A Thesis for the Degree of Master of Science

Intrusion-Resilient Key-Evolving Protocol under the Discrete Logarithm Problem

Joong Man Kim

School of Engineering

Information and Communications University

2004

Intrusion-Resilient Key-Evolving Protocol under the Discrete Logarithm Problem

Intrusion-Resilient Key-Evolving Protocol under the Discrete Logarithm Problem

Advisor : Professor Kwangjo Kim

by

Joong Man Kim
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

Intrusion-Resilient Key-Evolving Protocol under the Discrete Logarithm Problem

Joong Man Kim

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

M.S. Joong Man Kim
20022036 **Intrusion-Resilient Key-Evolving Protocol under the Discrete Logarithm Problem**

School of Engineering, 2004, 39p.
Major Advisor : Prof. Kwangjo Kim.
Text in English

Abstract

These days, with the advancement and propagation of the Internet and Information Technology, many security issues have emerged. One of keys enabling to deal with such issues is to adopt Cryptography. Unfortunately, cryptography will not work well if a piece of critical information (*e.g. secret key*) is not kept secret from unauthorized entities. When the secret key is revealed, all cryptographic systems will be compromised. Actually, exposing secret keys seems to be unavoidable. And we call this the key exposure problem.

Recently, the notion of *key-evolving paradigm* (*or key-evolving protocol*) was proposed as a means of mitigating the harmful effects that key exposure can cause. In this model, the whole lifetime is divided into distinct periods such that at time period j, the signer holds the secret key SK_j and updates it periodically, while the public key PK is fixed during its lifetime.

In this thesis, we investigate the key exposure problem in a key-evolving protocol. We then present the concept of intrusion-resilience, one of alternative concepts such as forward-security, key-insulated security, etc., standing against the key exposure problem. Our intrusion-resilience has the following property: If secret keys of all periods are not compromised, it is impossible to forge signatures relating to non-exposed secret keys. In the next stage, we propose a key-evolving protocol which guarantees intrusion-resilience. Our scheme is constructed from the unforgeably secure Schnorr signature scheme, one of the schemes based on *the discrete logarithm problem* (DLP). Applying for a threshold scheme is also enabling to make our scheme robust. Finally, we can show equivalence between existence of a forger and feasibility of solving the DLP under the random oracle model.

Contents

List of Figures

List of Abbreviations

DLP	Discrete Logarithm Problem
IRKE	Intrusion-Resilient Key-Evolving
IRKE-SIG	Intrusion-Resilient Key-Evolving Signature Scheme
KE-SIG	Key-Evolving Signature Scheme
SIG	Ordinary Signature Scheme
UF-CMA	Unforgeable Against Chosen Message Attack

List of Notations

Agent a kind of server executing simple computation

$\mathbb{G}_q$ a cyclic group of a prime order q

g the order of $\mathbb{G}_q$

N a security parameter

$\mathcal{O}_{sig}$ an oracle that given a message returns a signature on that message

PK a public key

$\mathcal{S}$ a secret information

T a total time period

$x \overset{\mathcal{R}}{\leftarrow} \mathcal{X}$ an element x randomly selected according to a probability space $\mathcal{X}$

$x \leftarrow y$ x is replaced by y

$\mathbb{Z}_p^*$ a group under multiplication modulo p

$\in_{\mathcal{R}}$ chosen at random

$\cdot$ a product operation

$\bullet$ an inner product operation

$\circ$ a binary operation

Chapter 1 Introduction

1.1 The Key Exposure Problem

In practice the greatest threat against the security of all cryptographic schemes such as public key cryptosystem and a digital signature scheme, etc., is exposure of the secret (encrypting or signing) key, due to compromise of the security of the underlying system or machine storing the key. The danger of successful cryptanalysis of the cryptographic 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.

In the key exposure problem, the secret key is assumed to be slowly compromised over time, so that more and more information about a secret key is eventually leaked. This models the general situation in the real world where memory, storage systems and device cannot perfectly hide all information for long time (due to physical and operational leakages). In this setting, in order to protect against exposure threats, the secret key is represented in an "exposure-resilient" form, which is periodically refreshed with the following guarantee: as long as the adversary does not learn "too much" information about the current representation of the secret between successive refreshes, the system should remain secure.

