pmod n$

$$\text{and } D = (\alpha^2 + A\beta^2) \pmod n$$

Proof:

Place  $e \equiv_n (\alpha\gamma + A\beta\omega)$  and  $f \equiv_n (\beta\gamma - \alpha\omega)$  onto  $e^2 + Af^2$ , then we can compute:

$$\begin{aligned} e^2 + Af^2 &\equiv_n (\alpha\gamma + A\beta\omega)^2 + A(\beta\gamma - \alpha\omega)^2 \\ &\equiv_n (\alpha\gamma)^2 + (A\beta\omega)^2 + 2A\alpha\beta\gamma\omega + A(\beta\gamma)^2 + A(\alpha\omega)^2 - 2A\alpha\beta\gamma\omega \\ &\equiv_n (\alpha\gamma)^2 + (A\beta\omega)^2 + A(\beta\gamma)^2 + A(\alpha\omega)^2 \\ &\equiv_n (\alpha^2 + A\beta^2)(\gamma^2 + A\omega^2) \\ &\equiv_n DC \end{aligned}$$

□

Similarly to lemma 4.2, we know that  $\left[ \frac{b_p DC}{p_i} \right] = 1$ , where  $i=1, 2, 3, 4$ . Therefore,

$s^2 = b_p t^2 DC$  belongs to  $QR_n$ .

**Theorem 4.8:** Every valid proxy signature  $(s, b_o, b_p, u, e, f, \gamma, m_w)$  satisfies

$$s^2 DC \equiv_n b_o b_p YK(e^2 + Af^2)^2$$

Proof:

$s^2 = b_p t^2 DC$  and from lemma 4.3, we know that  $e^2 + Af^2 \equiv_n DC$ ; hence

$$\begin{aligned} & s^2 DC \\ & \equiv_n b_p t^2 (DC)^2 \\ & \equiv_n b_p b_o (c^2 + Ad^2) (DC)^2 \\ & \equiv_n b_o b_p YK(DC)^2 \\ & \equiv_n b_o b_p YK(e^2 + Af^2)^2. \end{aligned}$$

□

Theorem 4.7 show that the proxy key of the DWPS<sub>QR</sub> scheme is valid from original signer and theorem 4.8 show that the DWPS<sub>QR</sub> scheme satisfies *verifiable* requirement. The following discussion demonstrates that the DWPS<sub>QR</sub> scheme satisfies the *strong unforgeability* requirement. Attackers will encounter difficulty in solving the square root  $s$  and  $t$  without knowing the system private key  $(p_1, p_2, p_3, p_4)$  [FL96][FL98]. Although the attackers could select a modulus pair  $(c', d')$  to pass  $t^2 \equiv_n b_o (c'^2 + Ad'^2)$  verification, it is still difficult for  $(c', d')$  pair to pass  $(c'^2 + Ad'^2) \equiv_n YK$  examination [PS87]. Accordingly, the attackers have difficulty in forging a proxy authentication  $(t, b_p, u, c, d, m_w)$  during the proxy phase.

The SA prevents unqualified original signers from delegating warrant; moreover, prevents unqualified proxy signers from signing a document. Consequently, the SA mechanism enhances the warrant mechanism and avoids original signers from abusing her/his delegation in the same time.

During the proxy-signing phase, proxy signers use their private keys  $\alpha$  and  $\beta$  to

create a proxy signature  $(s, b_o, b_p, u, e, f, \gamma, m_w)$  on document  $m$ . This security mechanism means that attackers cannot forge  $e$  and  $f$  unless they know  $\alpha$  and  $\beta$ .

During the verification phase, any verifier can identify the corresponding proxy signer by using the public key of each proxy signer to check if  $(e^2 + Af^2) \equiv_n DC$ . The  $DWPS_{QR}$  scheme thus satisfies the **strong identifiability** requirement.

Additionally, a proxy signer cannot repudiate that they are the issuers of their signature because no one can create a proxy signature during polynomial time without knowing the private keys of the system and proxy. Consequently, the  $DWPS_{QR}$  scheme fulfills the **strong undeniability** requirement. From above discussions, the  $DWPS_{QR}$  scheme meets the security requirements defined by B. Lee [LK99][LKK01a].

#### 4.2.4 Time complexity and security analysis



The complexity of one-way hash function can be negligible compared to that of the multiplication operation. The proposed proxy signature based on QR scheme does not use exponential and divisional operations throughout the four proposed phases. Consequently, an original signer and a proxy signer complete the proxy phase in just 16 multiplications. During the proxy-signing phase, the proxy signer also uses just 5 multiplications to create a proxy signature. Only 17 multiplications are required to verify the validity of the proxy signature. The above computations are performed under  $Z_n^*$ . A modular exponent requires about 240 modular multiplications [MOV96]; a 2048-bit modular multiplication is of 8 ( $n^3$ ) times complexity than 1024-bit modular multiplication in worse case and  $\left\lfloor \frac{2(n+2)}{3} \right\rfloor + 3 = 5$  in [KT03]. For convenience, we ignore the negligible time complexity of addition operation in Table 4.5. Therefore, the proxy signature scheme based on QR is more efficient than any other scheme based on discrete logarithm.

Table 4.5 Time complexity of Manbo's and proposed scheme

	Proxy Generation	Proxy Verification	Signature Signing	Verification
Manbo's Scheme	$(2*240+1) = 481 T_{mm}$	$(240+1) T_{mm}$	$240 T_{mm}$	$(2*240+1) = 481 T_{mm}$
Proposed Scheme (1024)	$16 T_{mm}$	$8 T_{mm}$	$5 T_{mm}$	$17 T_{mm}$
Proposed Scheme (2048)	$\dagger 16 \times 8 = 128 T_{mm}$	$\dagger 8 \times 8 = 64 T_{mm}$	$\dagger 5 \times 8 = 40 T_{mm}$	$\dagger 17 \times 8 = 136 T_{mm}$

Note:  $T_{mm}$ : The number of modular multiplication.

$\dagger$  For comparing in 1024-bit, we time 8 to keep time complexity consistency.

The security of proposed schemes based on QR assumption. Since  $n=p_1 p_2 p_3 p_4$  and  $A=p_1 p_2$ , how to choose  $(p_1, p_2, p_3, p_4)$  is very important. Comparing to the security level of 1024 bit RSA or discrete logarithms; the proposed schemes have to choose  $(p_1, p_2, p_3, p_4)$  such that  $n$  is around 2048 bits. Because  $A=p_1 p_2$  is published,  $n$  is easy to be divided into  $A$  and  $p_3 p_4$  ( $n = A * p_3 p_4$ ). To break the proxy signature scheme based on QR can reduce to 1024 bits RSA in polynomial time. As a result, the  $n$  need 2048-bit for security issue. Furthermore, the multiplication in 2048 bits is still remarkable faster than exponential in 1024 bits as shown in Table 4.5.

## Chapter 5 Proxy Signature with Proactive Property

The security of proxy signature scheme guarantees last as long as the secret keys remain unrevealed. Many threshold proxy signature schemes [DF89][Zha97b][KPW97][SLH99][HWW03] are proposed enhance security against the key exposure problem prior to this date; but they still lack the property of proactive and proxy. On the other hand, M. Abdalla and L. Reyzin proposed “A new forward-secure digital signature scheme,” in Asiacrypt2000 [AR00] to deal with key exposure problem. Thereafter, Chang, Lin and Yeh proposed "Forward Secure Proxy Signature Scheme," which the proxy signer renews its proxy keys periodically [CLY03]; but in their scheme the proxy key cannot be recovery.

