A Thesis for the Degree of Master of Science

A Study on Digital Signature Secure Against Key Exposure

Chul-Joon Choi

School of Engineering

Information and Communications University

2004

A Study on Digital Signature Secure Against Key Exposure

A Study on Digital Signature Secure Against Key Exposure

Advisor : Professor Kwangjo Kim

by

Chul-Joon Choi
School of Engineering
Information and Communications University

A thesis submitted to the faculty of Information and Communications University in partial fulfillment of the requirements for the degree of Master of Science in the School of Engineering

Daejeon, Korea
Dec. 26. 2003
Approved by

(signed)

Professor Kwangjo Kim
Major Advisor

A Study on Digital Signature Secure Against Key Exposure

Chul-Joon Choi

We certify that this work has passed the scholastic standards required by Information and Communications University as a thesis for the degree of Master of Science

Dec. 26. 2003

Approved:

Chairman of the Committee
Kwangjo Kim, Professor
School of Engineering

Committee Member
Jae Choon Cha, Assistant Professor
School of Engineering

Committee Member
Dong Hoon Lee, PhD
National Security Research Institute

M.S. Chul-Joon Choi
20022153

A Study on Digital Signature Secure Against Key Exposure

School of Engineering, 2004, 56p.
Major Advisor : Prof. Kwangjo Kim.
Text in English

Abstract

A digital signature is one of the most widely used algorithms in software applications, due to its efficiency. It has a lot of types to generate signatures with its usage. Each algorithm offers their own security properties as well as standard security requirements. In practice, the greatest threat against the security of a digital signature scheme is exposure of the secret key, due to compromise of the security of the underlying system storing the key. The danger of successful cryptanalysis of the signature scheme itself is hardly as great as the danger of key exposure, as long as we stick to well known schemes and use large security parameters. But in real world, people are easy to lose their secret key and in most cases they are not aware of the fact. The most widely considered solution to the problem of key exposure is distribution of the key across multiple servers via secret sharing [10, 41]. However distribution of the key is quite expensive. Thus while we expect digital signatures to be very widely used, we do not expect most people to have the luxury of splitting their keys across several machines. Other ways of protecting against key exposure include use of protected hardware such as Smartcards, but these can be costly or impractical.

To protect some aspects of signature security against the exposure of the secret signing key, the notion of forward security is suggested, in particu-

lar without requiring distribution or protected storage devices, and without increasing key management costs. Anderson [1] used firstly the notion of forward secrecy in digital signature scheme. After that Bellare and Miner [11] suggested the first practical signature scheme that guarantees the forward security. And it is extended to intrusion resilient scheme by Itkis and Reyzin [25]. But these schemes are not compatible with previous conventional signature scheme.

With a different notion, Hwang, Lee and Lim [22] proposed c-times digital signature scheme which restricts the number of messages that can be signed. This scheme employs signature scheme that if a signer generates signatures more than threshold values c, then anyone can reveal the signer's secret key. By restricting the signing capability, we can reduce the risk of exposure of the secret signing key. This scheme is compatible with conventional scheme, but it has limitation in its usage.

In this thesis, we focus on the way to reduce the risk of exposure of secret signing key. We also consider the compatibility with the well-know digital signature scheme such as Schnorr signature scheme, ElGamal signature scheme and DSS (Digital Signature Standard). We propose key evolving forward secure signature with bilinear pairing based on G-DH problem. We prove the security of our scheme based on the fact that the C-DH problem is hard in the additive group of points of elliptic curve over a finite field. The security is analyzed in mathematical way such as cryptographic reduction. We also propose a practical $c - times$ signature. This scheme enables us to restrict the signing capability of proxy signer in terms of limiting the number of signatures. Our scheme is practical in the fact that it is compatible with conventional digital signature scheme.

Contents

Abstract **i**

Contents **iii**

List of Tables **vi**

List of Figures **vii**

List of Abbreviations **viii**

List of Notations **ix**

1 Introduction **1**

1.1 Key Exposure Problem 1

1.1.1 Distribution Paradigm via Secret Sharing 2

1.1.2 Forward Security 2

1.1.3 All-Or-Nothing Transform 3

1.1.4 $c-times$ Signature Scheme 4

1.1.5 Other methods 4

1.2 Our Contribution 4

1.3 Outline of the thesis 5

2 Related Works **6**

2.1 Forward Secure Signature Scheme 6

2.1.1 A Forward Secure Digital Signature Scheme 7

2.1.2 Forward Secure Signatures with Optimal Signing and Verifying 8

2.2 Forward Secure Encryption Scheme 9

2.3 $c-times$ Signature Scheme 10

3 Preliminaries **12**

3.1 Weil Pairing 12

3.2 Bilinear Diffie-Hellman Assumption 14

3.2.1 Gap-problems 14

3.2.2 Gap Diffie-Hellman Assumption 15

3.2.3 Bilinear Diffie-Hellman problem 16

3.2.4 Bilinear Diffie-Hellman Assumption 17

4 Proposed Scheme **18**

4.1 Forward Secure Signature with Bilinear Pairing 19

4.1.1 Basic Ideas 19

4.1.2 Definition 21

4.1.3 The Protocol 24

4.1.4 Application to Schnorr Signature with Bilinear pairing 25

4.1.5 Validity of Signatures 26

4.2 A practical $c-times$ signature 28

4.2.1 Definition 28

4.2.2 Main idea 28

4.2.3 The Protocol 29

4.2.4 Application to Proxy Signature 30

5 Security Analysis **33**

5.1 Security Proof of Scheme I 33

5.2 Security Analysis of Scheme II 45

6 Comparisons **46**

6.1 Comparison of Scheme I 46

6.2 Comparison of Scheme II 48

7 Conclusion **49**

국문요약 **50**

Appendix

References **52**

Acknowledgements **57**

Curriculum Vitae **58**

List of Tables

6.1 Comparison of key evolving forward secure signature schemes . . . 47
6.2 Comparison of $c - times$ signature schemes 48

List of Figures

4.1 A concept of designing key evolving forward secure signature 20
4.2 Experiment of evaluating success probability 23

5.1 Mathematical reduction 34
5.2 Experiment of solving C-SE problem in group $\mathbb{G}_1$ 35
5.3 Experiment of solving C-DH problem in group $\mathbb{G}_1$ 36

List of Abbreviations

AONT All-Or-Nothing Transform

B-DH Bilinear Diffie-Hellman

C-DH Computational Diffie-Hellman

C-SE Computational Square Exponent

CMA Chosen Message Attack

D-DH Decisional Diffie-Hellman

DH Diffie-Hellman

DLP Discrete Logarithm Problem

DSS Digital Signature Standard

ERF Exposure Resilient Function

FSS Forward Secure Signature

G-DH Gap Diffie-Hellman

GQ Guillou-Quisquator

HIBE Hybrid Identity Based Encryption

HLL Hwang, Lee and Lim scheme

LKK Lee, Kim and Kim scheme

MOV Menzes, Okamoto and Vanstone reduction

PPT Probabilistic Polynomial Time

List of Notations

$\left.\begin{array}{c}\mathcal{A}, \mathcal{A}_{PK}, \mathcal{A}_{Sig} \\ \mathcal{A}_{CSE}, \mathcal{A}_{CDH}, \mathcal{A}_{DDH} \\ \mathcal{A}_{BDH}, \mathcal{A}_{GDH}, \mathcal{A}_{FSS}\end{array}\right\}$ adversary

e, e_q Weil pairing

$\hat{e}$, $\hat{e}_q$ modified Weil pairing

E elliptic curve

$E(K)$ group of K-rational points on E

$E[k]$ group of k-torsion points on the elliptic curve E

$\left.\begin{array}{c}\varepsilon, \varepsilon, \varepsilon_{PK} \\ \varepsilon_{Sig}, \varepsilon_{CDH}, \varepsilon_{CSE}\end{array}\right\}$ success probability

$\mathbb{G}_1$, $\mathbb{G}_2$ cyclic groups of a prime order

h, h_A, h_P hashed results

H, H_1, H_2, H_3 hash operations

$\mathcal{IG}$ key generation algorithm which is modelled as PPT

k a security parameter

$K, \overline{K}$ for a field K, algebraic closure

m_W a warrant information in proxy signature

m a message

$\mathcal{M}$ message space

N a large composite number ($= pq$)

$\mathcal{O}$ point at infinity (on an elliptic curve)

Ω a set of ω_i

ω_i hashed value of g^{α_i}

$\mathcal{P}_i$ a session public key set

P, Q elements of cyclic group $\mathbb{G}_1$

Ψ random integer set

$\left.\begin{array}{c} PK \\ y, y_A, y_B \end{array}\right)$ public keys, public informations

q a large prime, the order of cyclic group $\mathbb{Z}_q^*$, $\mathbb{G}_1$, $\mathbb{G}_2$

q_{hash} a number of hash query

q_{sig} a number of signature query

$\left.\begin{array}{c} r, r_P, r_i \\ \alpha, \alpha_i \end{array}\right)$ randomly selected integers in $\mathbb{Z}_q^*$

$\sigma, \sigma_A, \sigma_P$ signatures

$\left.\begin{array}{c} S, SK, SK_i \\ x, x_A, x_B, x_P \end{array}\right)$ secret keys, secret informations

Succ success probability

InSec insecurity function

t, t_{PK}, t_{Sig}, t_{CDH}, t_{CSE} running time

T total time period

$\mathbb{Z}_q^*$ a group under multiplication modulo q

Chapter 1
Introduction

As computer becomes popular, a digital signature is used widely in software application. It offers security properties of authentication and identification. It has a lot of types to generate signatures with its usage. It is efficient and gives us convenience. But if we expose a secret signing key, all the security system can be broken. As a result, we are damaged by the threat of exposure of secret signing key. In this thesis, we argue on the risk of key exposure problem in digital signature scheme and construct digital signature scheme secure against key exposure problem.

1.1 Key Exposure Problem

A great deal of cryptography can be seen as finding ways to leverage the possession of a small but totally secret piece of knowledge (a secret key) into the ability to perform many useful and complex actions: from encryption and decryption to identification and message authentication. But if we expose a secret signing key, all the security system can be broken. It has been pointed out that key exposure is one of the greatest threats to security in practice [11]. There are several types of solutions to the problem of key exposure. We classify it with five categories.

1.1.1 Distribution Paradigm via Secret Sharing

The most widely considered solutions to the problem of key exposure are distribution of keys across multiple servers via secret sharing. Secret sharing schemes were discovered independently by Blakley [10] and Shamir [41]. The motivation for secret sharing is secure key management. In some situations, there is usually one secret key that provides access to many important files and identification to some contracts. If such a key is lost or exposed (e.g., the person who knows the key becomes unavailable, the computer which stores the key is destroyed, or someone takes the key without permission), then all the important files become inaccessible and all the contracts can be forged. The basic idea in secret sharing is to divide the secret key into pieces and distribute the pieces to different persons so that certain subsets of the persons can get together to recover the key. The general model for secret sharing is called an t-out-of-n scheme (or (t, n)-threshold scheme) for integers $(1 \leq t \leq n)$. In the scheme, there is a sender (or dealer) and n participants. The sender divides the secret into n parts and gives each participant one part so that any t parts can be put together to recover the secret, but any $t-1$ parts reveal no information about the secret. The pieces are usually called shares or shadows. Different choices for the values of t and n reflect the tradeoff between security and reliability. A secret sharing scheme is perfect if any group of at most $t-1$ participants (insiders) has no advantage in guessing the secret over the outsiders. Distribution across many systems, however, is quite expensive. Such an option may be available to large organizations, but is not realistic for the average user.

1.1.2 Forward Security

The goal of forward security is to protect some aspect of signature security against the risk of exposure of the secret signing key without increasing key management costs remarkably. As Bellare and Miner mentioned at [11], the

idea of "*Forward Security*" is that a distinction can be made between the security of documents pertaining to the past and those pertaining to the period after key exposure. More specifically, forward security property means that even if the current secret key is compromised, a forger cannot forge signatures for past time periods. This is accomplished by dividing the total time that given public key is valid into T *time period*, and using a different secret key in each time period while the public key remains fixed. Each subsequent secret key is computed from the current secret key via key update algorithm. The time period during which a message is signed becomes a part of the signature.

1.1.3 All-Or-Nothing Transform

In contrast with distribution paradigm, **E**xposure **R**esilient **F**unction(ERF)[12] [16] and All-or-nothing transform (AONT)[9] enable a single user to protect itself against partial key exposure on a single machine. A natural idea would be to use a secret sharing scheme to split the key into shares, and then attempt to provide protection by storing these shares instead of storing the secret key directly. Roughly speaking, an AONT is a randomized mapping which can be efficiently inverted if given the output in full, but which leaks no information about its input to an adversary even if the adversary obtains almost all the bits of the output. The AONT has been shown to have important cryptographic applications ranging from increasing the efficiency of block ciphers to protecting against almost complete exposure of secret keys . The first formalization and constructions for the AONT were given by Boyko [9] in the random oracle model. However, recently Canetti et. al. [12] were able to formalize and exhibit efficient constructions for the AONT in the standard computational model. They accomplished this goal by reducing the task of constructing AONT's to constructing a related primitive which they called an Exposure-Resilient Function(ERF). An ERF is a deterministic function

whose output looks random to an adversary even if the adversary obtains almost all the bits of the input.

1.1.4 $c - times$ Signature Scheme

The $c-times$ signature schemes [22] can be used for the temporary restriction of signing ability while the the cryptographic properties of the underlying signature system is maintained. This scheme restricts the number of messages that can be signed . They used c degree polynomial $f(z)$ for restricting the number of times of the signature. It employs signature scheme that if a signer generates signatures more then threshold value c, then anyone can calculate the signer's secret key using Lagrange interpolation method. And this characteristic is achieved by revealing partial information about secret key using polynomial. This scheme is quite efficient in the sense that it is compatible with DSS (Digital Signature Standard), ElGamal and Schnorr signature scheme, but it has to keep additional information to limit signing capability.

1.1.5 Other methods

Another widely considered proposal is the use of specially protected hardware such as smartcards, which can also be costly or inapplicable to many contexts. Thus, the cost or inconvenience of such solutions may make them prohibitive for many application; some users simply may not have the luxury to afford the investment such solutions would require.