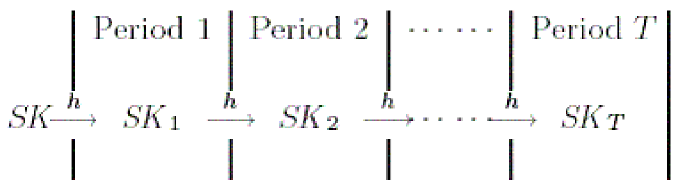
Period 1
Period 2
······
Period T
$SK \xrightarrow{h} SK_1 \xrightarrow{h} SK_2 \xrightarrow{h} \cdots\cdots \xrightarrow{h} SK_T$

Figure 1.1: Key-Evolving Protocol Paradigm

1.2 Key-Evolving Paradigm

Tzeng *et al.* [28] proposed a *key-evolving paradigm* (*or key-evolving protocol*), like the one used in forward-secure digital signature schemes. They deal with the key exposure problem of public-key encryption schemes, but signature scheme also will have similar paradigm. Let the whole lifetime be divided into periods, starting with 0. The public key PK of the signer is fixed for the whole lifetime. The signer's secret key at time period i is $SK_i, i \geq 0$. When time runs from period i to period $i+1$, the signer applies a hash function h to SK_i to get SK_{i+1} and then deletes SK_i immediately, possibly with the help of a trusted agent TA. The key-evolving paradigm is illustrated in Figure 1.1.

A signature always includes the value i of the time period during which it was produced, so that it is viewed as a pair $< i, \sigma >$, where i and σ mean the time period and the signature value respectively. The verifying algorithm takes the (fixed) public key PK, a message and candidate signature, and verifies that the signature is valid in a sense that it was produced by the legitimate user *in the period indicated in the signature.* We stress that although the user's secret key evolves with time, the public key remains fixed throughout a total time period, so that the signature verification process is unchanged, as are the public key certification and management processes.

The number of periods and the length of each period depends on your choice. For example, we want to use the scheme under a certain public key for one year, with daily updates, in which case $T = 365$ and each period has length one day.

1.3 Related Works

Key exposures appear to be inevitable. Especially, a method to prevent key exposure entirely (*e.g.*, by using tamper-resistant devices) is to be expensive and impractical for most common applications. Thus minimizing their negative impacts is extremely important. A long line of researches for dealing with this issue has been proposed.

Forward Security. In *forward-secure schemes* [1, 3], the secret key is stored by a single signer and this key is updated by the signer at the beginning of every time period. This security preserves the security of past signatures even after the secret signing key has been exposed: time is divided into predefined time periods, with the signer updating his secret at the end of each time period; the adversary is unable to forge signatures for past periods even if she learns the key for the current one. In this model, nothing can be done about the future periods: once the adversary exposes the current secret, she has the same information as the signer.

Threshold Security and Proactive Security. *Threshold schemes* [8, 26] distribute secrets among n devices so that exposure of secrets from, say, t of these devices will not allow an adversary to "break" the scheme. That is, the adversary, however, cannot generate valid signatures as long as the number of compromised devices is less than some predetermined security parameter (smaller than the number of devices needed to generate a valid signature). *Proactive schemes* [19, 14] improve upon this model by allowing multiple corruptions of *all* signers, limiting only the number of *simultaneous* corruptions. Proactive forward-secure signatures considered in [2] combine this with the advantages of forward-security.