The proactive cryptography was first proposed by Ostrovsky and Yung [OY91] and applied by Herzberg et al. in [HJKY95]. In the proactive security scheme, the secret can be distributed to each party; and each party can refresh her/his share and verify others share. Moreover, if any party lost her/his share, the other parties can help her/him to reconstruct her/his share. We list the properties of proactive security as following:

1. Distributing the secret
2. Verifying the shares
3. Refreshing the shares
4. Recovering the shares

H. M. Sun, N. Y. Lee, and T. Hwang, proposed " Threshold proxy signatures" in 1999 [SLH99] with nonrepudiable property to improve Zhang's threshold proxy signature scheme [Zha97b], but they still lack the property of renewing and recovery. W.G. Tzeng and Z.J. Tzeng proposed “Robust Forward-Secure Signature Schemes with Proactive Security,” in PKC 2001 [TT01] to enhances the security of Abdalla and Reyzin’s [AR00]



forward-secure signature scheme via threshold and proactive mechanisms. V. Nikov and S. Nikova investigates the security of Proactive Secret Sharing Schemes [NN04] which modifies model of Herzberg's et al. [HJKY95] by imposing less restriction to the adversary, but they still lack the property of proxy.

## 5.1 Proactive Secret Sharing Scheme

The proactive secret sharing proxy signature scheme are based on proactive secret sharing signature with proxy functionality; hence we describe proactive secret sharing signature in this session. Proactive secret sharing scheme [HJKY95] is based on verifiable secret sharing (VSS) [Ped91]. A VSS scheme allows players to be verified that no other players are lying about the contents of their shares. In other word, a VSS scheme distribute a secret to  $n$  persons such that each person can verify what he has received correct information about the secret without talking to other persons. We describe the proactive secret sharing proxy signature scheme as follows.

Let  $p$  be a large prime,  $q$  be a prime factor of  $p-1$ , and  $g$  be a generator of order  $q$  in  $\mathbf{Z}_p^*$ . A proactive secret sharing scheme includes  $n$  participants  $\{U_1, U_2, \dots, U_n\} \subset$  participant group ( $PG$ ) with  $(t, n)$  threshold that at least any  $t$  signers can recover the secret. And there are three schemes – a verifiable secret sharing scheme [Ped91], a secret sharing update scheme, and a secret sharing recovery scheme in a proactive secret sharing scheme [HJKY95] which are described as follows:


Each participant  $U_i$  in  $PG$  selects a secret  $s_i \in \mathbf{Z}_q^*$ . And the secret  $s = s_1 + s_2 + \dots + s_n$ . Then,  $U_i$  executes Algorithm 5.1 VSS ( $s_i, n, t, a$ ) to distribute secret  $s_i$  and publish  $g^{a_i,0}, g^{a_i,1}, g^{a_i,2}, \dots, g^{a_i,t-1}$ . Algorithm 5.1 is a method in which each participant  $U_i$  ( $1 \leq i \leq n$ )

distributes a secret  $s_i$  into  $n$  shares. Thereafter,  $U_i$  can compose her/his own share:

$$share_i = \sum_{j=1, j \neq i}^n f_j(i) \bmod q,$$

where function  $f_i(x)$  is defined in Algorithm 5.1.

To prevent participants from distributing wrong shares,  $U_i$  needs to publish  $g^{a_{i,0}}, g^{a_{i,1}}, g^{a_{i,2}}, \dots, g^{a_{i,t-1}}$ ;  $U_i$  can verify her/his own  $share_i$  by checking whether the following equation holds:

$$\begin{aligned} g^{share_i} &= g^{\sum_{j=1}^n f_j(i)} \\ &= g^s \prod_{j=1}^{t-1} (g^{a_j})^{i^j} \bmod p \\ &= g^s \prod_{j=1}^{t-1} (g^{\sum_{k=1}^n a_{k,j}})^{i^j} \bmod p \end{aligned}$$


$$\begin{aligned} f(x) &= s + \sum_{k=1}^n a_{k,1} x + \sum_{k=1}^n a_{k,2} x^2 + \dots + \sum_{k=1}^n a_{k,t-1} x^{t-1} \\ &= s + \sum_{j=1}^{t-1} \sum_{k=1}^n a_{k,j} x^j \end{aligned}$$

Without loss of generality, we assume that given any  $t$  shares  $share_1, \dots, share_t$  which can rebuild secret  $s = f(0)$  by Lagrange interpolating formula [MOV96] as follows:

$$s = f(0) = \sum_{k=1}^t share_k \prod_{j=1, j \neq k}^t \frac{(0-j)}{(k-j)} \pmod{q}.$$

Algorithm VSS( $s_i$ : a secret,  $n$  : number of participant group,  $t$  : number of recovery share,  $a$  : random coefficient code)

Summary: A verifiable secret sharing scheme without dealer. At least  $t$  participants from  $\{U_1, \dots, U_n\}$  can rebuild the secret  $s$ .

### Secret sharing generation

1. Obtain  $(p, q, g)$ .
2. Each participant  $U_i$  let  $a_{i,0} = s_i$  and selects random number  $a_{i,1}, a_{i,2}, \dots, a_{i,t-1}$ .
3.  $U_i$  generates a polynomial of degree  $t-1$ :  $f_i(x) = a_{i,0} + \sum_{k=1}^{t-1} a_{i,k} x^k \pmod{q}$ .
4. Then,  $U_i$  computes and sends  $f_i(j)$  to  $U_j$  (for  $j = 1, \dots, n; i \neq j$ ) in a secure manner; then  $U_i$  publishes  $g^{a_{i,0}}, g^{a_{i,1}}, g^{a_{i,2}}, \dots, g^{a_{i,t-1}}$ .

5. Let  $a_1 = \sum_{k=1}^n a_{k,1}, a_2 = \sum_{k=1}^n a_{k,2}, \dots, a_{t-1} = \sum_{k=1}^n a_{k,t-1}$ ; then

$$f(x) = s + \sum_{k=1}^n a_{k,1} x + \sum_{k=1}^n a_{k,2} x^2 + \dots + \sum_{k=1}^n a_{k,t-1} x^{t-1} = s + \sum_{j=1}^{t-1} \sum_{k=1}^n a_{k,j} x^j.$$

### Secret sharing acceptance

4. Each  $U_i$  receives  $f_j(i)$  (for  $j = 1, \dots, n; j \neq i$ ) from the other participants; then computes  $share_i = \sum_{j=1}^n f_j(i) \pmod{q}$  as her/his share.
5. Each  $U_i$  verifies  $share_i$  by checking the following equation holds:

$$g^{share_i} = g^{\sum_{j=1}^n f_j(i)} = g^s \prod_{j=1}^{t-1} (g^{a_j})^{i^j} \pmod{p} = g^s \prod_{j=1}^{t-1} (g^{\sum_{k=1}^n a_{k,j}})^{i^j} \pmod{p}.$$

6. Return  $share_i$ .

Algorithm 5.1 Verifiable Secret Sharing Scheme

## Secret share update

Each participant  $U_i$  in  $PG$  collaborates to renew his own share  $share_{i(old)}$  into new share  $share_{i(new)}$  by Algorithm 5.2 ShareUpdate( $share_{i(old)}$ ,  $n$ ,  $t$ ). The secret  $s$  is still kept, because algorithm VSS ( $0, n, t, a$ ) satisfies constraints  $f(0) = s$  and  $f_i(0) = 0$  respectively.

Algorithm ShareUpdate( $share_{i(old)}$ ): a secret,  $n$  : number of participant group,  $t$  : number of recovery share)

Summary: Update share without change the secret.