1.2 Our Contribution

Over the years, many schemes have been offered. Distribution paradigm and Exposure-Resilient Function are quite strong but costly. $c - times$ Signature Scheme is somewhat weak and can be used special purpose. And hardware

solutions such as smartcards are also be costly or inapplicable to many contexts. In this thesis, we construct a new key evolving forward secure signature scheme based on the given hard problem, which is compatible with conventional signature scheme. And additionally, we construct a practical $c-times$ signature based on Schnorr signature which can be applicable to proxy signature scheme. This is the first contribution of this thesis. We present a formal model for key evolving forward secure signature scheme based on Gap Diffie-Hellman problem where C-DH problem is hard but D-DH problem is easy to solve and make the definition of security for this model. We generalize the method to design a forward secure signature. In the security model, we prove that our scheme guarantee the forward security if the C-DH problem is intractable. This is the second contribution of this thesis

1.3 Outline of the thesis

In this thesis, we deal with the risk of key exposure problem and discuss the way to reduce the damage of exposure of secret signing key. The rest of thesis is organized as follows. In Chapter 2, several forward secure signatures are reviewed. Chapter 3 contains cryptographic primitives and definitions which forms the basis of our scheme. In Chapter 4, we propose two kinds of solutions. At first, our key evolving forward secure signature scheme is presented based on G-DH problem. Secondly, a practical $c-times$ signature scheme is presented and applied to proxy signature scheme. In Chapter 5, we give a formal proof of security for our scheme. In Chapter 6, we make a comparison with several existing forward secure signature scheme. we end with concluding remarks in Chapter 7.

Chapter 2
Related Works

2.1 Forward Secure Signature Scheme

The idea of a digital signature scheme with forward security was suggested by Anderson [1] in an invited lecture presented at the year 1997 ACM-CCS conference. It was formalized by Bellare and Miner [11] in the context of a forward-secure signature scheme. The basic idea presented in [11] is the use of a key-evolving signature scheme whose operation is divided into time-periods, with a different secret key being used for each time period. Each secret key is used to sign the message in the current time-period and derive the secret key for the next time-period. Like the ordinary signature scheme, the public key is constant for all time-periods. A verification scheme checks both the signature's validity and time-period. The signature scheme is forward-secure because it is impossible for an adversary to forge a signature for a previous time-period even if it obtains the current secret key. Following the initial work by [11], a sequence of other deviations of the Forward-secure signature [3, 24, 25, 28, 29, 32] was suggested. In [3], an improved forward-secure signature scheme with much shorter keys than those outlined in [11] was proposed. Krawczyk [29] suggested a method for constructing a forward-secure signature scheme from any signature scheme, and thus made the forward security of standard signature schemes possible. Itkis and Reyzin [25] proposed another forward-secure signature scheme based on the Guillou-Quisquater [19] signature. It provides efficient signing, verifying and storage, but the price tag

for these features is a longer running time for the key generation and update routine. Integrating forward-secure signature with threshold techniques has also been investigated in [2]. Forward-secure signature scheme using bilinear map was suggested by Hu, Wu and Irwin at [24]. In that scheme, they update keys using binary tree method which was introduced in [13]. Key insulated cryptography [15] and intrusion resilient cryptography [25] were recently introduced to achieve a higher level of security.

2.1.1 A Forward Secure Digital Signature Scheme

Bellare and Miner [11] suggested the first practical signature scheme that guarantees forward security. The idea of this scheme is as follows :

KEYS AND KEY GENERATION : The signer's public key contains a modulus N and l points $U_1, \cdots, U_l$ in $\mathbb{Z}_N^*$. The corresponding base secret key SK_0 contains points $S_1, \cdots, S_l$ in $\mathbb{Z}_N^*$, where S_j is a 2^{T+1}-th root of U_j for $j = 1, \cdots, T$. The modulus is a Blum-Williams integer, meaning the product of two distinct primes each congruent to $3 \bmod 4$. They refer to U_i as the i -th *component* of the public key. The public key PK is treated like that of any ordinary signature scheme as far as registration, certification and key generation are concerned. The factors p, q of N are deleted once the key generation process is completed, so that they are not available to an attacker that might later break into the system on which the secret key is stored.

KEY EVOLUTION : During time period j, the signer signs using key SK_j. This key is generated at the start of period j, by applying a key update algorithm to the key SK_{j-1}. The update algorithm $Upd(SK_{j-1})$ squares the l points of the secret key at the previous stage to get the secret key at the next stage. Once this update is performed, the signer deletes the key SK_{j-1}. Thus key exposure during period j will yield to an attacker the current key (and also future keys) but not past keys. Notice that the components of the

secret key for period j are related to those of the base key as follows:

$$(S_{1,j}, \cdots, S_{l,j}) = (S_{1,0}^{2^j}, \cdots, S_{l,0}^{2^j}) \tag{2.1}$$

SIGNING : Signature generation during period j is done via the algorithm $Sgn_{SK_j}^H(M)$. It takes as input the secret key SK_j of the current period, the message m to be signed, and the value j of the period itself, to return a signature $\langle j, (Y, Z)\rangle$, where (Y, Z) are values in $\mathbb{Z}_N^*$. The signer first generates the "commitment" Y , and then hashes $Y$and the message m via a public hash function $H_l : \{0,1\} \rightarrow \{0,1\}^l$ to get an l-bit "challenge" $c = c_1, \cdots, c_l$ which is viewed as selecting a subset of the components of the public key. The signer then computes a 2^{T+1-j}-th root of the product of the public key components in the selected subset, and multiplies this by a 2^{T+1-j}-th root of Y to get the value Z. Thus, in the first time period, the signer is computing 2^T -th roots; in the second time period,2^{T-1}-th roots; and so on, until the last time period, where it is simply computing square roots. Notice that in the last time period, It simply has the Fiat-Shamir signature scheme.

VERIFYING : Verification of a candidate signature $\langle j, (Y, Z)\rangle$ for a given message m with respect to a given public key PK is performed via the $Vf_{PK}^H(m, \langle j, (Y, Z)\rangle)$. Note the verification process depends on the claimed "time-stamp" or period indicator j in the signature, meaning that the period j too is authenticated.

2.1.2 Forward Secure Signatures with Optimal Signing and Verifying

Itkis and Reyzin [25] proposed forward secure signature scheme based on GQ scheme.

SIGNING AND VERIFYING : The distinguishing feature of this scheme is the efficiency of the signing and verification algorithms. Both are the same as the previous efficient ordinary GQ scheme (verifying has the additional, negligible component of testing whether e is in the right range and odd).

KEY GENERATION : This scheme needs to make strong assumptions on the distributions of primes in order to estimate efficiency of key generation. First, it assumes that at least one in $O(k)\lceil k/2\rceil$-bit numbers is a prime, and that at least one in $O(k)$ of those is of the form $2q+1$, where q is prime. Then, generating n takes $O(k^2)$ primarily tests. Each primarily test can be done in $O(k^3)$ bit operations. Thus, the modulus N is generated in $O(k^5)$ bit operations. Similarly, They assume that at least one in $O(l)$ integers in each bucket $[2^l(1+(i-1)/T), 2^l(1+i/T)]$ is a prime, so generating each e_i takes $O(l^4)$ bit operations. In addition to generating N and the e_i's, key generation needs to compute the product of the e_i's modulo $\phi(N)$, which takes $O(Tkl)$ bit operations, and three modular exponentiations, each taking $O(k^2l)$ bit operations. Therefore, key generation takes $O(k^5 + l^4T + k^2l + klT))$ bit operations.

KEY UPDATE : Key update cannot multiply all the relevant e_i's modulo $\phi(N)$, because $\phi(N)$ is not available (otherwise, the scheme would not be forward-secure). Therefore, it has to perform $O(T)$ modular exponentiations separately, in addition to regenerating all the e_i's. However, for practical values of l (on the order of 100) and k (on the order of 1000), l^4T is roughly the same as k^2lT, so this only slows down the key update algorithm by a small constant factor.

2.2 Forward Secure Encryption Scheme

Canetti, Halevi, and Katz [13] proposed forward secure public key encryption scheme based on bilinear Diffie-Hellman assumption. This scheme assumes for simplicity that the total number of time periods T is a power of 2; that is , $T = 2^l$. Then, a full binary tree of height l can be made. This scheme let $< i >$ denote the $l - bit$ representation of integer i (where $0 \leq i \leq 2^t - 1$). The leaves of the tree correspond to successive time periods. There is a fixed public key PK associated with the tree, and every node w in the tree has

an associated secret key sk_w. However, only the secret keys of the leaves are directly used for decryption. The other keys are kept only to help derive the leaf keys when needed. The properties that needed from these keys are as follows:

1. To decrypt a message encrypted using PK during period i, only the leaf key $sk_{<i>}$ is needed.

2. Given a key of an internal node, sk_w, it is possible to efficiently derive keys for its children sk_{w0} and sk_{w1}.

3. It is infeasible to decrypt messages encrypted during period i (and, in particular, to derive the secret key of a leaf $< i >$) without knowledge of the secret key of one of the ancestors of leaf $< i >$ in the tree.

The above requirements essentially imply the forward-security of this scheme. This scheme is based on hierarchical identity-based encryption scheme (HIBE) [18]

2.3 $c - times$ Signature Scheme

Hwang, Lee and Lim[22] proposed $c-time$ signature scheme. They formalized the notion of $c-times$ signature schemes and construct an implicit $c-times$ signature scheme ${}^cDS = ({}^c\mathsf{KG}, {}^c\mathsf{Sig}, {}^c\mathsf{VF})$ using conventional digital signature scheme based on discrete logarithm problem(DLP) : $SDS = (\mathsf{KG}, \mathsf{Sig}, \mathsf{VF})$.
Key generation algorithm : Given input 1^k, the key generation algorithm ${}^c\mathsf{KG}$ first generates a DLP-based triple parameter (p, q, g). Next it runs KG on input 1^k to generate a key pair $(x, y := g^x)$ for the underlying signature scheme SDS. Then it generates a polynomial $f(z)$ of degree c over $\mathbb{Z}_q^*$ with ϵ as its constant coefficient. Namely it selects uniformly coefficients $(\epsilon_1, ..., \varepsilon_c)$ in $\mathbb{Z}_q^*$ for $f(z)$, $(1 \leq i \leq c)$. It outputs $PK = (y, (g^{\epsilon_1}, ..., g^{\epsilon_c}), (p, q, g))$ as the public key and $SK = (x, (\epsilon_1, ..., \epsilon_c))$ as the secret key.

SIGNING ALGORITHM :. The signing algorithm ${}^c\mathsf{Sig}$, on input the secret key SK and a message $m \in \{0,1\}^*$, computes $\sigma = \mathsf{Sig}_{SK}(m)$ and $f(h) = x+\epsilon_1 h+\epsilon_2 h^2+\ldots+\epsilon_c h^c \bmod q$ where $h = H_3(m,\sigma)$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$ is a public hash function. Then it outputs a signature, ${}^c\sigma = (\sigma, f(h))$.

VERIFICATION ALGORITHM : The verification algorithm ${}^c\mathsf{VF}$ on input PK, m and ${}^c\sigma$, outputs 1 if $\mathsf{VF}(m,\sigma,y) = 1$ and $g^{f(h)} = y\cdot(g^{\epsilon_1})^h\cdot(g^{\epsilon_2})^{h^2}\cdot\ldots\cdot(g^{\epsilon_c})^{h^c}$, 0 otherwise.

LEGAL FORGERY PROPERTY : For given $c+1$ signature values and their corresponding messages, the secret value of the signer is calculated by using the Lagrange interpolation formula. For a given

$$\left(\begin{array}{c|c} & 1 \le i \le c+1 \\ & {}^c\sigma \leftarrow \mathsf{Sig}_x(m_i) \\ ({}^c\sigma_i, m_i, PK) & 1 \leftarrow {}^c\mathsf{VF}(m,\sigma,y) = 1 \\ & m \in \mathcal{M} \\ & {}^c\sigma_i = (\sigma_i, f(z_i)) \\ & z_i = H_2(m_i,\sigma_i) \end{array}\right) \tag{2.2}$$

computes $f(l) = \sum_{i=1}^{c+1} f(z_i) \prod_{j=1, j\neq i}^{c+1} \frac{l-z_j}{z_i-z_j} \pmod q$ is easy.

This scheme enables us to restrict the signing capability of the signer up to $c-times$ which was predefined in key generation step.

Chapter 3
Preliminaries

3.1 Weil Pairing

We can make use of any bilinear map on an elliptic curve to construct a group $\mathbb{G}$ in which the C-DH problem is intractable, but the D-DH problem is tractable [4, 5, 7, 26, 31]. In particular, we make use of bilinear maps, in particular the Weil-pairing.

Let E be an elliptic curve over a base field K and let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of order q for some large prime q. The *Weil pairing* [4, 5, 7, 8, 33, 40] is defined by a bilinear map e between these groups,

$$e_q : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$$

It can e shown that the following properties hold. Let $P, Q \in \mathbb{G}_1$.

(i) *Identity*: For all $P \in \mathbb{G}_1$, $e_q(P, P) = 1$.

(ii) *Alternation*: For all $P, Q \in \mathbb{G}_1$, $e_q(P, Q) = e_q(Q, P)^{-1}$.

(iii) *Bilinearity*: For all $P_1, P_2, P_3 \in \mathbb{G}_1$, $e_q(P_1 + P_2, P_3) = e_q(P_1, P_3) \cdot e_q(P_2, P_3)$ and $e_q(P_1, P_2 + P_3) = e_q(P_1, P_2) \cdot e_q(P_1, P_3)$.

(iv) *Non-degeneracy*: If $e_q(P, Q) = 1$ for all $Q \in \mathbb{G}_1$, then $P = \mathcal{O}$, where $\mathcal{O}$ is a point at infinity.

(v) If $\mathbb{G}_1 \subset E(K)$, then $e_q(P, Q) \in K$ for all $P, Q \in \mathbb{G}_1$ (that is $\mathbb{G}_2 \subset K^*$).

In addition to these properties, we have an efficient algorithm to compute $e_q(P, Q)$ for all $P, Q \in \mathbb{G}_1$. In practice, in our basic scheme, we employ the *modified Weil pairing* $\hat{e}_q(P, Q) = e_q(P, \phi(Q))$, where ϕ is an automorphism on the group of points of E [5, 7]. For more details, we can refer to [5, 8, 33].

We write the Weil pairing and the modified Weil pairing as e and $\hat{e}$ in the place of e_q and $\hat{e}_q$ respectively.