KEY-INSULATED SECURITY. *Key-insulated scheme* [9] presented by Dodis *et al.* addresses the limitation of forward security: the adversary cannot generate signatures for the future (as well as past) time periods even after learning the current signing key. This is accomplished via the use of two modules : a (possibly mobile) *signer*, and a (generally stationary) *home base*. The signer has the secret signing key, and can generate signatures on its own. At the end of each time period, the signing key expires and the signer needs to *update* his keys by communicating with the home base and performing some local computations (the communication with the base is, in fact, limited to a single message from the base to the signer). Thus, although the signer's keys are vulnerable (because they are frequently accessed, and, moreover, because the signer may be mobile), key exposure is less valuable to the adversary, as it reveals only short-term keys. Perhaps the most compelling application of such a model is the example of a frequently traveling user, whose laptop (or hand-held) is the signer, and office computer is the home base. This model enables security that is not possible in ordinary or even forward-secure schemes: even if the signing key is compromised (for up to k time periods, for predetermined security parameter k), the adversary will be unable to forge signature for *any* other time periods. (Notice that in forward-secure schemes model, signatures for any time period following a compromise are *necessarily* forgeable.)

INTRUSION-RESILIENCE. Recently presented *intrusion-resilient scheme* [15] combines some benefits from the key-insulated [9], forward-secure [1, 3], and proactive [19, 14] security notions. A detailed discussion of relationships among all these notions is included in [15], and we therefore omit it here.

Intuitively, intrusion-resilient model divides into *time periods*, and augments each signature with a "time-stamp" (the time period number, during which the signature was generated). Verification is essentially similar to the ordinary signature schemes, except it includes the time period as input — if the time period is changed, the signature becomes invalid. These extensions to ordinary signatures are the same as for forward-secure [1, 3] and key-insulated models [9].

In addition, the ordinary signer is replaced in intrusion-resilient model — similarly to the key-insulated model of [9] — with two modules: *signer* and *home base* (thus, the name of the model: Signer-Base Intrusion-Resilient, or SiBIR for short). The signer, using its secret key, can generate signatures, but only for the current time period (unlike forward-secure signer, which can also generate signatures for all the future periods). At the end of the current period, the signer must *update* its secret key for the next period — this requires an *update message* from the base.

Thus, intrusion-resilient scheme, as the key-insulated one [9], preserves security of both past and future time periods, when the signers compromised. Moreover — unlike in the key-insulated model — for intrusion-resilient schemes this security is preserved even if both signer and base are compromised, as long as the compromises are not simultaneous. In the case of simultaneous compromise, the security of past (but not the future) periods is preserved — this is the best that can be achieved.

To achieve such security, SiBIR model, much as proactive schemes [19, 14], provides a *refresh* mechanism: base changes its key and sends a refresh message to the signer, which uses it to refresh its key as well. This can be done at arbitrary times and as many times as desired (unlike update, which is done only at the pre-arranged times, and typically has a limit on the total number of updates); refresh is transparent to the verifier (in contrast, update changes the time period number and is therefore visible to the verifier). Refresh prevents the attacker from learning anything by breaking into the base. Specifically, learning the base secrets at arbitrary times by itself does not help attacker in any way. Only if the attacker compromises both signer and base simultaneously, dose she learn all the system's secrets and can therefore generate — just as the system itself — valid signatures for all the future periods. But if the attacker misses at least one refresh message between the compromising of the base and signer secrets, then compromising the base does not add anything useful to her knowledge. Thus, SiBIR model minimizes the impact of the key compromises, which explains the *Intrusion-Resilient* part of its name.

We note that each of these models may be appropriate in different environments. Forward-secure schemes are advantageous in that the user is self-sufficient and need not interact with any other device. On the other hand, the security provided by key-insulated and intrusion-resilient schemes is better and these schemes might therefore be used when interacting with a server is feasible and does not represent a serious drawback. Finally, although the intrusion-resilient model offers stronger security guarantees than the key-insulated model, we note that solutions for the latter are (thus far) much more efficient. The choice of which type of scheme to use therefore depends heavily on an assumption about the (physical) security of the server.

1.4 Our Contribution

In this thesis, we define the different notion of intrusion-resilience from one which Itkis *et al.* [15] have proposed. Our intrusion-resilience will be derived from the resilience which Tzeng *et al.* [28] have defined and will be achieved using the properties of a linear system of equations and the threshold cryptography [6]. Also, we study key-evolving protocol under the discrete logarithm problem and present an intrusion-resilient key-evolving protocol applicable for schemes based on the discrete logarithm problem. Our scheme exhibits an efficient key update algorithm and introduces no more significant overhead than the underlying scheme.