1. Obtain  $(p, q, g)$ .
2. Each participant  $U_i$  selects random number  $b_{i,1}, b_{i,2}, \dots, b_{i,t-1}$ .
3.  $U_i$  generates a polynomial:  $f_i(x) = \sum_{k=1}^{t-1} b_{i,k} x^k \pmod{q}$  which satisfy  $f_i(0) = 0$ .
4.  $U_i$  publishes  $g^{b_{i,1}}, g^{b_{i,2}}, \dots, g^{b_{i,t-1}}$ .
5.  $U_i$  computes  $f_i(j)$  and sends it to  $U_j$ .
6.  $U_i$  computes  $share_{i(new)} = share_{i(old)} + \sum_{j=1}^n f_j(i) \pmod{q}$ .
7.  $U_i$  verifies  $share_{i(new)}$  by checking

$$g^{share_{i(new)}} = g^{share_{i(old)} + \sum_{j=1}^n f_j(i)} = g^s \prod_{j=1}^{t-1} (g^{a_{j(new)}})^{i^j} \pmod{p} = g^s \prod_{j=1}^{t-1} (g^{\sum_{k=1}^n (a_{k,j} + b_{k,j})})^{i^j} \pmod{p}.$$

8. Return  $share_{i(new)}$ .

Algorithm 5.2 Share Update

## Secret share recovery

Suppose that  $U_r$  is a participant whose share corrupted and could not pass secret sharing acceptance of Algorithm 5.1. At least  $t$  participants who pass secret sharing acceptance of Algorithm 5.1 can execute Algorithm 5.3 ShareRecovery( $r, n, t$ ) to help  $U_r$  recover  $share_r$ . From the  $t$  participants' help,  $U_r$  can rebuild  $\tilde{f}(x)$ . Because the function  $f_i(r) = 0$  in Algorithm 5.3, the rebuild function  $\tilde{f}(r) = f(r) = share_r$ . Furthermore, due to  $\tilde{f}(0)$  is randomized without parameter  $s$ ,  $U_r$  can not calculate the secret  $s$ .

Algorithm ShareRecovery( $r$  : the under fixed participant  $U_r$ ,  $n$  : the number of participant group,  $t$  : the number of recovery share)


Summary:  $t$  participants  $\{U_1, \dots, U_t\}$  collaborate to rebuild the secret share of  $U_r$ .

1. Each participant  $U_i \in \{U_1, \dots, U_t\}$  selects random number  $c_{i,0}, c_{i,1}, c_{i,2}, \dots, c_{i,t-1}$ .
2.  $U_i$  generates a polynomial:  $f_i(x) = \sum_{k=0}^{t-1} c_{i,k} x^k \pmod{q}$  which satisfies  $f_i(r) = 0$ .
3.  $U_i$  send  $f_i(j)$  to  $U_j$  where  $j = 1, \dots, t, j \neq i$ .
4. On receiving  $f_j(i)$ ,  $U_i$  computes  $recovery_{ri} = \sum_{j=0}^t f_j(i)$  and forwards it to  $U_r$ .
5.  $U_r$  uses the return values  $\{recovery_{r1}, recovery_{r2}, \dots, recovery_{rt}\}$  and Lagrange interpolation formula to obtain  $\tilde{f}(x) = \sum_{k=1}^t recovery_{rk} \prod_{j=1, j \neq k}^t \frac{(0-j)}{(k-j)} \pmod{q}$ . Then recover her/his share  $f(r) = \tilde{f}(r) \pmod{q}$ .
6. Return  $share_r = f(r)$ .

Algorithm 5.3 Share Recovery

## 5.2 Proactive Secret Sharing Proxy Signature Scheme

We are the first one who combine proxy and proactive properties to propose a proactive secret sharing proxy signature scheme. In our scheme, original signer could distribute the secret to designated signers, called proxy signers. The proxy signers could renew their own proxy shares periodically without changing the secret. Moreover, if any proxy signer lost her/his share, the other  $t$  proxy signers can help her/him to reconstruct her/his share. Therefore, we enhance the security of proxy signature scheme via proactive mechanisms to overcome the key exposure and key recovery problem.

There exist a system authority (*SA*) and a certificate authority (*CA*) in the proactive secret sharing proxy signature scheme.  The *SA* manages the public directory and initiates the system parameter  $(p, q, g)$  used in the following section; and the *CA* certifies proxy signers' key pair. In our scheme, the function  $h(\cdot)$  denotes as a one-way hash function; Alice and  $\{ U_1, \dots, U_n \} \subset PG$  (proxy group) denotes an original signer and proxy signers respectively. Alice's key pairs are  $(x_0, y_0 = g^{x_0} \bmod p)$  and each proxy signer  $U_i$  has  $id_i$  and key pairs  $(x_i, y_i = g^{x_i} \bmod p)$ , where  $i=1, \dots, n$  which are certified by the *CA*. Between an original signer Alice and proxy signers  $\{ U_1, \dots, U_n \}$ , there is a warrant  $m_w$  to describe the relationship of delegation including the identities of *PG*, the original signer, Alice and proxy duration etc.

The proactive secret sharing proxy signature scheme contains five sub-functions: proxy generation, proxy share update, proxy signature generation, proxy signature verification and proxy key share recovery. We describe as follows:

## 5.2.1 Proxy Generation

### Step 1. (Group key generation)

$SA$  chooses a random number  $x_G$  as a group key and selects random numbers  $d_1, \dots, d_{t-1}$  to create  $f_G(x)$  as following:

$$f_G(x) = x_G + d_1x + \dots + d_{t-1}x^{t-1} \pmod{q}.$$

Then,  $SA$  sends the shares  $\gamma_i = f_G(i) \pmod{q}$  to each corresponding proxy signer  $U_i \in PG$  (where  $i=1, \dots, n$ ) in a secure manner and publishes:

$$g^{x_G}, D_1 = g^{d_1}, \dots, D_{t-1} = g^{d_{t-1}}.$$

### Step 2. (Proxy key generation)

The original signer Alice chooses randomly  $k \in Z_q^*$ ; computes  $K = g^k \pmod{p}$ , and creates proxy key as following and publish  $g^\sigma$ :

$$\sigma = k + x_0 h(m_w, K) \pmod{q}.$$

### Step 3. (Proxy sharing)

The original signer Alice executes algorithm 5.1 VSS( $\sigma, n, t, b$ ) to share proxy key  $\sigma$  and the shares are  $b_0 (= \sigma), b_1, \dots, b_{t-1}$ . Let  $B_j = g^{d_j} \pmod{p}, j = 0, \dots, t-1$ . Then Alice distributes  $f_j(i) (i, j = 1, \dots, n)$  and  $(m_w, K)$  to the corresponding proxy signers in a secure manner and publishes  $B_j (j=0, \dots, t-1)$ . The function  $f(x)$  will be:

$$f(x) = \sigma + \sum_{k=1}^n b_{k,1} x + \sum_{k=1}^n b_{k,2} x^2 + \dots + \sum_{k=1}^n b_{k,t-1} x^{t-1} = s + \sum_{j=1}^{t-1} \sum_{k=1}^n b_{k,j} x^j$$

### Step 4. (Group key acceptance)

Once proxy signer  $U_i \in PG$  receiving  $\gamma_i$  and  $f_j(i) (i, j = 1, \dots, n)$ , she/he computes

her/his own share  $share_i = \sum_{j=1, j \neq i}^n f_j(i)$  and executes acceptance of algorithm 5.1 to

check validity of  $share_i$  and  $\gamma_i$ .

Step 5. (Proxy key share generation)

If the shares are valid, each proxy signer  $U_i \in PG$  creates her/his proxy key share:

$$\sigma'_i = share_i + \gamma_i h(m_w, K)(\text{mod } q)$$

### 5.2.2 Proxy share update

Step 1. Each proxy signer  $U_i \in PG$  executes algorithm 5.2 ShareUpdate( $share_{i(old)}, n, t$ ) and obtains  $share_{i(new)}, i = 1, \dots, n$ .