As noted in [5], the existence of the bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ as above has two direct implications to these groups.

The MOV reduction: Menezes, Okamoto, and Vanstone[34] show that DLP in $\mathbb{G}_1$ is no harder than DLP in $\mathbb{G}_2$. To see this, let $P, Q \in \mathbb{G}_1$ be an instance of DLP in $\mathbb{G}_1$ where both P, Q have order q. We wish to find an $a \in \mathbb{Z}_q$ such that $Q = aP$. Let $g = \hat{e}(P, P)$ and $h = \hat{e}(Q, P)$. Then, by bilinearity of $\hat{e}$ we know that $h = g^a$. By non-degeneracy of $\hat{e}$ both g and h have order q in $\mathbb{G}_2$. Hence, we reduced DLP in $\mathbb{G}_1$ to DLP in $\mathbb{G}_2$. It follows that for discrete log to be hard in $\mathbb{G}_1$ we must choose our security parameter so that discrete log hard in $\mathbb{G}_2$.

Decision DH is easy: The D-DH problem [6] in $\mathbb{G}_1$ is to distinguish between the distributions $\langle P, aP, bP, abP\rangle$ and $\langle P, aP, bP, cP\rangle$ where a, b, and c are random in $\mathbb{Z}_q$ and P is random in $\mathbb{G}_1$. Joux and Nguyen [26] point out that D-DH in $\mathbb{G}_1$ is easy. To see this, observe that given $\{P, aP, bP, cP\} \in \mathbb{G}_1^*$ we have

$$c = ab \bmod q \iff \hat{e}(P, cP) = \hat{e}(aP, bP).$$

The C-DH problem in $\mathbb{G}_1$ can still be hard. Joux and Nguyen [26] give examples of mappings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where C-DH in $\mathbb{G}_1$ is believed to be hard even though D-DH in $\mathbb{G}_1$ is easy.

3.2 Bilinear Diffie-Hellman Assumption

3.2.1 Gap-problems

The computational assumptions when constructing cryptographic schemes can mainly be classified into two types. One is the intractability of an inverting problem such as inverting the RSA function, and computing the Diffie-Hellman (DH) problem. The other is the intractability of a decision problem such as the D-DH problem.

In addition to these problems, Okamoto and Pointcheval [38] define a new class of problems, called the Gap-problems. Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be any relation. The inverting problem of f is the classical computational version, and we can define a generalization of the decision problem, by the R-decision problem of f, for any relation

$$R : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\},$$

- The *inverting problem* of f is, given α, to compute any β such as $f(\alpha, \beta) = 1$ if it exists, or to answer Fail.

- The *R-decision problem* of f is, given (α, β), to decide whether $R(f, \alpha, \beta) = 1$ or not. Here β may be the null string, $\perp$.

The Gap-problem deals with the gap of difficulty between these problems. The Gap-problem can be defined as follows:

Definition 3.1 *The* R-gap problem *of* f *is to solve the inverting problem of* f *with the help of the oracle of the* R*-decision problem of* f*.*

Okamoto and Pointcheval [38] claimed that the DH problems are the typical instance of the Gap-problem. Since the inverting problem can be viewed as the computational problem, C-DH problem corresponds to the inverting one, and D-DH problem does to the R-decision one. Here, we describe the G-DH problem. Let $\mathbb{G}$ be any group of prime order q.

- The C-DH problem: given a triple of $\mathbb{G}$ elements (g, g^a, g^b), find the element $C = g^{ab}$.

- The D-DH problem: given a quadruple of $\mathbb{G}$ elements (g, g^a, g^b, g^c), decide whether $c = ab \pmod q$ or not.

- The G-DH problem: given a triple of $\mathbb{G}$ elements (g, g^a, g^b), find the element $C = g^{ab}$ with the help of a D-DH oracle (which answers whether a given quadruple is a DH quadruple or not).

The Tate-pairing is given as a specific example that satisfies the property of the G-DH problem [38]. For example [38], with an elliptic curve $E = J(\mathbb{F}_q)$ of trace $t = 2$ and $q = \#E = q + 1 - t = q - 1$, we have $J_q(\mathbb{F}_q) = J(\mathbb{F}_q)/qJ(\mathbb{F}_q) = E$ and $\mu_q(\mathbb{F}_q) = \mathbb{F}_q^*$. Then

$$e : E \times E \to \mathbb{F}_q^*,$$

which is called officially a bilinear map. Let us consider a DH quadruple, P, $A = a \cdot P$, $B = b \cdot P$ and $C = c \cdot P$,

$$e(A, B) = e(a \cdot P, b \cdot P) = e(P, P)^{ab} = e(P, ab \cdot P) = e(P, C).$$

And the latter equality only holds with the correct candidate C.

3.2.2 Gap Diffie-Hellman Assumption

Let $\mathcal{IG}$ be a B-DH parameter generator, and let $\mathcal{A}_{\mathsf{CDH}}$ be an algorithm whose input consists of a group $\mathbb{G}_1$ of prime order q, and algorithm $\mathcal{A}_{DDH}$ solving D-DH problem, a generator P of $\mathbb{G}_1$, aP and bP $(a, b \in Z_q^*)$ and whose output is an element of $\mathbb{G}_1$ that is expected to be abP. As usual the advantage of $\mathcal{A}_{\mathsf{CDH}}$ with respect to $\mathcal{IG}$ is defined to be

$$\Pr\left[\mathcal{A}_{\mathsf{CDH}}(\mathbb{G}_1, \mathcal{A}_{DDH}, P, aP, bP) = abP \;\middle|\; \begin{array}{c}(\mathbb{G}_1, \mathcal{A}_{DDH}) \longleftarrow \mathcal{IG}(1^k),\\ P \xleftarrow{R} \mathbb{G}_1,\\ a, b \xleftarrow{R} \mathbb{Z}_q^*\end{array}\right]$$

$\mathcal{IG}$ is said to satisfy the G-DH *assumption* if any polynomial time algorithm $\mathcal{A}_{\mathsf{CDH}}$ has $Adv \leq 1/f(k)$ for polynomial f, that is, no polynomial time algorithm can solve C-DH problem with not-negligible advantage.

3.2.3 Bilinear Diffie-Hellman problem

Since the D-DH problem in $\mathbb{G}_1$ is easy, we cannot use the D-DH problem to build cryptosystems in the group $\mathbb{G}_1$. Instead, the security of our protocol is based on a variant of the C-DH problem called the bilinear Diffie-Hellman (B-DH) problem.

Definition 3.2 The B-DH problem in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is the following: given (P, aP, bP, cP) for some $a, b, c \in \mathbb{Z}_q^*$, compute $v \in \mathbb{G}_2$ such that $v = \hat{e}(P, P)^{abc}$.

B-DH parameter generator: We say that a randomized algorithm $\mathcal{IG}$ is a B-DH *parameter generator* if

(1) $\mathcal{IG}$ takes a security parameter $0 < k \in \mathbb{Z}$,

(2) $\mathcal{IG}$ runs in polynomial time in k, and

(3) $\mathcal{IG}$ outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and the description of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

We require that the groups have the same prime order $q = |\mathbb{G}_1| = |\mathbb{G}_2|$. We denote the output of $\mathcal{IG}$ by $\mathcal{IG}(1^k)$. A concrete example of the B-DH parameter generator is given in [5] as follows. Given a security parameter k the B-DH parameter generator picks a random k-bit prime p such that $p = 2 \bmod 3$ and $p = 6q - 1$ for some prime q. The group $\mathbb{G}_1$ is the subgroup of order q of the group of points on the elliptic curve $y^2 = x^3 + 1$ over $\mathbb{F}_p$. The group of $\mathbb{G}_2$ is the subgroup of order q of $\mathbb{F}_{p^2}^*$. The bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is the modified Weil paring defined in Section 3.1.

Note that the isomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ induced by the Weil pairing are one-way functions [5, 7]. For a point $Q \in \mathbb{G}_1^*$ defines the isomorphism $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ by $f_Q(P) = \hat{e}(P, Q)$. It is well known that an efficient algorithm for inverting f_Q would imply an efficient algorithm for deciding D-DH in the group $\mathbb{G}_2$. Throughout this thesis the D-DH problem is believed to be hard in the group $\mathbb{G}_2$. Hence, all the isomorphisms $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ are believed to be one-way functions.

3.2.4 Bilinear Diffie-Hellman Assumption

Let $\mathcal{IG}$ be a B-DH parameter generator. We say that an algorithm $\mathcal{A}_{BDH}$ has advantage $\varepsilon_{BDH}(k)$ in solving the B-DH problem for $\mathcal{IG}$ if for sufficiently large k;

$$Adv_{\mathcal{A}_{BDH}(k)} = \Pr\left[\mathcal{A}_{BDH}\begin{pmatrix} q, \hat{e}, \\ \mathbb{G}_1, \mathbb{G}_2, \\ P, aP, \\ bP, cP \end{pmatrix} = \hat{e}(P,P)^{abc} \quad \left\langle \begin{matrix} q, \hat{e}, \\ \mathbb{G}_1, \mathbb{G}_2 \end{matrix} \right\rangle \leftarrow \mathcal{IG}(1^k), \\ P \leftarrow \mathbb{G}_1^*, \\ a, b, c \leftarrow \mathbb{Z}_q^* \right] \geq \varepsilon_{BDH}(k)$$

We say that $\mathcal{IG}$ satisfies the B-DH assumption if for any randomized polynomial time (in k) algorithm $\mathcal{A}_{BDH}$ and for any polynomial $f \in \mathbb{Z}[x]$ we have that $Adv_{\mathcal{IG},\mathcal{A}}(k) < 1/f(k)$ for sufficiently large k. When $\mathcal{IG}$ satisfies the B-DH assumption we say that B-DH is hard in groups generated by $\mathcal{IG}$.

Chapter 4
Proposed Scheme

In this chapter, we propose digital signature scheme secure against key exposure problem. As we mentioned in section 1.1, there are several approaches to overcome the risk of key exposure problem. Here we suggest two kinds of solutions. One belongs to the notion of forward security and the other to $c-times$ signature scheme. At first, we will discuss a new method to guarantee the forward security and suggest key evolving forward secure signature scheme. This scheme generalize the key evolving methods that can applicable to standard signature scheme which is based on pairing. Secondly, we suggest a practical $c-times$ signature scheme. By using this scheme, we can restrict the signing capability of the signer within the threshold value c.

Scheme I

4.1 Forward Secure Signature with Bilinear Pairing

4.1.1 Basic Ideas

There are several ways to design a forward secure signature scheme. Bellare and Miner [11] employed Binary certification tree scheme in which key and signature sizes are logarithmic in T. But basically, this method fixes public key that it is not easy to design forward secure signature scheme which is compatible with conventional signature. Therefore we aim to build efficient way of designing forward secure signature scheme. We generalize the method to evolve the key pairs in time period j. The concept is shown in figure 4.1. In each period j, a signer updates key pairs $(SK_j, \mathcal{P}_j)$ which we call as j-th session secret key and session public key. To illustrate, the signer begins by generating a key pair (SK_0, PK) of the standard scheme. He publishes the master public key (PK) of the key evolving scheme at public directory and sets the starting secret key of the key evolving scheme to SK_0. At the start of period 1 the signer creates a new session key pairs $(\mathcal{P}_1, SK_1)$, and deletes SK_0. The signature of a message m in period 1 is $(1, \langle h_1, \sigma_1 \rangle = \mathsf{Sign}_{SK_1}(m), \mathcal{P}_1)$. A signature $(1, \langle h_1, \sigma_1 \rangle, \mathcal{P}_1)$ on message m is verified by checking that $\mathsf{VF}_{\mathcal{P}_1}(\sigma_1) = 1$ and $\mathsf{VF}_{PK}(\mathcal{P}_1) = 1$. This continues iteratively, so that at the start of period $j > 2$, the signer, in possession of SK_{j-1} and $\mathcal{P}_{j-1}$ of previous period, creates a new session key pair $(\mathcal{P}_j, SK_j\ )$. The signature of a message m in period j is $(j, \langle h_j, \sigma_j \rangle = \mathsf{Sign}_{SK_j}(m), \mathcal{P}_j)$. A signature $(j, \langle h_j, \sigma_j \rangle, \mathcal{P}_j)$ on a message m is verified by checking that $\mathsf{VF}_{\mathcal{P}_j}(\sigma_j) = 1$ and $\mathsf{VF}_{PK}(\mathcal{P}_j) = 1$.