1.5 Outline of the thesis

The thesis is organized as follows: We review background knowledge related to our scheme in Chapter 2. We then give formal models and definitions of various terminologies in Chapter 3. The description of our intrusion-resilient key-evolving protocol under the discrete logarithm problem called a “IRKE” is made in Chapter 4. In Chapter 5, we analyze correctness, efficiency and security proofs of our scheme. Finally, we will make conclusions in Chapter 6.

Chapter 2
Preliminaries

2.1 Random Oracle Model

In many signature schemes, a cryptographic hash function, such as MD5 [23] or SHA-1 [27], is used, namely to reduce the size of the message. Such a cryptographic hash function has the property that it is collision-resistant, and therefore one-way.

Many recent proofs [5, 20, 22] make the assumption that this cryptographic hash function is an *ideal random function* also known as *random oracle*: for any new query, the answer is uniformly distributed in the output set, independently of previous query/answer pairs. This is the so-called *random oracle model* [4].

Moreover, in this model, a simulator is allowed to set the output of the random oracle to specific values (uniformly distributed) for an input that had not yet been defined. The random oracle model assumes that hash functions used in a scheme are "truly random" hash functions. Although the security under the random oracle model is not rigid, it does provide satisfactory security argument to related schemes in most cases [6]. Also, since proofs in the random oracle model are just security arguments, but not the strongest proof of security that one could require, we try to minimize the use of random oracles.

2.2 Schnorr Type Signature Schemes

ElGamal [11] was the first to propose a signature scheme based on the discrete logarithm problem. Then, Schnorr [24, 25] improved the scheme using the modulo q truncating function, playing in a prime subgroup. This latter scheme has been formally proven unforgeable in the random oracle model relative to the discrete logarithm problem [20, 22] and also has become an essential tool for smart card. Thus, we will select this Schnorr signature scheme as one example among discrete logarithm based schemes for constructing our scheme. As other schemes, many variants have been defined and standardized by governments: the US-standard DSA [10] and the Korean-standard KCDSA [16].

2.3 Threshold Cryptosystem

The concept of a threshold scheme was first introduced by Shamir [26]. A threshold cryptographic protocol involves a set of players together, who each possesses a secret share, to accomplish a cryptographic task via exchange of messages among them. Threshold cryptographic protocols provide strong security assurance and robustness against a number of malicious attackers under a threshold. For example, in the Shamir's (k, n) threshold scheme, a secret s is divided into n pieces $s_1, s_2, \ldots, s_n$ such that:

1. Knowledge of any k or more s_i pieces makes s easily computable;

2. Knowledge of any $k - 1$ or fewer s_i pieces leaves s uncomputable.

As mentioned above, the threshold scheme enables possession of secret key to be distributed in public key cryptosystem. Consequently, only k parties or more can decrypt a ciphertext encrypted with the corresponding public key or produce a digital signature on a message. With fewer k parties, the work cannot be done. Shamir's (k, n) threshold scheme [26], one sort of threshold cryptosystems, will be adopted in updating signing secret keys in our scheme.

2.4 Number Theoretic Hard Problem

Cryptographic schemes work by trying to relate breaking its security goal to solving one or more mathematical hard problems. These mathematical problems should be intractable such that any attempt to solve problems becomes impractical in terms of time and computational resources. One such famous mathematical hard problem used in cryptography is the *Discrete Logarithm Problem* (DLP). It is defined as follows:

Definition 2.4.1 *Given an Abelian group $\mathbb{G}$ equipped with a binary operation $\circ$ and an element $h \in \mathbb{G}$. Find an integer x and $g \in \mathbb{G}$ satisfying $g^x = h$, where $g^x = g \circ g \circ \cdots \circ g$ (x times).*

Based on this problem, our key-evolving protocol will be constructed. And we describe it making use of the threshold scheme and Schnorr signature scheme described before. Therefore, our security proof will be based on the assumption that the DLP is intractable. We call it the DLP assumption.