Step 2. Each proxy signer  $U_i \in PG$  sends  $f_i(j) \text{ mod } q, j = 1, \dots, n$  to Alice and re-computes her/his own proxy key share  $\sigma'_{i(new)} = share_{i(new)} + \gamma_i h(m_w, K)(\text{mod } q)$ .

Step 3. Alice update the function  $f(x)$ :



$$f_{(new)}(x) = f_{(old)}(x) + \sum_{j=1}^n f_j(x);$$

The function  $f_{(new)}(x)$  still satisfies  $f_{(new)}(0) = s$ .

### 5.2.3 Proxy signature generation

Without loss of generality, we assume that  $\{U_1, \dots, U_t\} \subset PG$  is a set of proxy signers, who collaborate to sign a message  $m$  on behalf of the original signer.

Step 1. Each proxy signer  $U_i \in \{U_1, \dots, U_t\}$  executes algorithm 5.1 VSS ( $\alpha, n, t, c$ ) for sharing a random number  $\alpha$  ( $\alpha = c_0, C_j = g^{c_j} \text{ mod } p, j = 0, \dots, t-1$ ), obtains  $share_{ki}$  and publishes  $C_j$ , where  $i = 0, \dots, t-1$ .



Step 2. To create a proxy signature of the message  $m$ , each proxy signer  $U_i \in \{U_1, \dots, U_t\}$  computes  $SP_i = share_{ki} + \sigma'_i h(m, C_0) \pmod{q}$ . Then sends  $SP_i$  and  $\sigma'_i$  to other proxy signers  $U_j, j = 1, \dots, t, j \neq i$ .

On receiving all the  $SP_i$  and  $\sigma'_i, U_j (j = 1, \dots, t)$  rebuilds  $\sigma$  using Lagrange interpolating formula. And  $U_j$  checks whether

$$g^\sigma = K(y_0)^{h(m_w, K)} \text{ and}$$

$$g^{SP_j} = \prod_{i=0}^{t-1} C_i^{j^i} \left[ \prod_{i=0}^{t-1} B_i^{j^i} (g^{x_G} \prod_{i=1}^{t-1} D_i^{j^i})^{h(m_w, K)} \right]^{h(m, C_0)} \pmod{p}$$

Step 3. Each proxy signer  $U_i \in \{U_1, \dots, U_t\}$  computes  $T = c_0 + \sigma h(m, C_0)$  by applying Lagrange interpolating formula to  $SP_i$ . The proxy signature on  $m$  is:

$$(m, m_w, T, C_0, K).$$



## 5.2.4 Proxy signature verification

A verifier can verify the validity of the proxy signature  $(m, m_w, T, C_0, K)$  by checking whether following equation holds.

$$g^T = C_0 [K(y_0)^{h(m_w, K)}]^{h(m, C_0)}$$

because

$$\begin{aligned} g^T &= g^{c_0 + \sigma h(m, C_0)} = g^{c_0} g^{\sigma h(m, C_0)} \\ &= C_0 [g^\sigma]^{h(m, C_0)} = C_0 [g^\sigma]^{h(m, C_0)} \\ &= C_0 [g^{k + x_0 h(m_w, K)}]^{h(m, C_0)} \\ &= C_0 [K g^{x_0 h(m_w, K)}]^{h(m, C_0)} \\ &= C_0 [K(y_0)^{h(m_w, K)}]^{h(m, C_0)} \end{aligned}$$

### 5.2.5 Proxy share recovery

Suppose the result of which a proxy signer  $U_r$  verifies the share update is failed. At least  $t$  proxy signers can help  $U_r$  recovery her/his share by executing algorithm 5.3  $\text{ShareRecovery}(U_r, n, t)$ .

### 5.3 Comparing to other schemes

The proactive secret sharing proxy signature scheme also uses CA to identify group key and identities of both the original signer and proxy signers. Furthermore, our scheme periodical update key to prevent possible attack. If some proxy signer lost her/his own share, we also can recovery her/his own share through at least  $t$  shares of legal proxy signers. We compare to the other scheme in following table.



Table 5.1 Comparing of Proactive Secret Sharing Proxy Signature

	Proxy Functionality	Group-oriented	Verifiable	Secret Sharing	Share Renewing	Share Recovery
Manbo's Proxy [MUO96]	√					
HJKY's Proactive [HJKY95]		√	√	√	√	√
Sun's Threshold [SLH99]	√	√	√	√		
Tzeng's Proactive [TT01]		√	√	√	√	√
HWW threshold [HWW 03]	√	√		√		
Chang' Forward Proxy [CLY03]	√				√	
Our proposed scheme	√	√	√	√	√	√

# Chapter 6 Conclusion

## 6.1 Conclusion

In this dissertation, we survey lots of related works and propose a novel proxy signature scheme based on QR, strong proxy signature schemes based on DSA and ECDSA respectively and a proactive secret sharing proxy signature scheme. On the other hand, we also analyze one-way hash function, SHA-160, useful technique using in proxy signature scheme, on message schedule and propose an extended SHA-160 and proposed dynamic extended DES respectively.

Through the investigation, most of proxy signature schemes are based on discrete logarithm problem; and the proxy signature schemes based on ECC and factoring are proposed in 2002 and 2003 respectively. We discuss QR approach and propose a proxy signature scheme based on QR, which is a new approach to implement proxy signature.

Moreover, most of the proposed proxy signature schemes are not feasible in practice because the security of those schemes cannot be really proved. Therefore, based on standard signature, DSA/ECDSA, we propose the proxy signature schemes based on DSA/ECDSA, which is pretty well known by their security properties. In addition, the proposed schemes not only satisfy all the requirements of strong proxy signature, which proposed by Lee, et al. but also can combine PKI to prevent man-in-middle attack.

To solve key exposure problem, many threshold proxy signature schemes are proposed in which the  $k$  out of  $n$  threshold schemes deployed; but they still lack the proactive property. As a result, we propose a proactive secret sharing proxy signature scheme to enhance the security of proxy signature. The proxy shares of proposed scheme are periodically renewed; therefore, even if the proxy shares are compromised in some one

period, it will be hurtless. In addition, if any proxy share is ruined, the other proxy signer can help her/him to recovery her/his share.

Plenty of digital signatures such DSA, ECDSA, proxy signature etc. apply a one-way hash function to make they efficient. SHA-series are most famous one-way hash functions and also are standard one-way hash function in United States and Europe. Although FIPS-2, strengthened version of the SHA-1, is proposed, lots of applications are using SHA-160 prior to this date. We, therefore, analyze message schedule of SHA-160 and discover the decay phenomenon; nevertheless, we introduce two SHA-160 corrections to enhance the security of SHA-160.

Electronic Signature Law is established in many countries, the proxy signature scheme is one of most important digital signature applications. We hope our enhancement and proposed schemes can make proxy signature schemes feasible in practice.



## 6.2 Future works

Via SHA-160 analysis, we know that the more nonlinear terms are involved, the more terms of  $f_t$  will be effective. Basing on our result, we will analyze message schedule  $w_t = (w_{t-1})^{b1} \oplus (w_{t-2})^{b2} \oplus (w_{t-3})^{b3} \oplus (w_{t-16})^{b4}$  of SHA-160 to make the optimal development in the future. Wang et al. developed efficient methods to find collisions in SHA-160 with time complexity  $2^{69}$ ; as a result, the SHA-160 faces seriously potential attacks to be used in many applications. We will continue our analysis on SHA-256, 384, 512 and further develop the more secure one-way hash function.


The proxy signature based on QR is more efficient than proxy signature based on discrete logarithm or factoring but there are too many parameters in the proxy signature

scheme based on QR. To reduce complexity of parameters, and find another way to implement proxy signature based on QR without SA are future works. In addition, to compare to other bases proxy signature scheme is our future work too.