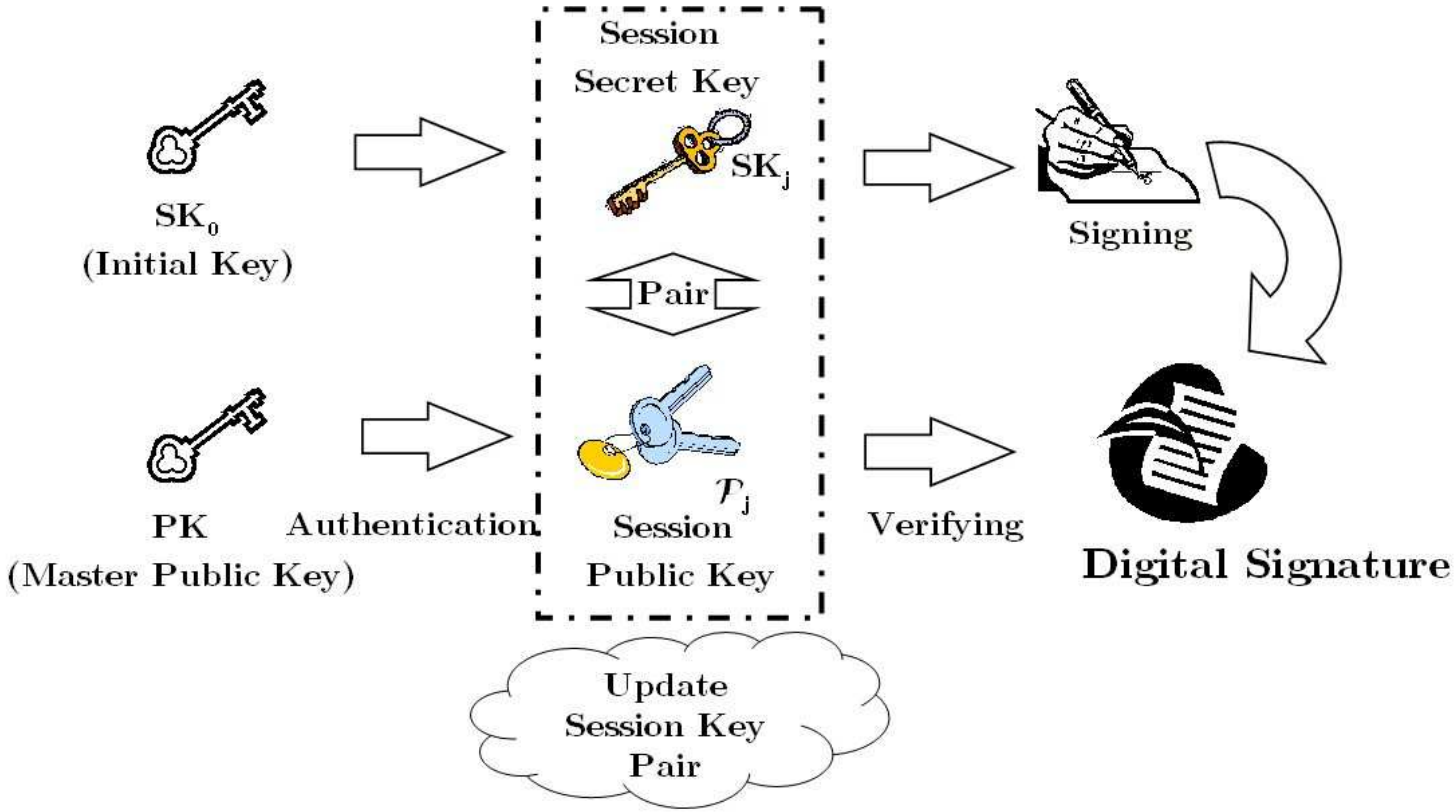

Figure 4.1: A concept of designing key evolving forward secure signature

4.1.2 Definition

Here we provide definitions of key evolving digital signature scheme. First, we define the form of algorithms to specify such schemes, and then discuss security. This definition is straightforward adaption of [11] except that verification algorithm is more complicated.

Definition 4.1 *Key Evolving Digital Signature Scheme* FSS=(Gen, Upd, Sign, VF) *is a 4-tuple of PPT algorithms such that :*

- The probabilistic *key generation algorithm* Gen takes as input a secret parameter 1^k and the total number of time periods T. It returns a master public key PK and an initial secret key SK_0.
- The probabilistic *key update algorithm* Upd takes as input the secret signing key SK_{j-1} of the previous period and an index j. It returns the j-th session secret key SK_j of the current period and j-th session public key $\mathcal{P}_j$.
- The probabilistic *Signing algorithm* Sign takes as input a message m and the secret signing key SK_j of the current period. It returns a signature $(j, \langle h_j, \sigma_j \rangle = \mathsf{Sign}_{SK_j}(m), \mathcal{P}_j)$ of message m for period j.
- The deterministic *Verification algorithm* VF takes the public key PK, message m and candidate signature $(j, \langle h_j, \sigma_j \rangle, \mathcal{P}_j)$. It returns a bit, with 1 meaning accept and 0 meaning reject. This *Verification algorithm* VF consists of two algorithms. One is session public key verification algorithm $\mathsf{VF}_{PK}(\mathcal{P}_j)$ and the other is signature verification algorithm $\mathsf{VF}_{\mathcal{P}_i}(\sigma_j)$.

We say that $(j, \langle h_j, \sigma_j \rangle, \mathcal{P}_j)$ is a valid signature of m for period j if $\mathsf{VF}_{\mathcal{P}_j}(\sigma_j) = 1$ and $\mathsf{VF}_{PK}(\mathcal{P}_j) = 1$.

In this scheme, we use random oracle model. The algorithm above additionally have oracle access to a public hash function H_2 which in the security analysis is assumed random.

The notion of forward security is that it should be computationally infeasible for any adversary to forge a signature for any past time period even in the event of exposure of the current secret key. To define forward security formally, the notion of a secure digital signature of [20] is extended in [11] and [25] to take into account the ability of the adversary to obtain a key by means of a break-in.

In this model, similar with [11, 25] the forger first conducts an adaptive chosen message attack (CMA), requesting signatures on messages of its choice for as many time periods as he desires. Whenever he chooses, he "breaks in": requests the secret key SK_j for the current time period j and then outputs an (alleged) signature on a message m of his choice for a time period $i < j$. The forger is considered to be successful if the signature is valid and the pair (m, i) was not queried during CMA.

The success probability in breaking the forward security of the signature scheme is evaluated by the experiment in figure 4.2.

In this formulation, it is understood that the state of $\mathcal{A}_{FSS}$ is preserved across its various invocations, once we first pick and fix for it. The adversary first gets access to an oracle for generating signature under SK_1. It queries this as often as it wants, and indicates it is done by outputting some value d. We move in to the next stage, providing the adversary an oracle for signing under the next key. Note that the process is strictly ordered; once an adversary gives up the oracle for signing under SK_j , by moving into the next phase or breaking-in, it cannot obtain access to that oracle again. At some point the adversary decides to use its break-in privilege, and is returned the key SK_j of the stage in which it did this. (If it does not break-in by the last period, we give it the key SK_j where $j = T + 1$. And this key is empty string.) It will now try to forge signatures under SK_i for some $i < j$ and is

Experiment $\mathcal{A}$**-forge**$(FSS, \mathcal{A}_{FSS})$

$(PK, SK) \xleftarrow{R} \mathsf{Gen}(1^k, \cdots, T)$

$j \leftarrow 0$

Repeat

$j \leftarrow j+1; (SK_j, PK_j) \leftarrow \mathsf{Upd}(SK_{j-1})$;

$d \leftarrow \mathcal{A}_{FSS}^{\mathsf{Sign}_{Sk_j}(\cdot)}(\mathrm{CMA}, PK_0)$

Until (d = break-in) or $(j = T)$

If (d $\neq$ break-in and $j = T$) then $j \leftarrow T+1$

$(m, i, \langle h_i, \sigma_i \rangle, \mathcal{P}_i) \leftarrow \mathcal{A}_{FSS}(forge, SK_j)$

If $(VF_{\mathcal{P}_i}(\sigma_i) = 1$ and $VF_{PK}(\mathcal{P}_i) = 1$ and

$(1 \leq i < j)$ and m is not queried of $\mathsf{Sign}_{SK_i}(\cdot)$ in period $i)$

then return 1 else return 0

Figure 4.2: Experiment of evaluating success probability

declared successful if the signature is valid and the message is new. Here we define the forward security which closely follows the one in [11]

Definition 4.2 (Standard Forward Security) *Let* $FSS = (\mathsf{Gen}, \mathsf{Upd}, \mathsf{Sign}, \mathsf{VF})$ *be a key-evolving signature scheme, and* $\mathcal{A}_{FSS}$ *an adversary attacking its forward security as described above. We let* $\mathsf{Succ}^{fwsig}(FSS[k,T], \mathcal{A}_{FSS})$ *denote the probability that experiment* $\mathcal{A} - Forge(FSS[k,T], \mathcal{A}_{FSS})$ *returns 1. Then the insecurity function of* FSS *is defined as*

$$\mathsf{InSec}^{fwsig}(FSS[k,T], t, q_{sig}, q_{hash}) = \max_{A}\{\mathsf{Succ}^{fwsig}(FSS[k,T], \mathcal{A}_{FSS})\}$$

where the maximum is taken over all adversaries $\mathcal{A}_{FSS}$ *for which the following condition holds : the execution time for the above experiment is at most* t*,* $\mathcal{A}_{FSS}$ *makes at most* q_{sig} *signing queries to the signing oracle and* q_{hash} *queries to the random hash oracle.*

We define $\mathsf{Succ}^{FSS}(FSS[k,T], \mathcal{A}_{FSS})$ to be the probability that $\mathcal{A}_{FSS}$ is successful and let the function $\mathsf{InSec}^{FSS}(FSS[k,T], t, qsig)$ (the insecurity

function) be the maximum, over all algorithms $\mathcal{A}_{FSS}$ that are restricted to running time t and q_{sig} signature queries, of $\mathsf{Succ}^{FSS}(FSS[k,T], \mathcal{A}_{FSS})$. The insecurity function above gives us a measure of how secure or insecure the scheme really is. Therefore, we want its value to be as small as possible. Our goal in a security proof is to find its upper bound.

4.1.3 The Protocol

Our scheme is based on G-DH problem in pairing and requires a hash function $H_2 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ which is assumed to be random.

Key Generation Algorithm

For a security parameter k, a pair of secret and public parameter is generated as follows :

Algorithm $\mathsf{Gen}(1^k, T)$

1. Let k be a security parameter, and q be a large prime
2. Run $\mathcal{IG}(1^k)$ to generate groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order q and,
   bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$
   Select two random hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*,\ H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^k$
   Select a large prime $s \in \mathbb{Z}_q^*$ and set $P = H_1(ID)$ and $S_{pub} = sH_1(ID)$
3. Select a large prime $r_q \in \mathbb{Z}_q^*$, and Set $x_0 = r_q \mod q$
   Set $Q_0 = x_0 P$ and $\mathcal{P}_0 = \phi$
4. Compute $x_j = x_{j-1}^2 \bmod q\ (1 \leq j \leq T)$
   Set $PK = (q, k, T, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q_0, S_{pub}, H_1, H_2)$ and
   $SK_0 = (\{(S_j, Q_j) | Q_j = x_j P,\ S_j = x_j S_{pub},\ (1 \leq j \leq T)\},\ \mathcal{P}_0)$
5. Delete $s, r_q, x_j,\ (0 \leq j \leq T)$
6. Return (PK, SK_0)

Key Update Algorithm

Key update algorithm takes (SK_{j-1}, j) as inputs and outputs $(SK_j, \mathcal{P}_j)$.

Algorithm $\mathsf{Upd}(SK_{j-1}, j)$

1. Set $\mathcal{P}_j = \mathcal{P}_{j-1} \cup (Q_j \in SK_{j-1})$
2. Delete S_{j-1} and set $SK_j = (\{(S_l, Q_l)|\ (j \leq l \leq T)\},\ \mathcal{P}_j)$
3. Return $(SK_j, \mathcal{P}_j)$

Signing and Verifying Algorithm

This scheme follows conventional signing and verifying algorithm except that additional session public key verifying algorithm is added in verification step.

Algorithm $\mathsf{VF}(m, j, \langle h_j, \sigma_j \rangle, \mathcal{P}_j))$

▶ *Additional session public key verifying algorithm*

For $l = 1$ to j

if $\hat{e}(Q_l, P) \neq \hat{e}(Q_{l-1}, Q_{l-1})$

then reject signature

else continue

▶ *From here, conventional verification algorithm starts*

4.1.4 Application to Schnorr Signature with Bilinear pairing

In this section, we present one example of our scheme based on Schnorr signature scheme with Bilinear pairings [23]. Key evolving mechanism (Gen, Upd) is used same way with above.

Signing Algorithm

Algorithm $\mathsf{Sign}(m, SK_j)$

1. Select a random number α in $\mathbb{Z}_q^*$
2. Compute $u = \hat{e}(\alpha P, P)$ and $h_j = H_2(m, u)$, then signature is $\sigma_j = h_j S_j + \alpha P$
3. return $(j, \langle \sigma_j, h_j \rangle, \mathcal{P}_j\ )$

Verifying Algorithm

Algorithm $\mathsf{VF}(m, j, \langle h_j, \sigma_j \rangle, \mathcal{P}_j))$

1. For $l = 1$ to j
   if $\hat{e}(Q_l, P) \neq \hat{e}(Q_{l-1}, Q_{l-1})$ then reject signature else continue
2. Compute $\bar{u} = \frac{\hat{e}(\sigma_j, P)}{\hat{e}(S_{pub}, Q_j))^{h_j}}$ and $\bar{h}_j = H_2(m, \bar{u})$
3. if and only if $h_j = \bar{h}_j$ then return 1, else return 0

4.1.5 Validity of Signatures

We construct a forward secure signature protocol. To guarantee the validity of our scheme, we should check that signatures generated by the signing process will always be accepted by the verification process.

Proposition 4.3 Let PK and SK_0 be a key pair generated by the *key generation* algorithm of FSS. Let $(j, \langle h_j, \sigma_j \rangle, \mathcal{P}_j))$ be an output of Sign (m, SK_j). Then $VF_{\mathcal{P}_i}(\sigma_i) = 1$ and $VF_{PK}(\mathcal{P}_i) = 1$.

Proof: (Case 1)[Validity of Session Public Key] From $\mathcal{P}_j$, we can extract Q_{l-1} and $Q_l (1 \leq l \leq j)$ and we have Q_0 from master public key. Then it is

essy to show that Q_l is the l-th session public key using D-DH problem

$$\begin{aligned}\hat{e}(Q_l, P) &= \hat{e}(x_0^{2^l}P, P) = \hat{e}(P,P)^{x_0^{2^l}} = \hat{e}(P,P)^{x_0^{2^{l-1}} \cdot x_0^{2^{l-1}}} \\ &= \hat{e}(x_0^{2^{l-1}}P, x_0^{2^{l-1}}P) = \hat{e}(Q_{l-1}, Q_{l-1})\end{aligned}$$

(Case 2)[Validity of Signatures] Let $Q_j \in \mathcal{P}_j$ be a j-th session public key and $\bar{h_j} = H_2(m, \bar{u_j})$, then we check that $\bar{u_j} = \frac{\hat{e}(\sigma_j, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}}$

$$\begin{aligned}\bar{u_j} &= \frac{\hat{e}(\sigma_j, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \frac{\hat{e}(h_j S_j + \alpha P, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \frac{\hat{e}(h_j S_j, P)\hat{e}(\alpha P, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \frac{\hat{e}(S_j, P)^{h_j}\hat{e}(\alpha P, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \frac{\hat{e}(S_{pub}, P)^{x_0^{2^j} \cdot h_j} \cdot \hat{e}(\alpha P, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \frac{\hat{e}(S_{pub}, x_0^{2^j}P)^{h_j} \cdot \hat{e}(\alpha P, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \frac{\hat{e}(S_{pub}, Q_j)^{h_j} \cdot \hat{e}(\alpha P, P)}{\hat{e}(S_{pub}, Q_j)^{h_j}} \\ &= \hat{e}(\alpha P, P) = u_j\end{aligned}$$

we get as desired ■

Scheme II

4.2 A practical $c-times$ signature

In this chapter, we propose Schnorr signature scheme with restricted signing capability which is a variant of $c-times$ signature scheme [22] and we apply it to proxy signature scheme. We modify LKK [30] scheme to meet the requirement of our scheme, and we construct Schnorr-based proxy signature scheme with restricted signing capability.