Chapter 3
Models and Definitions

Here we provide various models and formal definitions for a key-evolving protocol. We assume that there is an *Agent* who holds some secret share for updating secret key of the signer.

3.1 System Models

In this section we introduce our communication model and types of adversaries.

3.1.1 Communication Model

In this thesis, we will use the term *Agent* instead of *player* like in [12]. An *Agent* will considered as a kind of server executing simple computation. These *Agents* will help the signer completing Shamir's (k, n) threshold scheme [26]. The participants in our scheme includes a set of j *Agent*s who are connected by a broadcast channel. Additionally, they are capable of private point-to-point communication over secure channels. (Such channels might be implemented on the broadcast channel using cryptographic techniques.) Furthermore, we assume that the signer is trusted during the setup phase and that the *Agent*s are capable of both broadcast and point-to-point communication with him. Finally, we work in a synchronous communication model; that is, all participating *Agent*s including the signer have a common concept of time and, thus, can send their messages simultaneously in a particular round of a protocol.

3.1.2 Types of Adversaries

We assume that any adversary attacking our scheme can listen to all broadcasted information and may compromise the shares which *Agent*s hold in some way to learn their secret information. However, the adversary might work in a variety of contexts. We categorize the different types of adversaries here. In both categories described below, the last option listed describes the most powerful adversary, since it always encompasses the preceding options in that category.

The first category we consider is the power of an adversary can have more than a compromised *Agent*. We list the options, as outlined in [12]. First, an adversary may be *eavesdropping*, meaning that she may learn the secret information of an *Agent* but may not affect his behavior in any way. A more powerful adversary is one that not only can eavesdrop but can also stop the *Agent* from participating in the protocol. We refer to such an adversary as a *halting* adversary. Finally, the most powerful notion in this category is a *malicious* adversary, who may cause an *Agent* to deviate from the protocol in an unrestricted fashion.

The second category which defines an adversarial model describes the manner in which an adversary selects the set of *Agent*s to compromise. The first type is a *static* adversary, who decides before the protocol begins which set of *Agent*s to compromise. An *adaptive* adversary, on the other hand, may decide "on the fly" which *Agent* to corrupt based on knowledge gained during the run of the protocol. Finally, a *mobile* adversary is traditional one which is not only adaptive, but also may decide to control different sets of shares which *Agent*s including the signer hold during different time periods. In this case, there may be no share which has not been compromised throughout the run of the protocol, but the adversary is limited to controlling some maximum number of shares at any one time.

3.2 Functional Definitions

We first define a signature scheme and then a key-evolving signature scheme.

Definition 3.2.1 *A signature scheme is a triple of probabilistic polynomial-time algorithms,* SIG = (*Gen, Sign, Ver*):

1. *Gen, the key generation algorithm,* takes as input a random string and outputs a pair of keys (X, Y), where X is the private signature key, and Y is the public verification key.

2. *Sign, the signing algorithm,* takes as input a message M and the private signature key X, and produces a signature *Sig*.

3. *Ver, the verifying algorithm,* takes as input a message M, a signature *Sig*, the public verification key Y, and checks whether *Sig* is a valid signature of M.

This definition can be extended in a natural way to capture a key-evolving protocol of signature schemes as follows:

Definition 3.2.2 *A key-evolving signature scheme is a quadruple of probabilistic polynomial-time algorithms,* KE-SIG = (*Gen, Upd, Sign, Ver*):

1. *Gen, the key generation algorithm,* takes as input a security parameter N, the total number of time periods T over which the scheme will operate, and possibly other parameters, to return a *base public key* PK and corresponding *base secret key* SK_0. The algorithm is probabilistic.

2. *Upd, the key update algorithm,* takes as input the signing secret key SK_i of the current time period to return the signing secret key SK_{i+1} of the next time period. The algorithm is usually deterministic.