The proxy signature scheme can be used in mobile agents, which are autonomous software entities to migrate across different execution environments. Non-repudiation property is also considered in the electronic commerce circumstance. So a customer (proxy signer) representing an original signer generates and loads delegation key pair to the mobile agent for the heterogeneous environment. The proxy signature applying on mobile agents is an interesting topic for future works.



## References

- [AF99] C. Adms, S. Farrell, "Internet X.509 public key infrastructure certificate management protocols," March 1999.
- [ANSI99] ANSI X9.63, "Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography," Jan 1999.
- [AR00] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," Proceedings of Advances in Cryptology – Asiacrypt 2000, Springer-Verlag, 2000.
- [BC04] Eli Biham, Rafi Chen, "Near-Collisions of SHA-0," Advances in Cryptology - Crypto' 2004, LNCS 3152, Springer-Verlag, 2004.
- [Bla79] G. R. Blakley, "Safeguarding cryptographic keys," Proceedings of the National Computer Conference, 1979, American Federation of Information Processing Societies, v.48, pp. 242-268, 1979.
- [BPH02] L. Bassham, W. Polk, R. Housley,  "Algorithms and Identifiers for the Internet X.509 public key infrastructure certificate and Certificate Revocation List (CRL) profile," RFC3279, April 2002.
- [CFSMW03] S. Chokhan, W. Ford, R. Sabett, C. Merill, S. Wu, " Internet X.509 public key infrastructure certificate policy and certificate practices framework," RFC3647, November 2003.
- [Cha83] D. Chaum, "Blind signatures for untraceable payments," Advances in Cryptology - Crypto '82, Springer-Verlag, pp.199-203, 1983.
- [Chang05] Ming-Hsin Chang, "On proxy signatures with forward-secure and one-time properties and their application in PKI," National Chiao-Tung University, Dissertation, March 2005.
- [CJ98] F. Chabaud, and A. Joux, "Differential collisions in SHA-0," Crypto'98, H. Krawczyk ed., LNCS 1462, pp 56-71, 1998.
- [CLC02] Tzer-Shyong Chen, Tzuoh-Pyng Liu, Yu-Fang Chung, "A proxy-protected proxy signature scheme based on the elliptic curve cryptosystem,"

Proceedings of IEEE TENCON'02, pp. 184-187, 2002.

- [CLY03] Ming-Hsin Chang, Tzu-Shin Lin, Yi-Shung Yeh, "Forward Secure Proxy Signature Scheme," Proceedings of National Computer Symposium (NCS2003), pp.1381-1387, 2003.
- [DF89] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," In G. Brassard, editor, Advances in Cryptology – Crypto'89, LNCS No. 435, Springer-Verlag, pp.307-315, 1989.
- [DH76] W. Diffie, and M. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, v.22, pp. 644-654, 1976.
- [EFF98] Electronic Frontier Foundation, Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design, Sebastopol, CA, O'Reilly, 1998.
- [ElG85] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, July 1985.
- [FL96] C. I. Fan, and C. L. Lei,  "Efficient blind signature scheme based on quadratic residues," Electronic Letters, Vol.32, No. 9, pp 811-813, 25th April 1996.
- [FL98] C. I. Fan, and C. L. Lei, "User efficient blind signatures," Electronic Letters, Vol.34, No. 6, pp 544-545, 19th March 1998.
- [Han04] Darrel Hankerson, Alfred Menezes and Scott Vanstone, Guide to Elliptic Curve Cryptography, Springer, 2004.
- [HJKY95] Amir Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or how to cope with perpetual leakage," in Advances in Cryptology: (D. Coppersmith, ed.) CRYPTO '95, vol. 963 of Lecture Notes in Computer Science, Springer, pp. 339–352, 1995.
- [HTT04] Min-Shiang Hwang, Shiang-Feng Tzeng, and Chwei-Shyong Tsai, "Generalization of proxy signature based on elliptic curves," Computer Standards & Interfaces, vol.26, pp.73–84, March 2004.
- [HWW03] Chien-Lung Hsu, Tzong-Sun Wu, Tzong-Chen Wu, "Improvement of threshold proxy signature scheme," Applied Mathematics and Computation,

136, pp. 315–321 2003.

- [IEEE05] IEEE P1363, "Standard specifications for public-key cryptography," Draft version D21, July 17, 2005.
- [Ker83] Auguste Kerckhoffs, La cryptographie militaire, Journal des sciences militaires, vol. IX, pp. 5–83, Jan. 1883, pp. 161–191, Feb. 1883.
- [Kob87] Neal Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation, 48(177), pp. 203-209, 1987.
- [KPW97] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," Proceedings of ICICS'97, Springer-Verlag, LNCS 1334, pp. 223-232, 1997.
- [KT03] Marcelo E. Kaihara and Naofumi Takagi, "VLSI Algorithm for Modular Multiplication/Division," 16<sup>th</sup> IEEE Symposium on Computer Arithmetic (ARITH-16), pp. 220-227, 2003.
- [LA05] S. Lal and A.K. Awasthi, "Proxy blind signature scheme," Revised Version, Transaction on Cryptology, ePrint Archive, pp. 5-11, 2005, available at <http://eprint.iacr.org/2003/072.pdf>
- [Lam79] L. Lamport, "Constructing digital signatures from a one-way function," Technical Report CSL-98, SRI International, 1979.
- [LC03] Wei-Bin Lee and Tzung-Her Chen, "Constructing a proxy signature scheme based on existing security mechanisms," Information & Security International Journal, vol. 12, no. 2, pp.250-258, 2003.
- [LHW98] Narn-Yih Lee, Tzonelih Hwang, Chih-Hung Wang, "On zhang's nonrepudiable proxy signature schemes," ACISP 1998, pp. 415-422, 1998.
- [LK99] B. Lee, and K. Kim, "Strong proxy signatures," IEICE Trans. Fundamentals, vol. E82-A, no.1, pp.1-11, Jan 1999.
- [LKK01a] B. Lee, H. Kim and K. Kim, "Strong proxy signature and its applications," Proceedings of SCIS 2001, 11B-1, pp. 603-608, 2001.
- [LKK01b] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature", Proceedings of ACISP2001, LNCS vol. 2119,



Springer-Verlag, pp. 474-486, 2001.

- [LTH03] Li-Hua Li, Shiang-Feng Tzeng and Min-Shiang Hwang, "Generalization of proxy signature-based on discrete logarithms," *Computer & Security*, vol. 22, no. 3, pp. 245-255, 2003.
- [Men93] Alfred Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.
- [MOV96] Alfred J. Menezes, Paul C. Van, Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
- [MUO96] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign messages, " *IEICE Trans. Fundamentals*, vol. E79-A, no.9, pp.1338-1354, 1996.
- [Nessie04] Nessie Project , "Final report of European project number IST-1999-12324, named NESSIE," Information Society Technologies (IST), Springer-Verlag , April, 2004. <https://www.cosic.esat.kuleuven.ac.be/nessie/>
- [NIST00] NIST. "Digital Signature Standard (DSS)," Federal Information Processing Standards Publication 186, November 1994. Revision (To include ECDSA) 186-2, January, 2000.
- [NIST01] NIST. "Advanced Encryption Standard," Federal Information Processing Standards Publication 197, November 2001.
- [NIST02] NIST. "Secure hash standard," Federal Information Processing Standards Publication FIPS PUB 180-2, Aug. 2002.
- [NIST95] NIST. "Secure hash standard," Federal Information Processing Standards Publication 180-1, April 1995.
- [NN04] Venzislav Nikov and Svetla Nikova, "On Proactive Secret Sharing Schemes," SAC 2004, LNCS 3357, pp. 308-325, Springer-Verlag, 2005.
- [OpenSSL] OpenSSL Project's URL: <http://www.openssl.org>.
- [OW94] P. van Oorschot and M. Wiener, "Parallel collision search with application to hash functions and discrete logarithms," *Proceedings of 2nd ACM Conference*

on Computer and Communication Security, 1994.