4.2.1 Definition

We denote t as a number of pre-selected random secret integers. And we require a hash function, $H_3 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$. Additional notations are described as follows:

Definition 4.4 Let t be a small integer, and $\mathbf{\Psi}$, $\mathbf{\Omega}$ be sets having the following characteristics,

a) A set,
$\mathbf{\Psi} = \{\alpha_i \mid \alpha_i \in_R \mathbb{Z}_q^*,\ 1 \leq i \leq t\}$

b) A set,
$\mathbf{\Omega} = \{\omega_i \mid \omega_i = H_3(g^{\alpha_i} \mid\mid i),\ \alpha_i \in \Psi,\ 1 \leq i \leq t\}$

4.2.2 Main idea

The main idea of our scheme is that we pre-select random integer set $\mathbf{\Psi}$ in key generation phase. And we publish corresponding set $\mathbf{\Omega}$ at public directory. In verification step, verifier checks whether the hashed value of $v = g^{\sigma} \cdot y^{-e} \bmod p$ is the element of $\mathbf{\Omega}$ or not. This means that if the hash value of v is not an element of $\mathbf{\Omega}$, the signature is not valid and should be rejected. If the signer

uses a random secret value $\alpha_i \in \mathbf{\Psi}$ twice, then, the signer's secret key x is revealed as following,

$$\sigma_1 = x \cdot h_1 + \alpha_i \mod q, \quad h_1 = H_3(m_1 || \ g^{\alpha_i}) \tag{4.1}$$

$$\sigma_2 = x \cdot h_2 + \alpha_i, \mod q, \quad h_2 = H_3(m_2 || \ g^{\alpha_i}) \tag{4.2}$$

Equation (3.1) - (3.2) is

$$(\sigma_1 - \sigma_2) \equiv x \cdot (h_1 - h_2) \pmod q$$
$$\Rightarrow x \equiv \frac{(\sigma_1 - \sigma_2)}{h_1 - h_2} \pmod q$$

The signer can use α_i only one time to generate signature without revealing of his secret key. In that reason, the signer can generate limited number of signatures.

4.2.3 The Protocol

Our scheme is a variant of Schnorr signature scheme. So it employs a subgroup of order q in $\mathbb{Z}_p^*$, where p is some large prime number. This scheme also requires a hash function $H_3 \ : \ \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ as does in Schnorr signature scheme [35].

Key Generation

1. Select prime numbers q and p with the property that q divides $(p-1)$.
2. Select a generator g of the unique cyclic group of order q in $\mathbb{Z}_p^*$.
   - Select an element $\gamma \in \mathbb{Z}_p^*$ and,
   - Compute $g = \gamma^{(p-1)/q} \mod p$.

- If $g = 1$ then repeat 2.

3. Select a random integer x such that $1 \leq x \leq q - 1$.
4. Compute $y = g^x \bmod p$.
5. Select random integer set $\mathbf{\Psi}$.
6. Compute a set $\mathbf{\Omega}$ where $\omega_j = H_3(g^{\alpha_j}||j)$, and $\alpha_j \in \mathbf{\Psi}$.
7. Keep x and $\mathbf{\Psi}$ as a secret value, and publish $p,\ q,\ g,\ y$ and a set $\mathbf{\Omega}$.

Signature Generation

1. Select a random secret value $\alpha_j \in \mathbf{\Psi}$.
2. Compute $R = g^{\alpha_j} \bmod p,\ \ h = H_3(m||R||j)$, and $\sigma = x \cdot h + \alpha_j \bmod q$.
3. Alice's signature for m is the pair $(\sigma,\ h,\ j)$

Signature Verification

1. Obtain Alice's authentic public key $\{p,\ q,\ g,\ y\}$ and set $\mathbf{\Omega}$.
2. Compute $\bar{R} = g^{\sigma} \cdot y^{-h} \bmod p$, and $\bar{h} = H_3(m||\bar{R}||j)$.
3. Accept the signature if and only if $h = \bar{h}$ and $H_3(\bar{R}||j) \in \mathbf{\Omega}$.

4.2.4 Application to Proxy Signature

We apply our scheme to proxy signature scheme. We demonstrate the proxy signature scheme with limited number of signatures. our scheme restrict the number of signatures in the range of 1 to t where t is the number of pre-selected random secret integers.

Proxy Generation

- Proxy signer *Bob* : (x_B, y_B)

  1. Select random secret integer set Ψ and,
  2. Compute corresponding public integer set $\mathbf{\Omega}$.
  3. Send public integer set $\mathbf{\Omega}$ to original signer, *Alice*, using public channel.

- Original signer *Alice* : (x_A, y_A)

  1. Select a random integer r_A in $\mathbb{Z}_q^*$.
  2. Compute public value $R_A = g^{r_A} \bmod p$.
  3. Compute $\sigma_A = x_A \cdot h_A + r_A \bmod q$, where $h_A = H_3(m_w||R_A||\mathbf{\Omega})$.
  4. Send $(h_A,\ \sigma_A,\ m_w)$ to proxy signer, *Bob*, using public channel.
  5. Original signer, *Alice* publishes $\mathbf{\Omega}$ at public directory.

Proxy Key Generation

- Proxy signer *Bob* : (x_B, y_B)

  1. Proxy signer, Bob does the following step
     - Compute $\bar{R}_A = g^{\sigma_A} \cdot y_A^{-h_A} \bmod p$.
     - Compute $\bar{h}_A = H_3(m_w||\bar{R}_A||\mathbf{\Omega})$.
     - Accept if and only if $h_A = \bar{h}_A$.
  2. Compute proxy key $x_p = \sigma_A + x_B \bmod q$.

Proxy Signature Generation

- Proxy signer *Bob* : (x_P, y_B)

  1. Select $\alpha_j \in \mathbf{\Psi}$, and compute $R_j = g^{\alpha_j} \bmod p$.

2. Generate a proxy signature $\sigma_j = x_P \cdot h_P + \alpha_j \bmod q$, where $h_P = H_3(m||R_j||j)$.
3. Send $(\sigma_j,\ h_A,\ h_P,\ m,\ m_w,\ R_A)$ to verifier $Carol$.

Proxy Signature Verification

- Verifier $Carol$:

1. Compute $\bar{R}_j = g^{\sigma_j} \cdot (y_A^{h_A} \cdot y_B \cdot R_A)^{-h_P} \bmod p$,
2. Compute $\bar{h}_P = H_3(m||\bar{R}_j||j)$.
3. Accept if and only if $H_3(\bar{R}_j||\ j) \in \mathbf{\Omega}$ and $h_P = \bar{h}_P$. If the check is hold simultaneously, then the signature is valid.

Chapter 5
Security Analysis

5.1 Security Proof of Scheme I

The exact security of our scheme is close to that of [11, 24, 25]. It closely follows the one in [3] and [11], combining ideas from [4, 25, 31].

We are able to prove that as long as C-DH problem is computationally intractable, it is computationally infeasible to break the forward security of our scheme. First, we state the Theorem 5.1 that will allow us to upper-bound the insecurity function.

Theorem 5.1 *If there exists an adversary $\mathcal{A}_{FSS}$ for $FSS[k,T]$ that runs in time at most t_{FSS}, asking at most q_{hash} queries and q_{Sig} signing queries, such that $\mathsf{Succ}^{fwsig}(FSS[k,T],\mathcal{A}_{FSS}) \geq \varepsilon_{FSS}$, then there exist an algorithm $\mathcal{A}_{CDH}$ that solves C-DH problem in $\mathbb{G}_1$ in expected time at most t_{CDH} with probability at least ε_{CDH}, where*

$$t_{CDH} = \max\{3t_{EPK} + O(2^k), \quad 2t_{Sig} + O(k^2)\}$$

$$\varepsilon_{FSS} \leq \frac{(q_{hash}+1)(q_{Sig}+T)}{2^k} + T\sqrt{+(q_{hash}+1)\varepsilon_{CDH}} + \sqrt[3]{\varepsilon_{CDH}}$$

Proof: [Outline] In Lemma 5.5 and 5.6, we compute t_{EPK}, ε_{EPK}, t_{Sig} and ε_{Sig}. and in Lemma 5.8, we prove that the maximum success probability of FSS is the sum of the maximum success probability of two subroutine algorithms EPK and Sig. ■

To prove our scheme, we divide our scheme into two subroutine algorithm EPK and Sig. We reduce the complexity of our scheme to the complexity of C-DH problem as can be seen in figure 5.1.

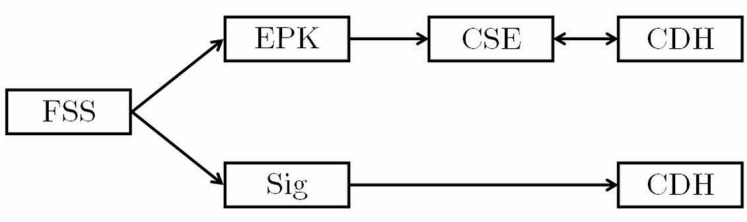

Figure 5.1: Mathematical reduction

Firstly, we will prove that public key evolving algorithm is computationally infeasible as long as C-DH problem is computationally intractable. Secondly, we will also prove that signature algorithm is computationally infeasible as long as C-DH problem is computationally intractable. Finally, we will show that FSS is forward secure using these two proofs.

Case 1. [Public Key Evolving Algorithm] We start with the public key evolving algorithm. we use C-SE problem as a bridge of reducing the complexity of our scheme to C-DH problem. We define two experiment as can be seen in 5.2 and 5.3.

Let $ECSE(\mathcal{A}_{CSE}, k)$ be any algorithm that takes input $\mathcal{A}_{CSE}$ and k, attempts to return instance of the C-SE problem, and let t_{CSE} be the running time and ε_{CSE} be the success probability that above $ECSE(\mathcal{A}_{CSE}, k)$ algorithm solving the square exponent problem in group $\mathbb{G}_1$.

Let $ECDH(\mathcal{A}_{CDH}, k)$ be any algorithm that takes input $\mathcal{A}_{CDH}$ and k, attempts to return instance of the C-DH problem, then we define

Definition 5.2 *Let $\mathcal{A}_{CDH}$ be an adversary for* C-DH *problem and let* $\mathsf{Succ}^{CDH}(\mathcal{A}_{CDH}, k)$ *denote the probability that experiment $ECDH(\mathcal{A}_{CDH}, k)$ returns 1.*

$ECSE(\mathcal{A}_{CSE}, k)$

1. Generate $\langle q, \hat{e}, \mathbb{G}_1, \mathbb{G}_2 \rangle \leftarrow \mathcal{IG}(1^k)$
2. Select $P \xleftarrow{R} \mathbb{G}_1$
3. Repeat
   - Select $x \xleftarrow{R} \mathbb{Z}_q^*$
   - Compute $Q = xP$, $Q_2 = x^2P$
   - Run $Q' \longleftarrow \mathcal{A}_{CSE}(Q, P)$
   - if $Q_2 = Q'$ then return 1, else return 0

Figure 5.2: Experiment of solving C-SE problem in group $\mathbb{G}_1$

The insecurity of C-DH *problem is the function*

$$\mathsf{InSec}^{CDH}(\mathcal{A}_{CDH}, k) = \max_{A}\{\mathsf{Succ}^{CDH}(\mathcal{A}_{CDH}, k)\}$$

where the maximum here is taken over all adversaries $\mathcal{A}_{CDH}$ for which the above experiment runs in time at most t.

To prove that public key evolving algorithm is computationally infeasible as long as C-DH problem is computationally intractable, we begin with the following lemma.

Lemma 5.3 *If there exist an adversary $\mathcal{A}_{EPK}$ for session public key forgery attack to our scheme with running time t_{EPK}, and probability ε_{EPK} then their exist an adversary $\mathcal{A}_{CSE}$ for solving* C-SE *problem in $\mathbb{G}_1$ which has running time t_{CSE} and probability ε_{CSE}, where $t_{CSE} = t_{EPK}$ and $\varepsilon_{CSE} = \varepsilon_{EPK}$*

Proof: If the public key evolving oracle of our scheme $\mathcal{O}_{EPK}$ is given, the adversary of C-SE problem $\mathcal{A}_{CSE}$ queries to $\mathcal{O}_{EPK}$ with input $(1, Q_0 = x_0P)$, then $\mathcal{O}_{EPK}$ outputs $Q_1 = x_0^2P$. It directly implies that $\mathcal{A}_{CSE}$ succeeds to solves the C-SE problem. The running time $t_{CSE} = t_{EPK}$ and the success probability is $\varepsilon_{CSE} = \varepsilon_{EPK}$. We get as desired ■

$ECDH(\mathcal{A}_{CDH}, k)$

1. Generate $\langle q, \hat{e}, \mathbb{G}_1, \mathbb{G}_2 \rangle \leftarrow \mathcal{IG}(1^k)$
2. Select $P \xleftarrow{R} \mathbb{G}_1$
3. Repeat
   - Select $x \xleftarrow{R} \mathbb{Z}_q^*$ and $y \xleftarrow{R} \mathbb{Z}_q^*$
   - Compute $Q_x = xP$, $Q_y = yP$ and $Q_{xy} = xyP$
   - Run $Q' \longleftarrow \mathcal{A}_{CDH}(Q_x, Q_y, P)$
   - if $Q_{xy} = Q'$ then return 1, else return 0

Figure 5.3: Experiment of solving C-DH problem in group $\mathbb{G}_1$

From here, we will show that the complexity of C-SE problem is equivalent to that of C-DH problem in $\mathbb{G}_1$. We consider following Lemma.