- [OY91] R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks," Proceedings of the 10<sup>th</sup> ACM symposium on Principles of Distributed Computing (PODC), pp.51-61, 1991.
- [Ped91] T. P. Pedersen, "Non-interactive and information theoretic secure verifiable secret sharing," Crypto' 91 Proceedings, LNCS Vol. 576, Springer-Verlag, pp. 129-140, 1991.
- [Per86] Rene C. Peralta "A simple and fast probabilistic algorithm for computing square roots modulo a prime number," IEEE Trans on Information Theory, Vol. IT-32, No. 6, pp. 846-847, November 1986.
- [PS87] J. M. Pollard, C. P. Schnorr, "An efficient solution of the congruence  $X_2 + ky^2 = m \pmod{n}$ ," IEEE Trans. Information Theory, 33 (5), pp. 702-709, 1987.
- [Ros05] Kenneth H. Rosen, Elementary Number Theory and its Applications, 5<sup>th</sup>, Addison-Wesley publishing company, 2005.
- [RRSY98] L. Ronald Rivest, M.J.B. Robshaw, R. Sidney, Y.L. Yin, "The RC6 Block Cipher ", Version 1.1, August 20 1998.
- [RSA00] RSA Laboratories, "PKCS #10: Certification request syntax specification," RFC 2986, Version 1.7, November 2000.
- [RSA78] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM 21 (2), pp. 120-126, Feb. 1978.
- [Sch90] C.P. Schnorr, "Efficient Signature Generation for Smart Cards," Advances in Cryptology-Crypto `89, Springer-Verlag, pp. 239-252, 1990.
- [Sch96] Bruce Schneier, Applied cryptography 2<sup>nd</sup>, John Wiley & Sons, 1996.
- [SH04] H. M. Sun and B. T. Hsieh, "On the Security of the some proxy blind signature scheme," Australasian Information Security Workshop (AISW2004), vol. 32, pp 75-78, 2004.
- [Sha79] A. Shamir, "How to share a secret," Communications of the ACM, v.22, n.11,

pp. 612-613, 1979.

- [Shao02] Zuhua Shao, "Proxy Signature Schemes Based on Factoring," Information Processing Letters 85, pp.137–143, 2002.
- [Sim79] Gustavus J. Simmons, "Symmetric and Asymmetric Encryption," Computer surveys Vol. 11(4) pp. 305-330, December 1979.
- [SLH99] H. M. Sun, N. Y. Lee, and T. Hwang, "Threshold proxy signatures" IEE proceedings – Computers and Digital Techniques, vol. 146, no. 5, pp. 259-263, 1999.
- [SPC95] M. Stadler, J.M. Piveteau and J. Camenisch, "Fair blind Signature," Advances in Cryptology-Eurocrypt'95, Lecture Notes in Computer Science, 921, Springer-Verlag, pp.209-219, 1995.
- [Sta03] William Stallings, Cryptography and network security principles and practices 3<sup>rd</sup>, Prentice Hall, 2003.
- [Sti02] Douglas R. Stinson, Cryptography Theory and Practice, 2<sup>nd</sup>, CRC Press, 2002.
- [Sun99] H. M. Sun, "An efficient nonrepudiable threshold proxy signature scheme with known signers," Computer Communications, vol. 22, no. 8, New York, IPC Science and Technology Press, pp. 717-722, 1999.
- [TLT02] Z. Tan, Z. Liu and C. Tang, "Digital proxy blind signature schemes based on DLP and ECDLP," MM Research Preprints, No. 21, MMRC, AMSS, Academia, Sinica, Beijing, pp. 212-217, December 2002.
- [TT01] Wen Guey Tzeng and Zhi Jia Tzeng, "Robust Forward-Secure Signature Schemes with Proactive Security," PKC2001, LNCS 1992, pp. 264-276, Springer-Verlag, 2001.
- [WFLY05] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", Cryptology ePrint Archive, 2005.
- [WV99] C. K. Wu and V. Varadharajan., "Modified Chinese Remainder Theorem and its application to proxy signatures," In ICPP Workshop, pp.146-151, 1999.
- [WY05] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions", EUROCRYPT 2005, LNCS 3494, pp. 19–35, 2005.

- [YH02] Yi-Shiung Yeh and Ching-Hung Hsu "An Extended DES," Journal of Information Science and Engineering, Vol.18, No. 3, pp349-365, May 2002.
- [XC05] Qingshui Xue, Zhenfu Cao, "Factoring based proxy signature schemes," Journal of Computational and applied mathematics, 2005.
- [Zha97a] K. Zhang, "Nonrepudiable proxy signature schemes," Manuscript, Available at <http://citeseer.nj.nec.com/360090.html>, 1997.
- [Zha97b] K. Zhang, "Threshold Proxy Signature Schemes," 1997 Information Security workshop, Japan, pp.191-199, Sep. 1997.



## Appendix A

Appendix A list the result of  $t_1, t_2, t_3$  in the equation

$$w_t = \text{ROTL}^1(w_{t-t_1} \oplus w_{t-t_2} \oplus w_{t-t_3} \oplus w_{t-t_4}), \text{ where } 16 \leq t \leq 79 \}.$$

$t_1$	$t_2$	$t_3$	Total terms	Maximum number of involved terms in $w_t$	Average terms involved of all $w_t$
1	2	3	7222	177	112.5938
1	2	4	7143	203	111.3594
1	2	5	7377	172	115.0156
1	2	6	7173	169	111.8281
1	2	7	8383	194	130.7344
1	2	8	7065	171	110.1406
1	2	9	7427	169	115.7969
1	2	10	7279	175	113.4844
1	2	11	8670	212	135.2188
1	2	12	7189	182	112.0781
1	2	13	8663	212	135.1094
1	2	14	7155	190	111.5469
1	2	15	6705	169	104.5156
1	3	4	6547	151	102.0469
1	3	5	5670	134	88.34375
1	3	6	6706	161	104.5313
1	3	7	6025	149	93.89063
1	3	8	6050	136	94.28125
1	3	9	6136	133	95.625
1	3	10	6873	165	107.1406
1	3	11	6312	157	98.375
1	3	12	5937	153	92.51563
1	3	13	7436	172	115.9375
1	3	14	5701	137	88.82813
1	3	15	7337	172	114.3906
1	4	5	6070	140	94.59375
1	4	6	6372	157	99.3125

1	4	7	4686	129	72.96875
1	4	8	5903	146	91.98438
1	4	9	5940	132	92.5625
1	4	10	4364	116	67.9375
1	4	11	6636	165	103.4375
1	4	12	6234	149	97.15625
1	4	13	4176	111	65
1	4	14	6663	156	103.8594
1	4	15	7066	180	110.1563
1	5	6	5716	137	89.0625
1	5	7	5454	126	84.96875
1	5	8	5051	116	78.67188
1	5	9	4654	102	72.46875
1	5	10	6057	137	94.39063
1	5	11	5608	126	87.375
1	5	12	5352	125	83.375
1	5	13	5161	120	80.39063
1	5	14	5987	139	93.29688
1	5	15	5993	135	93.39063
1	6	7	6247	172	97.35938
1	6	8	5691	144	88.67188
1	6	9	5176	122	80.625
1	6	10	6004	161	93.5625
1	6	11	3234	83	50.28125
1	6	12	5904	162	92
1	6	13	5944	170	92.625
1	6	14	5640	140	87.875
1	6	15	6049	169	94.26563
1	7	8	6325	170	98.57813
1	7	9	5331	121	83.04688
1	7	10	4835	143	75.29688
1	7	11	6093	165	94.95313
1	7	12	6229	168	97.07813
1	7	13	4722	132	73.53125
1	7	14	6041	166	94.14063

1	7	15	6109	155	95.20313
1	8	9	5574	132	86.84375
1	8	10	5747	141	89.54688
1	8	11	5863	154	91.35938
1	8	12	5219	144	81.29688
1	8	13	6259	178	97.54688
1	8	14	5646	146	87.96875
1	8	15	5286	146	82.34375
1	9	10	5276	121	82.1875
1	9	11	4819	106	75.04688
1	9	12	5022	119	78.21875
1	9	13	4615	103	71.85938
1	9	14	4925	111	76.70313
1	9	15	4776	112	74.375
1	10	11	5813	167	90.57813
1	10	12	5156	134	80.3125
1	10	13	4716	130	73.4375
1	10	14	5149	131	80.20313
1	10	15	5907	162	92.04688
1	11	12	5491	150	85.54688
1	11	13	5263	152	81.98438
1	11	14	5241	139	81.64063
1	11	15	5306	145	82.65625
1	12	13	5713	157	89.01563
1	12	14	4503	124	70.10938
1	12	15	5474	155	85.28125
1	13	14	5123	148	79.79688
1	13	15	4984	122	77.625
1	14	15	4567	123	71.10938
2	3	4	4002	118	62.28125
2	3	5	4243	138	66.04688
2	3	6	4233	134	65.89063
2	3	7	4308	134	67.0625
2	3	8	4208	113	65.5
2	3	9	4267	130	66.42188

2	3	10	4094	112	63.71875
2	3	11	4519	135	70.35938
2	3	12	3936	129	61.25
2	3	13	4633	151	72.14063
2	3	14	3764	139	58.5625
2	3	15	3694	130	57.46875
2	4	5	3714	99	57.78125
2	4	6	2972	83	46.1875
2	4	7	3981	107	61.95313
2	4	8	2394	62	37.15625
2	4	9	3875	109	60.29688
2	4	10	3086	77	47.96875
2	4	11	4058	110	63.15625
2	4	12	2846	79	44.21875
2	4	13	4272	119	66.5
2	4	14	2866	73	44.53125
2	4	15	4345	121	67.64063
2	5	6	3980	111	61.9375
2	5	7	4009	126	62.39063
2	5	8	3175	89	49.35938
2	5	9	3648	111	56.75
2	5	10	3328	101	51.75
2	5	11	3568	108	55.5
2	5	12	3734	105	58.09375
2	5	13	3462	113	53.84375
2	5	14	3729	107	58.01563
2	5	15	3885	117	60.45313
2	6	7	3458	109	53.78125
2	6	8	2772	78	43.0625
2	6	9	3386	111	52.65625
2	6	10	2770	75	43.03125
2	6	11	3450	108	53.65625
2	6	12	2900	81	45.0625
2	6	13	3560	111	55.375
2	6	14	2946	90	45.78125



2	6	15	3780	127	58.8125
2	7	8	3538	114	55.03125
2	7	9	3212	101	49.9375
2	7	10	3289	101	51.14063
2	7	11	3443	116	53.54688
2	7	12	2891	93	44.92188
2	7	13	3524	112	54.8125
2	7	14	3369	102	52.39063
2	7	15	3550	113	55.21875
2	8	9	3000	90	46.625
2	8	10	2874	74	44.65625
2	8	11	3109	85	48.32813
2	8	12	2730	71	42.40625
2	8	13	3355	99	52.17188
2	8	14	2654	71	41.21875
2	8	15	3258	92	50.65625
2	9	10	2875	88	44.67188
2	9	11	2820	84	43.8125
2	9	12	2792	81	43.375
2	9	13	2772	85	43.0625
2	9	14	2744	82	42.625
2	9	15	2717	80	42.20313
2	10	11	3204	98	49.8125
2	10	12	2808	81	43.625
2	10	13	2896	90	45
2	10	14	2546	64	39.53125
2	10	15	3249	95	50.51563
2	11	12	2997	82	46.57813
2	11	13	3152	92	49
2	11	14	3099	90	48.17188
2	11	15	3159	98	49.10938
2	12	13	3101	90	48.20313
2	12	14	2608	74	40.5
2	12	15	3060	92	47.5625
2	13	14	2855	86	44.35938

2	13	15	2875	88	44.67188
2	14	15	2574	82	39.96875
3	4	5	2566	70	39.84375
3	4	6	2540	74	39.4375
3	4	7	2578	72	40.03125
3	4	8	2324	56	36.0625
3	4	9	2724	77	42.3125
3	4	10	2614	78	40.59375
3	4	11	2938	80	45.65625
3	4	12	2402	68	37.28125
3	4	13	2919	79	45.35938
3	4	14	2413	65	37.45313
3	4	15	2346	66	36.40625
3	5	6	2606	80	40.46875
3	5	7	2275	69	35.29688
3	5	8	2503	65	38.85938
3	5	9	2455	68	38.10938
3	5	10	2726	78	42.34375
3	5	11	2553	71	39.64063
3	5	12	2349	72	36.45313
3	5	13	2599	79	40.35938
3	5	14	2364	76	36.6875
3	5	15	2846	85	44.21875
3	6	7	2571	86	39.92188
3	6	8	2344	74	36.375
3	6	9	2172	69	33.6875
3	6	10	2255	72	34.98438
3	6	11	2398	74	37.21875
3	6	12	2378	78	36.90625
3	6	13	2336	80	36.25
3	6	14	2579	86	40.04688
3	6	15	2449	76	38.01563
3	7	8	2151	68	33.35938
3	7	9	2291	73	35.54688
3	7	10	2358	68	36.59375

3	7	11	2042	64	31.65625
3	7	12	2334	83	36.21875
3	7	13	2321	74	36.01563
3	7	14	2413	86	37.45313
3	7	15	2196	77	34.0625
3	8	9	2346	68	36.40625
3	8	10	2299	65	35.67188
3	8	11	2345	71	36.39063
3	8	12	2081	60	32.26563
3	8	13	1934	57	29.96875
3	8	14	2271	75	35.23438
3	8	15	2367	73	36.73438
3	9	10	2407	70	37.35938
3	9	11	2256	68	35
3	9	12	2113	64	32.76563
3	9	13	2195	68	34.04688
3	9	14	2230	71	34.59375
3	9	15	1866	58	28.90625
3	10	11	2312	65	35.875
3	10	12	2127	63	32.98438
3	10	13	2005	58	31.07813
3	10	14	2071	58	32.10938
3	10	15	2139	60	33.17188
3	11	12	2110	59	32.71875
3	11	13	2084	59	32.3125
3	11	14	2207	66	34.23438
3	11	15	2031	58	31.48438
3	12	13	2099	63	32.54688
3	12	14	1890	60	29.28125
3	12	15	2089	61	32.39063
3	13	14	1979	68	30.67188
3	13	15	1974	63	30.59375
3	14	15	1759	54	27.23438
4	5	6	1845	56	28.57813
4	5	7	1919	58	29.73438