Lemma 5.4 *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q, and there exist a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, then given an oracle $\mathcal{O}_{CSE}$ which breaks C-SE problem in $\mathbb{G}_1$ with success probability at least ε_{CSE}, there exists an adversary $\mathcal{A}_{CDH}$ that breaks C-DH problem in $\mathbb{G}_1$ with success probability at least ε_{CDH}.*

Proof: Similar proof is done by Maurer and Wolf in [37] and Sadeghi and Steiner in [42]. They proved the equivalence between the C-SE and C-DH problem in $\mathbb{Z}_q^*$.
The key observation is that

$$\hat{e}((x+y)^2P, P) = \hat{e}(2xyP, P)\hat{e}(x^2P, P)\hat{e}(y^2P, P)$$

This implies

$$\begin{aligned} \hat{e}((2xy)P, P) &= \hat{e}(((x+y)^2 - x^2 - y^2)P, P) \\ &= \hat{e}((x+y)^2P, P)\hat{e}(x^2P, P)^{-1}\hat{e}(y^2P, P)^{-1} \end{aligned}$$

Therefore, we can solve C-DH problem with three oracle calls (one for each of $\hat{e}((x+y)^2P, P)$, $\hat{e}((x)^2P, P)$ and $\hat{e}((y)^2P, P)$) and some additional computations such as inverse function.

[Running Time] To break the C-DH problem, there are 3 independent calls to the C-SE oracle and some additional computations which is in the order of group operation. Therefore, the running time $t_{CDH} = 3t_{CSE} + O(2^k)$
[Probability] There are 3 independent calls to the C-SE oracle and thus the success probability of algorithm $\mathcal{A}_{CDH}$ is $\varepsilon_{CDH} = \varepsilon_{CSE}^3$ ■

Lemma 5.5 *If there exist an adversary $\mathcal{A}_{EPK}$ for session public key forgery attack to our scheme with running time t_{EPK}, and probability ε_{EPK} then their exist an adversary $\mathcal{A}_{CDH}$ for solving C-DH problem in $\mathbb{G}_1$ which has running time t_{CDH} and probability ε_{CDH}, where $t_{CDH} = 3t_{EPK} + O(2^k)$ and $\varepsilon_{CDH} = \varepsilon_{EPK}^3$*

Proof: Combining the Lemma 5.3 and 5.4, we get the result as we desired. The running time t_{CDH} is $3t_{CSE} + O(2^k)$ by Lemma 5.4 and $t_{EPK} = T \cdot t_{CSE}$ by Lemma 5.3 that $t_{CDH} = 3t_{EPK} + O(2^k)$. And the success probability ε_{CDH} is ε_{CSE}^3 by Lemma 5.4 and $\varepsilon_{CSE} = \varepsilon_{EPK}$ by Lemma 5.3 that $\varepsilon_{CDH} = \varepsilon_{EPK}^3$ ■

Case 2. [Signature Algorithm] To show that signature algorithm is computationally infeasible as long as C-DH problem is computationally intractable, we state the following theorem:

Lemma 5.6 *If there exist an adversary $\mathcal{A}_{Sig}$ for adaptive chosen message attack with fixed public key to our scheme with running time t_{Sig}, asking at most q_{hash} hash queries and q_{sig} signing queries, then their exist an adversary $\mathcal{A}_{CDH}$ for solving C-DH problem in $\mathbb{G}_1$ which has running time t_{CDH} and probability ε_{CDH}. where*

$$t_{CDH} = 2t_{Sig} + O(K^2)$$

$$\varepsilon_{Sig} = \frac{(q_{hash}+1)(q_{Sig}+T)}{2^k} + T\sqrt{(q_{hash}+1)\varepsilon_{CDH}}$$

Proof: [Oracle Reply Attack](Reduction to C-DH problem.) First, we assume that the adversary $\mathcal{A}_{Sig}$ outputs $(b,(h,\sigma))$ as a forgery, then the hashing oracle has to be queried on (b,u,m) where $u = \frac{\hat{e}(\sigma,P)}{\hat{e}(S_{pub},Q_b)^h}$. We will also assume that $\mathcal{A}_{Sig}$ performs necessary bookkeeping so as not to ask same hash query twice. Note that $\mathcal{A}_{Sig}$ may ask the same signature query twice, because the answers will most likely be different.

$\mathcal{A}_{CDH}$ has to guess the time period at which $\mathcal{A}_{Sig}$ will ask the break-in query: it randomly select j, $(1 < j \leq T+1)$ hoping that the break-in will occur at j or later, and the forgery will be for a time period earlier that j. It then sets $Q_j = x^{2^j}P$ and $PK = (Q_j, T)$, and runs the adversary $\mathcal{A}_{Sig}$ first time. Note that x is randomly selected in $\mathbb{Z}_q^*$ and x^{2^j} is modulus q. $\mathcal{A}_{CDH}$ then comes up with a random tape for $\mathcal{A}_{Sig}$, and runs $\mathcal{A}_{Sig}$ on the tape. $\mathcal{A}_{CDH}$ maintains two tables : a hash query table and a signing query table

Signature queries can be answered almost at random, because $\mathcal{A}_{CDH}$ controls the hash oracle. In order to answer a signature query number l_1 on a message m_{l_1} during time period j_{l_1}. $\mathcal{A}_{CDH}$ selects a random $\sigma_{l_1} \in \mathbb{G}_1$ and $h_{l_1} \in \{0,1\}^k$, computes $u_{l_1} = \frac{\hat{e}(\sigma_{l_1},P)}{\hat{e}(S_{pub},Q_{j_{l_1}})^h}$, and checks its signature query table to see if a signature query on m_{l_1} during time period j_{l_1} has already been asked and if u is used in answering it. If so, $\mathcal{A}_{CDH}$ changes σ_{l_1} and h_{l_1} to the σ and h that were used in answering that query. Then $\mathcal{A}_{CDH}$ adds the entry $(l_1, j_{l_1}, u_{l_1}, h_{l_1}, \sigma_{l_1}, m_{l_1})$ to its signature query table and outputs $(j_{l_1}, \sigma_{l_1}, h_{l_1})$.

Hash queries are also answered at random. To answer the l_2-th hashing query $(j_{l_2}, \sigma_{l_2}, m_{l_2})$, $\mathcal{A}_{CDH}$ first checks its signature query table to see if there is an entry $(l_1, j_{l_1}, u_{l_1}, h_{l_1}, \sigma_{l_1}, m_{l_1})$ such that $(j_{l_1}, \sigma_{l_1}, m_{l_1}) = (j_{l_2}, \sigma_{l_2}, m_{l_2})$. If so, it just outputs h_{l_1}. Otherwise, it picks a random $h_{l_2} \in \{0,1\}^k$, records in its hash query table $(l_2, j_{l_2}, u_{l_2}, h_{l_2}, \sigma_{l_2}, m_{l_2})$ and outputs h_{l_2}.

Now, we assume the break-in query occurs during time-period j, and the forgery $(b,(\sigma,h))$ is output for a time period $b < j$. To answer the break-in query, $\mathcal{A}_{CDH}$ outputs S_j. Let $u = \frac{\hat{e}(\sigma,P)}{\hat{e}(S_{pub},Q_b)^h}$. As we set $\mathcal{A}_{Sig}$ to first ask a hash query on (j,u,m), we have that, for some l, $(l,b,u,m,h) = (l, j_l, u_l, m_l, h_l)$ in

the hash query table. $\mathcal{A}_{CDH}$ finds such an l in its table and remember it. $\mathcal{A}_{CDH}$ now reset $\mathcal{A}_{Sig}$ with the same random tape as the first time, and runs it again, giving the exact same answers to all $\mathcal{A}_{Sig}$'s queries before the l-th hash query. This means $\mathcal{A}_{Sig}$ will ask the same l-th hash query (b, u, m). As soon as $\mathcal{A}_{Sig}$ asks the l-th hash query, $\mathcal{A}_{CDH}$ stops giving the answers from the tables and comes up with new answers at random, in the same manner as the first time.

Assume again the break-in query occurs during time-period j, and the forgery (b', σ', h') is output for a time period $b' < j$. If the second forgery was based on hash query number l, then we have $b = b'$, and

$$\begin{aligned}
u &= u' \\
\frac{\hat{e}(\sigma, P)}{\hat{e}(S_{pub}, Q_b)^h} &= \frac{\hat{e}(\sigma', P)}{\hat{e}(S_{pub}, Q_b)^{h'}} \\
\hat{e}(S_{pub}, Q_b)^{h-h'} &= \hat{e}(\sigma, P) \cdot \hat{e}(\sigma', P)^{-1} \\
\hat{e}(S_{pub}, P)^{x^{2^b} \cdot (h-h')} &= \hat{e}(\sigma - \sigma', P) \\
\hat{e}(x^{2^b} \cdot (h-h')S_{pub}, P) &= \hat{e}(\sigma - \sigma', P) \qquad (5.1)
\end{aligned}$$

From Eqn. 5.1, we get

$$\begin{aligned}
x^{2^b} \cdot (h-h')S_{pub} &= \sigma - \sigma' \\
x^{2^b} S_{pub} &= (\sigma - \sigma') \cdot (h-h')^{-1} \qquad (5.2)
\end{aligned}$$

As $S_{pub} = sH_1(ID) = sP$, that Equation 5.2 is $(x^{2^b})sP = (\sigma - \sigma') \cdot (h - h')^{-1}$. By knowing $Q_b = x^{2^b}P$ and $S_{pub} = sP$ and breaking forward security, we can solve C-DH problem. Therefore we show that the adversary $\mathcal{A}_{CDH}$ can solve C-DH problem. The proof is completed as we desired.

■

[Running Time Analysis] Adversary $\mathcal{A}_{CDH}$ runs $\mathcal{A}_{Sig}$ twice. Answering hashing and signing queries takes $\mathcal{A}_{CDH}$ no longer than it would the real oracles. To find the hashing query which the signature corresponds to, to compute $(\sigma - \sigma') \cdot (h - h')^{-1}$ takes one division and two substraction. Thus the running time of $\mathcal{A}_{CDH}$ exceeds that of $\mathcal{A}_{Sig}$ by $O(k^2)$, that is $t_{CDH} = 2t_{Sig} + O(k^2)$.

[Probability Analysis] First, we consider the following lemma.

Lemma 5.7 *Let* $a_1, a_2, \cdots, a_\lambda$ *be real numbers, Let* $a = \sum_{\mu=1}^{\lambda} a_\mu$. *Let* $\pi = \sum_{\mu=1}^{\lambda} a_\mu^2$. *Then* $\pi \geq \frac{a^2}{\lambda}$

Proof: Let $\rho = \frac{a}{\lambda}$ and $\rho_\mu = \rho - a_\mu$. Note that $\sum_{\mu=1}^{\lambda} \rho_\mu = \lambda\rho - \sum_{\mu=1}^{\lambda} a_\mu = a - a = 0$. Then

$$\sum_{\mu=1}^{\lambda} a_\mu^2 = \sum_{\mu=1}^{\lambda} (\rho - \rho_\mu)^2 = \lambda\rho^2 - 2\rho \sum_{\mu=1} \lambda\rho_\mu + \sum_{\mu=1} \lambda\rho_\mu^2 \geq \lambda\rho^2 = \frac{a^2}{\lambda}$$

This is done by Abdalla and Reyzin in [3] and we rewrite it here. ■

At first, we consider the probability that $\mathcal{A}_{CDH}$'s answer to $\mathcal{A}_{Sig}$'s oracle queries are distributed as those of the true oracles that $\mathcal{A}_{Sig}$ expects. Because σ is picked random in $\mathbb{G}_1$ and $u = \frac{\hat{e}(\sigma,P)}{\hat{e}(S_{pub},Q_b)^h}$ is a random element of $\mathbb{G}_2$. The probability of collision with a value from a hash query in the same execution of $\mathcal{A}_{Sig}$ is $\frac{q_{hash}+1}{2^k}$. Thus, the difference of probability between $\mathcal{A}_{CDH}$ and real signer is $\frac{q_{sig}(q_{hash}+1)}{2^k}$. Let $\delta = \varepsilon_{Sig} - \frac{q_{sig}(q_{hash}+1)}{2^k}$.
Let ε_j be the probability that $\mathcal{A}_{Sig}$ produces a successful forgery and that its break-in query occurs in time-period j. Then, $\delta = \sum_{i=2}^{T+1} \varepsilon_j$. If $\mathcal{A}_{CDH}$ picked a specific j as the time-period for break-in, Then the probability of that is $1/T$.
We will now calculate the probability of the event that $\mathcal{A}_{Sig}$ outputs a valid forgery based on the same hash query both times and that the hash query was answered differently the second time and that the break-in query was j both times. Let $\varepsilon_{<l,j>}$ be the probability that, in one run, $\mathcal{A}_{Sig}$ produces a valid forgery based on hash query number l after break-in query in time-period j. Then,

$$\varepsilon_j = \sum_{l=1}^{q_{hash}+1} \varepsilon_{<l,j>}$$

Let $\varepsilon_{<l,j,s>}$ (for a sufficiently long binary string s of length ζ) be the probability that, in one run, $\mathcal{A}_{Sig}$ produces a valid forgery based on hash query number l after break-in query in time-period j, given that the string s was used to determine the random tape of $\mathcal{A}_{Sig}$ and the responses to all the oracle queries of $\mathcal{A}_{Sig}$ until the l-th hash query. We have that

$$2^k \varepsilon_{<l,j>} = \sum_{s \in \{0,1\}^\zeta} p_{<l,j,s>}$$

Given such a fixed string s, the probability that $\mathcal{A}_{Sig}$ produces a valid forgery based on the hash query number l after break-in query in time-period j in both runs is $\varepsilon^2_{<l,j,s>}$. because the first forgery is now independent of the second forgery. The additional requirement that the answer to the hash query in the second run be different reduces this probability to $\varepsilon_{<l,j,s>} \cdot (\varepsilon_{<l,j,s>} - 2^{-k})$. Thus, the probability $\varepsilon_{<l,j>}$ that $\mathcal{A}_{Sig}$ produces a valid forgery based on the hash query number l in both runs and that the answer to the hash query is different in the second run and that the break-in query was j in both runs is

$$\begin{aligned}
\varepsilon_{<l,j>} &= \sum_{s \in \{0,1\}^\zeta} 2^{-\zeta} \varepsilon_{<l,j,s>} \cdot (\varepsilon_{<l,j,s>} - 2^{-k}) \\
&= 2^{-\zeta} \left(\sum_{s \in \{0,1\}^\zeta} \varepsilon^2_{<l,j,s>} - 2^{-k} \sum_{s \in \{0,1\}^\zeta} \varepsilon_{<l,j,s>} \right) \\
&\geq \frac{2^{-\zeta} (\varepsilon_{<l,j>} \cdot 2^\zeta)^2}{2^\zeta} - 2^{-k} \varepsilon_{<l,j>} = \varepsilon^2_{<l,j>} - 2^{-k} \varepsilon_{<l,j>}
\end{aligned}$$

by Lemma 5.7.
The probability that $\mathcal{A}_{Sig}$ outputs a valid forgery based on the same hash query both times and that the hash query was answered differently in the second run and that the break-in query occurred in time-period j is now

$$\sum_{h=1}^{q_{hash}+1} \varepsilon_{<l,j>} \geq \sum_{h=1}^{q_{hash}+1} \varepsilon^2_{<l,j>} - \sum_{h=1}^{q_{hash}+1} 2^{-k} \varepsilon_{<l,j>} \geq \frac{\varepsilon_j^2}{q_{hash}+1} - 2^{-k} \varepsilon_j$$

If this happens, then the forgery occurs in time-period $b < j$, because the forgery has to occur before the break-in period, so $\mathcal{A}_{CDH}$ will be able to solve the C-DH problem. Finally, we remove the assumption that $\mathcal{A}_{CDH}$ picked j as the time-period to get the probability of $\mathcal{A}_{CDH}$'s success:

$$\varepsilon_{CDH} \geq \frac{1}{T}\sum_{j=2}^{T+1}\left(\frac{\varepsilon_j^2}{q_{hash}+1} - 2^{-k}\varepsilon_j\right) \geq \frac{\delta^2}{T^2(q_{hash}+1)} - \frac{\delta}{2^k T}$$

by lemma 5.7.