4	5	8	1706	45	26.40625
4	5	9	1990	53	30.84375
4	5	10	1956	63	30.3125
4	5	11	2027	58	31.42188
4	5	12	1816	53	28.125
4	5	13	2083	65	32.29688
4	5	14	1795	63	27.79688
4	5	15	1781	61	27.57813
4	6	7	1811	61	28.04688
4	6	8	1240	34	19.125
4	6	9	1832	51	28.375
4	6	10	1476	43	22.8125
4	6	11	1829	55	28.32813
4	6	12	1348	43	20.8125
4	6	13	1969	65	30.51563
4	6	14	1464	45	22.625
4	6	15	1976	66	30.625
4	7	8	1661	44	25.70313
4	7	9	1758	49	27.21875
4	7	10	1090	30	16.78125
4	7	11	1831	53	28.35938
4	7	12	1757	50	27.20313
4	7	13	1246	38	19.21875
4	7	14	1799	54	27.85938
4	7	15	1828	53	28.3125
4	8	9	1617	41	25.01563
4	8	10	1374	36	21.21875
4	8	11	1603	45	24.79688
4	8	12	896	23	13.75
4	8	13	1601	48	24.76563
4	8	14	1390	38	21.46875
4	8	15	1573	41	24.32813
4	9	10	1552	40	24
4	9	11	1806	52	27.96875
4	9	12	1674	46	25.90625

4	9	13	1734	50	26.84375
4	9	14	1449	42	22.39063
4	9	15	1678	45	25.96875
4	10	11	1488	44	23
4	10	12	1298	34	20.03125
4	10	13	1075	29	16.54688
4	10	14	1350	40	20.84375
4	10	15	1403	38	21.67188
4	11	12	1504	37	23.25
4	11	13	1704	50	26.375
4	11	14	1629	46	25.20313
4	11	15	1665	45	25.76563
4	12	13	1504	41	23.25
4	12	14	1360	40	21
4	12	15	1544	46	23.875
4	13	14	1485	47	22.95313
4	13	15	1484	45	22.9375
4	14	15	1357	43	20.95313
5	6	7	1389	45	21.45313
5	6	8	1355	35	20.92188
5	6	9	1503	36	23.23438
5	6	10	1447	37	22.35938
5	6	11	1464	40	22.625
5	6	12	1498	45	23.15625
5	6	13	1591	48	24.60938
5	6	14	1305	39	20.14063
5	6	15	1355	38	20.92188
5	7	8	1336	38	20.625
5	7	9	1288	36	19.875
5	7	10	1434	37	22.15625
5	7	11	1350	37	20.84375
5	7	12	1307	37	20.17188
5	7	13	1490	45	23.03125
5	7	14	1380	42	21.3125
5	7	15	1473	43	22.76563

5	8	9	1304	34	20.125
5	8	10	1256	34	19.375
5	8	11	1245	33	19.20313
5	8	12	1325	35	20.45313
5	8	13	1398	39	21.59375
5	8	14	1299	33	20.04688
5	8	15	1438	40	22.21875
5	9	10	1403	37	21.67188
5	9	11	1325	33	20.45313
5	9	12	1440	42	22.25
5	9	13	1206	32	18.59375
5	9	14	1406	38	21.71875
5	9	15	1404	39	21.6875
5	10	11	1481	46	22.89063
5	10	12	1328	37	20.5
5	10	13	1296	40	20
5	10	14	1327	36	20.48438
5	10	15	1164	32	17.9375
5	11	12	1380	44	21.3125
5	11	13	1281	36	19.76563
5	11	14	1288	36	19.875
5	11	15	1327	39	20.48438
5	12	13	1307	40	20.17188
5	12	14	1209	40	18.64063
5	12	15	1370	44	21.15625
5	13	14	1257	43	19.39063
5	13	15	1223	36	18.85938
5	14	15	1118	36	17.21875
6	7	8	1062	28	16.34375
6	7	9	1138	29	17.53125
6	7	10	1183	31	18.23438
6	7	11	1145	35	17.64063
6	7	12	1183	38	18.23438
6	7	13	1268	38	19.5625
6	7	14	1063	32	16.35938

6	7	15	1145	37	17.64063
6	8	9	1112	29	17.125
6	8	10	860	21	13.1875
6	8	11	1084	30	16.6875
6	8	12	832	20	12.75
6	8	13	1211	34	18.67188
6	8	14	870	24	13.34375
6	8	15	1148	29	17.6875
6	9	10	1103	29	16.98438
6	9	11	1117	31	17.20313
6	9	12	1060	30	16.3125
6	9	13	1192	34	18.375
6	9	14	1179	32	18.17188
6	9	15	1110	30	17.09375
6	10	11	1037	31	15.95313
6	10	12	938	25	14.40625
6	10	13	1103	35	16.98438
6	10	14	900	24	13.8125
6	10	15	1148	30	17.6875
6	11	12	1014	29	15.59375
6	11	13	1023	29	15.73438
6	11	14	1007	28	15.48438
6	11	15	998	29	15.34375
6	12	13	1087	34	16.73438
6	12	14	918	25	14.09375
6	12	15	1080	36	16.625
6	13	14	1061	39	16.32813
6	13	15	1059	38	16.29688
6	14	15	958	33	14.71875
7	8	9	929	23	14.26563
7	8	10	913	23	14.01563
7	8	11	992	28	15.25
7	8	12	905	26	13.89063
7	8	13	1069	31	16.45313
7	8	14	892	28	13.6875

7	8	15	987	29	15.17188
7	9	10	942	25	14.46875
7	9	11	931	25	14.29688
7	9	12	961	27	14.76563
7	9	13	951	31	14.60938
7	9	14	995	29	15.29688
7	9	15	998	25	15.34375
7	10	11	948	25	14.5625
7	10	12	967	29	14.85938
7	10	13	598	16	9.09375
7	10	14	917	25	14.07813
7	10	15	952	25	14.625
7	11	12	1015	34	15.60938
7	11	13	929	30	14.26563
7	11	14	1005	33	15.45313
7	11	15	875	26	13.42188
7	12	13	958	32	14.71875
7	12	14	902	30	13.84375
7	12	15	958	32	14.71875
7	13	14	871	31	13.35938
7	13	15	870	27	13.34375
7	14	15	827	31	12.67188
8	9	10	777	19	11.89063
8	9	11	850	21	13.03125
8	9	12	788	23	12.0625
8	9	13	877	25	13.45313
8	9	14	813	24	12.45313
8	9	15	859	23	13.17188
8	10	11	820	24	12.5625
8	10	12	612	15	9.3125
8	10	13	828	30	12.6875
8	10	14	670	17	10.21875
8	10	15	866	26	13.28125
8	11	12	757	21	11.57813
8	11	13	885	27	13.57813



8	11	14	768	21	11.75
8	11	15	878	28	13.46875
8	12	13	736	20	11.25
8	12	14	624	19	9.5
8	12	15	762	22	11.65625
8	13	14	761	27	11.64063
8	13	15	798	27	12.21875
8	14	15	717	24	10.95313
9	10	11	731	19	11.17188
9	10	12	655	18	9.984375
9	10	13	738	23	11.28125
9	10	14	684	18	10.4375
9	10	15	757	24	11.57813
9	11	12	699	18	10.67188
9	11	13	708	18	10.8125
9	11	14	775	25	11.85938
9	11	15	708	20	10.8125
9	12	13	738	23	11.28125
9	12	14	714	23	10.90625
9	12	15	687	21	10.48438
9	13	14	680	23	10.375
9	13	15	648	25	9.875
9	14	15	639	19	9.734375
10	11	12	558	13	8.46875
10	11	13	625	17	9.515625
10	11	14	591	15	8.984375
10	11	15	656	19	10
10	12	13	623	16	9.484375
10	12	14	468	14	7.0625
10	12	15	613	18	9.328125
10	13	14	592	15	9
10	13	15	586	17	8.90625
10	14	15	564	14	8.5625
11	12	13	517	13	7.828125
11	12	14	505	15	7.640625

11	12	15	552	19	8.375
11	13	14	575	16	8.734375
11	13	15	515	13	7.796875
11	14	15	516	14	7.8125
12	13	14	442	12	6.65625
12	13	15	482	14	7.28125
12	14	15	448	14	6.75
13	14	15	366	13	5.46875