Now, we remove the δ, then the success probability of $\mathcal{A}_{CDH}$ is

$$\varepsilon_{CDH} \geq \frac{(\varepsilon_{Sig} - \frac{q_{sig}(q_{hash}+1)}{2^k})^2}{T^2(q_{hash}+1)} - \frac{\varepsilon_{Sig} - \frac{q_{sig}(q_{hash}+1)}{2^k}}{2^k T} \tag{5.3}$$

We let $\theta = \left(\varepsilon_{Sig} - \frac{q_{sig}(q_{hash}+1)}{2^k}\right) \cdot T^{-1}$ then, we can define equation 5.3 as follows :

$$f(\theta) \stackrel{def}{=} 2^k\theta^2 - (q_{hash}+1)\theta - 2^k(q_{hash}+1)\epsilon_{CDH} \leq 0$$

As a function of θ the function f is first decreasing, then increasing. It has two roots, and we need upper bound of probability. By quadratic formula we have

$$\begin{aligned}
\theta &= \frac{(q_{hash}+1) + \sqrt{(q_{hash}+1)^2 + 2^{2(k+1)}(q_{hash}+1)\varepsilon_{CDH}}}{2^{k+1}} \\
&= 2^{-(k+1)}(q_{hash}+1) + 2^{-(k+1)}(q_{hash}+1) + \sqrt{(q_{hash}+1)\varepsilon_{CDH}} \\
&= 2^{-k}(q_{hash}+1) + \sqrt{(q_{hash}+1)\varepsilon_{CDH}}
\end{aligned} \tag{5.4}$$

As a result

$$\begin{aligned}
\varepsilon_{Sig} &= 2^{-k}q_{Sig}(q_{hash}+1) + T\cdot 2^{-k}(q_{hash}+1) + T\sqrt{(q_{hash}+1)\varepsilon_{CDH}} \\
&= \frac{(q_{hash}+1)(q_{Sig}+T)}{2^k} + T\sqrt{(q_{hash}+1)\varepsilon_{CDH}}
\end{aligned}$$

Until now, we show that two subroutine algorithm of FSS is infeasible under the assumption that C-DH problem is computationally intractable. To merge these two algorithm, we state the following lemma:

Lemma 5.8 *Let $\mathcal{A}_{EPK}$ and $\mathcal{A}_{Sig}$ be an adversary attacking our scheme with session public key forgery attack and adaptive chosen message attack with fixed session public key respectively. And we denote $\mathsf{Succ}^{EPK}(VF_{PK_0}(\mathcal{P}_i), \mathcal{A}_{EPK})$ is the probability that $VF_{PK_0}(\mathcal{P}_i)$ return 1, and $\mathsf{Succ}^{SIG}(VF_{\mathcal{P}_i}(\sigma_i), \mathcal{A}_{Sig})$ is the probability that $VF_{\mathcal{P}_i}(\sigma_i)$ return 1. Then the success probability of $\mathcal{A} - Froge$ is*

$$\begin{aligned}\mathsf{Succ}^{fwsig}(FSS[k,T], \mathcal{A}_{FSS}) \quad &\leq \quad \max\{\mathsf{Succ}^{EPK}(VF_{PK_0}(\mathcal{P}_i), \mathcal{A}_{EPK})\} \\ &+ \quad \max\{\mathsf{Succ}^{SIG}(VF_{\mathcal{P}_i}(\sigma_i), \mathcal{A}_{Sig})\}\end{aligned}$$

Proof: From experiment $\mathcal{A} - Forge$, an attacker $\mathcal{A}_{FSS}$ who tries to break $FSS[k,T]$ wins the game if he succeed to forge one of these algorithms. First, if he can make a valid session public key without secret information, then he also can make valid signatures corresponding to this public key. Furthermore, if he can forge valid signature without forging session public key, it is straightforward that he can break $FSS[k,T]$. Let Pr(C) be a $\mathsf{Succ}^{fwsig}(FSS[k,T], \mathcal{A}_{FSS})$ and Pr(A), Pr(B) be $\mathsf{Succ}^{EPK}(VF_{PK_0}(\mathcal{P}_i), \mathcal{A}_{EPK})$, $\mathsf{Succ}^{SIG}(VF_{\mathcal{P}_i}(\sigma_i), \mathcal{A}_{Sig})$ respectively then,

$$\begin{aligned}\Pr(\mathrm{C}) &= \Pr(\mathrm{A} \cup \mathrm{B}) \\ &= \Pr(\mathrm{A}) + \Pr(\mathrm{B}) - \Pr(\mathrm{A} \cap \mathrm{B}) \\ &\leq \Pr(\mathrm{A}) + \Pr(\mathrm{B})\end{aligned}$$

Therefore, the maximum success probability of $\mathcal{A} - Forge$ can be written as the sum of the maximum probability of two subroutine algorithms. We get as desired. ■

Theorem 5.1 allows us to state the following theorem about the insecurity function of our scheme.

Theorem 5.9 *Let $FSS[k,T]$ represent our key evolving signature scheme with security parameter k, and total time periods T. Then for any t, q_{Sig} and q_{hash},*

$$\begin{aligned}\mathsf{InSec}^{fwsig}(FSS[k,T],t,q_{sig},q_{hash}) \quad \leq \\ \frac{(q_{hash}+1)(q_{Sig}+T)}{2^k} + \sqrt[3]{\mathsf{InSec}^{CDH}(k,t_{CDH})} \\ + T\sqrt{+(q_{hash}+1)\mathsf{InSec}^{CDH}(k,t_{CDH})}\end{aligned}$$

where $t_{CDH} = max\{3t_{EPK} + O(2^k), \quad 2t_{Sig} + O(k^2)\}$

Proof: The upper bound of insecurity function is simply derived from Theorem 5.1. ■

The security parameter k must be chosen large enough that $\mathsf{InSec}^{CDH}(k,t_{CDH})$ is very low. The theorem tells us that the probability of being able to compromise the forward security of the signature scheme is also low.

5.2 Security Analysis of Scheme II

Basically, our scheme is the variant of Schnorr signature scheme and LKK scheme [30]. Only the difference is that we pre-select the random integers in key generation step. Therefore, as long as the random integers kept secretly, the security level of our scheme is almost same as that of Schnorr signatures [39]. Additionally, our scheme is a kind of $c-tiems$ signature. We will show that our scheme has $c-times$ characteristic.

$c-times$ characteristic

If a proxy signer generates signatures with same random value, $\alpha_i \in \mathbf{\Psi}$, then the secret key of the proxy signer can be revealed as follows:

$$\sigma_i = x_P \cdot h_i + \alpha_i \mod q, \quad h_i = H(m_i||R_i||i), R_i = g^{\alpha_i} \mod p \tag{5.5}$$

$$\sigma_j = x_P \cdot h_j + \alpha_i \mod q, \quad h_i = H(m_j||R_i||j), R_i = g^{\alpha_i} \mod p \tag{5.6}$$

Equation (5.5) - (5.6) is

$$x_P = \frac{\sigma_j - \sigma_i}{h_j - h_i} \pmod q \tag{5.7}$$

And

$$x_B = \sigma_A + x_P \mod q \tag{5.8}$$

By Eqs. (5.7) and (5.8), x_B is revealed.
In that reason, the proxy signer can not use $\alpha_i \in \mathbf{\Psi}$ twice.
We pre-select random numbers, $\alpha_i \in \mathbf{\Psi}$ in key generation step, and we publish corresponding hash values in public directory that the proxy signer can sign on at most t numbers of messages, so our scheme has $c-times$ characteristics.

Chapter 6
Comparisons

We assume that the elliptic curve is chosen in the same way as [5]. As pointed out in [5], from the practical point of view, we can assume that p and q is a 512-bit prime and 140-bit prime respectively, since the MOV reduction [34] leads to a DLP in a finite field of size approximately 2^{1024}. In addition, we assume that system parameter q for our basic scheme are 140-bit and the modulus N for other scheme is 1024-bit and the output of random hash value is 160-bit.

6.1 Comparison of Scheme I

In this section, we compare our key evolving forward secure signature scheme with the prior scheme in terms of compatibility, computational overhead and signature size. We denote M the cost of modular multiplication and E the cost of exponentiation over a given finite field and A the cost of addition and PA the cost of point addition and PM the cost of point addition over a given elliptic curve and B the cost of computing bilinear map and H the cost of hash operation. Table 6.1 shows the comparison of our key evolving forward secure signature scheme.

From Table 6.1, we can state of the properties of our scheme as follows: (1) Compare to Itkis and Reyzin scheme, our scheme is more efficient in signing algorithm but less efficient in verifying algorithm. Our scheme has overhead in signature size and public key verifying algorithm. As time goes

Table 6.1: Comparison of key evolving forward secure signature schemes

	Our scheme	Bellare and Miner	Itkis and Reyzin	Hu, Wu and Irwin
Security proof	Yes	Yes	Yes	Yes
Compatibility with Conventional Scheme	Yes (Schnorr)	NO (More precisely)	NO (More precisely)	No
Underlying Problem	G-DH	IFP	GQ	BDH
Comp. Complexity (Signing)	1PA+2PM +2B+1H	$(2+k)$M +kE+1H	(1)M+2E+1H	1PA+1PM +1H
Comp. Complexity (Verifying)	$\max\{T\}$B + 1H	$(k+1)$M+1H +$(k+1)$E	(1)M+2E+1H	$(1+\log T)$PM +$(2+2\log T)$B
Signature Size(bits)	$(140)(2+\max\{T\})$	(2048)	(1024) +2 log(1024)	$(1+\log T)\cdot(140)$
Public Key Size(bits)	140	$(1024)\cdot l$	(1024)	(140)
Secret Key Size(bits)	$(280)\cdot\max\{T\}$	$(1024)\cdot l$	(2048) +2 log(1024)	$2(\log T-1)\cdot(140)$

on, signature size increase proportion to the time period until total time period T. Therefore we have a further work to find more efficient one. (2) Our scheme is compatible with conventional signature scheme. This is a big advantage in practical use, because in real world, conventional signature scheme has been already used and changing this signature scheme to new one needs very high cost. Though, our scheme has some overhead, it is directly applicable to conventional scheme such as Schnorr signature scheme. (3) Our scheme use bilinear pairing that it is possible to extend to ID-based signature scheme. (4) Another merit of our scheme is that it can be applied to proxy signature scheme. Bellare and Miner's scheme and Itkis and Reyzin's scheme are very difficult to apply to proxy signature because they use modulus N which is a composite number.

6.2 Comparison of Scheme II

In this section, we compare our $c-times$ signature scheme whit that of HLL [22] scheme. we assume that the signer generates t number of signatures.

Table 6.2: Comparison of $c-times$ signature schemes

	Our scheme	HLL scheme
Comp. Complexity (Signing)	0 (Not Increase)	$t\mathsf{A} + t\mathsf{M}$ (Increase)
Comp. Complexity (Verifying)	$1\mathsf{H}$ (Increase)	$(t+1)\mathsf{E}+ t\mathsf{M}$ (Increase)
Signature Size(bits)	0 (Not Increase)	(160) (Increase)
Public Key Size(bits)	$t \cdot (160)$ (Increase)	$t \cdot (512)$ (Increase)

From Table 6.2 we can state that our scheme is more efficient than HLL scheme. Our scheme has an additional step to check the validity of random value $R_i = g^{\alpha_i}$ that it increase public key size by $t \cdot (160)$ and computational complexity of verifying step by one hash operation. But HLL scheme has to compute t-degree polynomials that public key size and computation complexity of singing and verifying is increased much more than that of our scheme.

Chapter 7
Conclusion

In this thesis, we study the risk of key exposure problem. We review previous works and present current concerns on key exposure problem. And then we present our suggestions to solve the problems. We suggest two kinds of solutions. In scheme I, we present a key evolving forward secure signature scheme based on the G-DH problem using bilinear pairing. The key evolving forward secure signature scheme is compatible with conventional signature scheme which is based on DH problem using the pairings. To guarantee that our scheme gives forward security, first we define the key evolving signature scheme and standard forward security. We show that any attacker that can break the forward security can be transformed into an efficient algorithm to solve the underlying well-studied problem, the C-DH problem in pairing. Finally, we obtain an exact analysis of the security of the scheme rather than asymptotic ones. In scheme II, we present a practical $c - times$ signature. This scheme can limit the signing capability of signer in terms of the number of signature. We apply this scheme with proxy signature scheme. And we can efficiently restrict the signing capability of proxy signer. As a result, we can protect original signer from the abuse of proxy signer's signing ability.
As a future work, it remains to find more efficient key evolving algorithm to guarantee forward security which is compatible with conventional signature scheme. We will apply this key evolving forward secure signature to proxy signature scheme as well.

키 노출에 안전한 전자서명 기법에 관한 연구

최철준

전자서명 기법은 그 효율성 때문에 응용소프트웨어에서 가장 많이 쓰이는 어플리케이션 중의 하나이다. 전자서명은 사용 용도에 따라 다양한 서명 생성 기법이 존재한다. 각 서명기법은 표준 보안요구 사항뿐만 아니라 사용 용도에 따른 고유한 보안요구 사항을 제공하고 있다. 전자서명 알고리즘 자체의 안전성은 검증된 전자서명 기법을 사용하게 되면 현재의 계산 능력에서는 안전하다는 사실이 대부분 증명되어 있다. 그러나 실생활에 있어서 전자서명의 가장 큰 위험요소는 알고리즘 자체의 안전성 보다는 비밀키 노출에 의해 안전성이 파괴되는 것이다. 비밀키의 노출은 사용자의 부주의나 의도적인 접근으로 쉽게 이루어 질 수 있고, 실제 생활에서 흔히 일어나고 있는 일이다. 또 이렇게 비밀키를 노출하게 되면, 안전성이 증명되어 있는 전자서명 알고리즘도 해당 비밀키에 대해서는 안전성이 모두 파괴된다. 이러한 문제를 해결하기 위한 가장 일반적인 방법이 비밀분산 기법이다 [10, 41]. 그러나 비밀분산기법은 다수의 사용자에게 비밀을 분산함으로써 고비용을 유발시킨다. 따라서 일반적인 전자서명 기법에는 적합하지 않다. 또한 스마트카드와 같이 하드웨어를 이용하는 방법은 특별한 목적으로 사용하게 되면 매우 유용하지만 비밀분산 기법과 같이 고비용을 유발하므로 일반적인 전자서명기법에서는 적합하지 않다.

비밀키 노출 시에 비밀분산기법이나 하드웨어 장치를 사용하지 않고 피해를 최소화 하는 방법으로 전방보안의 개념이 제안되었다. 전자서명기법에서는 Anderson [1]이 처음 전방보안(Forward Secure)의 개념을 사용하였다. 이후 Bellare 와 Miner[11]가 처음으로 전방보안을 보장할 수 있는 실용가능한 전자서명기법을 제안하였다. Itkis와 Reyzin [25]은 전방보안의 개념을 확장하여 침입감내 서명기법(Intrusion Resilient Signature Scheme)을 제

안하였다. 또 다른 접근 방법 중의 하나로 황정연등[22]은 $c-times$ 전자서명기법을 제안하였다. 이 기법은 서명자의 서명 능력을 c번으로 제한하는 방법을 사용한다. 이 서명기법은 c 차수의 다항식을 사용하여 비밀값에 대한 정보를 각 서명과 함께 공개함으로써 서명자가 c번 이상의 서명을 생성하게 되면 자신의 비밀값이 드러나게 되는 방법을 사용하고 있다. 이 기법은 기존 서명기법과 호환된다는 점에서 실용적이라고 할 수 있지만 사용목적이 제한되어 있다.

본 학위논문에서는 비밀키 노출에 의해 발생하는 문제점들에 주목하였다. 또한 전자서명 기법을 제안하는데 있어서 Schnorr, ElGamal 그리고 DSS와 같은 기존 서명기법과의 호환성을 고려하였다. 본 학위논문에서는 전방보안을 보장하는 키 전개방식 서명기법을 설계하고 이 서명 기법의 안전성에 대한 정량적 근거를 제시 한다. 특히 복잡도이론(Complexity Theory)에 근거한 암호학적 축소(Cryptographic Reduction) 기법을 사용하여 주어진 시스템의 안전성이 얼마나 되는지 공격자의 자원에 관한 함수로 제시한다. 또 Schnorr 서명기법을 응용한 $c-times$ 서명기법을 설계하고 효율성을 평가 하였다. 우리가 제안하는 서명기법은 Schnorr 서명기법 뿐만 아니라 Diffie-Hellman 가정을 기반으로 하는 전자서명 기법들과 호환가능하다. 또한 제안하는 서명기법의 안전성을 증명함으로써 제안하는 서명기법의 안전성이 기존의 서명기법의 안전성과 동일하거나 더 안전하다는 것을 증명한다.

References

1. R. Anderson, "Cryptography and Security Policy," Invited Talk of the *ACM-CCS'97*, 1997.

2. M Abdalla, S. Miner, and C. Namprempre, "Forward-secure threshold signature schemes" In Proc. of *CT-RSA'01*, LNCS vol. 2020, pp.441–456. 2001.

3. M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme" In Proc. of *ASIACRYPT'00*, LNCS vol. 1976, pp.116–129. 2000.

4. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing" SIAM Journal on Computing, vol. 32, no 3, pp.586–615. 2003.

5. D. Boneh and M. Franklin, "ID-based encryption from the Weil-pairing", In Proc. of *CRYPTO'01*, LNCS vol. 2139, pp. 213–229, 2001.

6. D. Boneh, "The decison Diffie-Hellman problem", In Proc. of *ANTS'98*, LNCS vol. 1423, pp. 48–63, 1998.

7. D. Boneh, H. Shacham, and B. Lynn, "Short signatures from the Weil-pairing", In Proc. of *ASIACRAPT'01*, LNCS vol. 2248, pp. 514–532, 2001.

8. I. Blake, G. Seroussi and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Prress, LNS 265, 1999.

9. V. Boyko, "On the Security Properties of the OAEP as an All-or-Nothing Transform", In Proc. of *CRYPTO'99*, LNCS 1666, pp. 503–518, 1999.

10. G. Blakley, "Safeguarding cryptographic keys", Proc. of *AFIPS'79*, 1979

11. M. Bellare and S. Miner, "A forward-secure digital signature scheme," In Proc. of *CRYPTO'99*, LNCS vol. 1666, pp. 431–448. 1999.

12. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz and A. Sahai, "Exposure-Resilient Functions and All-Or-Nothing Transforms", In Proc. of *EUROCRYPT'00*, LNCS vol. 1807, pp. 453–469, 2000.

13. R. Canetti, S. Halevi, and J. Katz, "A forward secure public key encryption scheme" In Proc. of *EUROCRYPT'03*, LNCS vol. 2656, pp.255–271. 2003.

14. J. Cha and J. Cheon, "An identity-based signature from gap Diffie-Hellman groups" In Proc. of *PKC'03*, LNCS vol. 2139, pp.18–39. 2003.

15. Y. Dodis, J. Katz, S. Xu and M. Yung , "Key-insulated public key cryptiosystems", In Proc. of *EUROCRYPT'02*, LNCS vol. 2332, pp. 65–88, 2002.

16. Y. Dodis1, A. Sahai, and A. Smith, "On Perfect and Adaptive Security in Exposure-Resilient Cryptography," Proc. of *EUROCRYPT'01*, LNCS vol. 2045, pp. 301–324, 2001.

17. Y. Desmedt and Y. Frankel, "Threshold Cryptosystems", Proc. of *CRYPTO'89*, LNCS vol. 435 ,pp. 307–315, 1989

18. C. Gentry and A. Silverberg, "Hierarchical ID-Based Cryptography", In Proc. of *ASIACRYPT'02*, LNCS vol. 2501, pp. 548–566, 2002.

19. L. Guillou and J. Quisquater , "Paradoxical identity-based signature scheme resulting from zero-knowledge", In Proc. of *CRYPTO'88*, LNCS vol. 403, pp. 216–231, 1990.

20. S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks", SIAM Journal on computing, vol. 17, no. 2, pp. 281–308, 1988.

21. A. Hertberg, M. Jackobson, S. Jarecki, H. Krawczyk and M. Yung, "Proactive public key and signature schemes", Proc. of *ACM-CCS'97*, 1997.

22. J. Hwang, D. Lee and J. Lim, "Digital signature scheme with restriction on signing capability," Proc. of *ACISP 2003*, LNCS vol. 2727, pp. 324–335, 2003

23. F. Hess, "Efficient identity based signature schemes based on pairings" In Proc. of *SAC'02*, LNCS vol. 2595, pp.310–324. 2002.

24. F. Hu, C. Wu and J. Irwin, "A new forward-secure signature scheme using bilinear maps" In IACR e-print achieves 188, 2003.

25. G. Itkis and L. Reyzin, "Forward Secure Signatures with optimal signing and verifying" In Proc. of *CRYPTO'01*, LNCS vol. 2139, pp.332–354. 2001.

26. A. Joux and K. Nguyen, "Seperating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups", In e-Print Achieve 188, 2001.

27. S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," In Proc. of *ICICS'97*, LNCS vol. 1334, pp. 223–232, 1997.

28. A. Kozlov and L. Reyzin, "Forward-secure signatures with fast key update" In Proc. of *CSCN'02*, LNCS vol. 2579, pp.247–262. 2002.

29. H. Krawczyk, "Simple forward secure signatures for any signature scheme" In Proc. of *ACM-CCS'00*, pp.108–115. 2000.

30. B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong Nnon-designated proxy signature," Proc. of *ACISP'01*, LNCS vol. 2119, pp. 474–486, 2001.

31. M. Kim, *Provably secure identification protocol based on the bilinear Diffie-Hellman problem*, Master Thesis in ICU, 2002.

32. T. Maklin, D. Micciancio and S. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods" In Proc. of *EUROCRYPT'02*, LNCS vol. 2332, pp.400–417. 2002.

33. A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

34. A. Menezes, T. Okamoto, and S. A. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", *IEEE Trans. Inform. Theory*, vol. 39, pp. 1639–1646, 1993.

35. A. Menezes, P. Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, pp. 451–460, 2002.

36. M. Mambo, K. Usuda and E. Okamoto, "Proxy signatures : delegation of the power to sign messages," IEICE Trans. Fundamentals, vol. E79-A, no. 9, 1996.

37. M. Maurer and S. Wolf, "Lower bounds on generic algorithms in groups." In Proc. of *EUROCRYPT'98*, LNCS vol. 1403,, pp. 72–84, 1998.

38. T. Okamoto and D. Pointcheval, "The gap-problem: a new class of problems for the security of cryptographic schemes", In Proc. of *PKC 2001*, LNCS vol. 1992, pp. 104–118, 2001.

39. D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures", Journal of *CRYPTOLOGY'00*,vol 13, no. 3, pp. 361–396, 2000.

40. J. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, GTM 106, 1986.

41. A. Shamir, "How to share a secret", Communications of the *ACM*, vol. 22, pp. 612-6-13, 1979.

42. A. Sadeghi and M. Steiner. "Assumptions related to discrete logarithms: Why subtleties make a real difference". In Proc. of *EUROCRYPT'01*, LNCS 2045, pp. 243–260, 2001.
- Full version of paper, available from
`http://www.semper.org/sirene/lit/abstrA1.html`.

Acknowledgements

First, I would like to express my sincere gratitude to Prof. Kwangjo Kim, my academic advisor, for his constant direction and support. He always has shown his consistent a affection and encouragement for me to carry out my research and life in ICU. Special thanks also goes to Prof. Jae Choon Cha and Dr. Dong Hoon Lee for their generosity and agreeing to serve as committee members of my thesis. I also would like to thanks to all members of cryptology and information security laboratory: Byunggon Kim, Songwon Lee, Hwasun Chang, Jaehyrk Park, Soogil Choi, Sangwon Lee, Zeen Kim , Sungjoon Min, Joongman Kim, Kyusuk Han, Seok-kyu Kang, Jeong-Kyu Yang, Yunkyung Jung, SangBae Park, Vo Duc Lim from Vietnam, Yan Xie, Wang Ping, and Xiaofeng from China, and Divyan from India, for giving me lots of interests and good advices during the course of my study. In addition, I appreciate to the graduates, Jungyeon Lee, Junbaek Ki, Jongseong Kim, Myungsun Kim, Hyunrok Lee, Hyunggi choi, Wooseok Ham, Dang Nguyen Duc for their everlasting guidance in life and study of ICU and I want to present my sincere gratitude to my fellow student Juhong Kim. It has been a precious time due to their being. Most of all, I greatly appreciate my lovely wife, Jiyoung Kim, for her belief and constant encouragement, and my father and mother for their endless concerns and devotional affection. I cannot forget their trust and encouragement on me. My father in law and mother in law have given me warmhearted concerns and my brothers and sisters also have given me cordial concerns. I hope God bless my family and to be happy. Finally, I will always remember the life of ICU. It made me a grownup person.

Curriculum Vitae

Name : Chul-Joon Choi

Date of Birth : Feb. 29. 1972

Sex : Male

Nationality : Korean

Education

1991.3–1998.2 Electronic Engineering
SungKyunKwan University (B.S.)

2002.2–2004.2 Engineering
Information and Communications University (M.S.)

Career

1998.1–2002.2 Engineer
Semiconductor R&D Division
Samsung Electronics

2002.3–2002.11 Graduate Research Assistant
ID기반 인증 모델 및 응용 서비스에 관한 연구
한국전자통신연구원

2002.9–2003.2	Graduate Research Assistant 온라인 게임 아이템의 안전한 전자거래시스템 개발 산학연과제, 샘틀로미디어
2002.2–2003.2	Graduate Research Assistant 정보보호 고급인력양성 과제 I, II, III 정보통신부
2003.3–2004.2	Graduate Research Assistant 국제 정보보호 기술연구소 운영지원사업 정보통신부
2003.3–2003.8	Graduate Research Assistant Link Layer Security 한국전자통신연구원
2002 Fall	Teaching Assistant Education Research Center for Gifted in IT
2003 Spring	Teaching Assistant Education Research Center for Gifted in IT
2003 Spring	Graduate Teaching Assistant ICE514 Concrete Mathematics
2003 Fall	Teaching Assistant Education Research Center for Gifted in IT

Publications

(1) 2002.8 C. Choi and J. Cheon, "An Overview of Bluetooth Security," Proceedings of *KIISC* Conference Region Chung-Cheong, pp. 95–111, 2002.

(2) 2003.6 Z. Kim, S. Lee, C. Choi and K. Kim "A Study on Next Generation Key Management Protocol for IPsec", Proceedings of *CISC'03*, pp. 272–275, 2003.

(3) 2003.10 C. Choi, Z. Kim and K. Kim, "Schnorr Signature Scheme with Restricted Signing Capability", Proc. of *CSS'03*, pp. 385–390, 2003. Kitakyushu, Japan, IPSJ

(4) 2003.12 C. Choi and K. Kim "Key Evolving Forward Secure Signature Scheme", Proceedings of *CISC-W'03*, pp. 428–433 , 2003.