3. *Sign, the signing algorithm,* takes the signing secret key SK_i of the current time period and a message M to return a *signature* (Sig, i) of M for the current time period i. The algorithm may be probabilistic. The signature is always a pair consisting of the value i of the current time period and a tag *Sig*.

4. *Ver, the verifying algorithm,* takes the public key PK, message M and candidate signature (Sig, i) to return a bit, with 1 meaning *valid* and 0 meaning *invalid*. The algorithm is typically deterministic.

We say that (Sig, i) is a *valid* signature of M for time period i if the verifying algorithm returns 1. It is required that a signature of M generated via the signing algorithm be a valid signature of M for time period i. We assume that the secret key SK_i for time period $i \in \{1, \ldots, T\}$ always contains the value i itself and also always contains the value T of the total number of time periods.

We may assume a single *Agent* for simplicity in constructing our scheme. In practice, we distribute trust to multiple *Agent*s such that each $Agent_j$ holds a share s_j of the system secret s. The signer with secret key SK_i and the *Agent*s together can compute SK_{i+1} in a secure way through Shamir's (k, n) threshold scheme [26]. Here, *Agent*s do not need to be trusted but only simple computation execution is required. We discuss this in detail in Chapter 4.

3.3 Security Requirements

We will say about a key-evolving protocol, which means all processes except for the signing and the verifying algorithms in the above scheme. Some desirable properties for a key-evolving protocol are presented as follows [18]:

Definition 3.3.1 *A key-evolving protocol is forward-secret if the compromise of SK_i will not compromise SK_j for all $j < i$.*

Definition 3.3.2 *A key-evolving protocol is backward-secret if the compromise of SK_i will not compromise SK_j for all $j > i$.*

Definition 3.3.3 *A key-evolving protocol is key-independent if it is forward-secret and backward-secret.*

We will propose a key-evolving protocol satisfying key-independence as well as intrusion-resilience described in the next section.

3.4 Intrusion-Resilience

Tzeng *et al.* [28] introduced the concept of resilience for public-key encryption scheme. It will be similar for the signature scheme as follows.

Definition 3.4.1 (Resilience) *Assume a security model for signature scheme. A key evolving signature scheme is z-resilient if the attacker cannot break the signature scheme under the assumed security model even if he gets z secret keys $SK_{i_1}, SK_{i_2}, ..., SK_{i_z}$.*

Even if the attacker gets z secret keys $SK_{i_1}, SK_{i_2}, ..., SK_{i_z}$ of z time periods, he cannot get another secret key SK_i, for $i \neq i_l, 1 \leq l \leq z$. Actually, our scheme becomes to be $(T-1)$-resilient scheme, where T represents the total number of time periods. Therefore, we present a main definition about the concept of resilience.

Definition 3.4.2 (Intrusion-Resilience) *A key-evolving signature scheme is intrusion-resilient if it is $(T-1)$-resilient.*

3.4.1 Comparison with the previous concept

We can see that Itkis *et al.*'s intrusion-resilience [15] and our intrusion-resilience are considerably different concepts. The basic difference between two concepts is that in Itkis *et al.*'s concept, even if there are only two entities — *signer* and *home base*— home base must do heavy computation, which requires the same computing power as the signer. On the other hand, in our concept, even if there are several entities — *Agent*s — as well as the signer, all *Agent*s are required to perform much simpler computation than the signer. Therefore, we can say that the ability for home base in Itkis *et al.*'s scheme is distributed to several *Agent*s performing simple computation in our scheme. In other word, we can alleviate the danger in which home base will be compromised, by means of several *Agent*s' practice.

We also believe that our intrusion-resilience offers much stronger security than key-independence, which will be proved later through the security analysis. In the next chapter, we will propose a key-evolving protocol which guarantees an intrusion-resilience.

Chapter 4
Proposed Scheme

Our key-evolving protocol (denoted IRKE) using the Shamir's (k, n) threshold scheme [26] to share a secret information used in updating secret key, where several polynomials are employed, is presented as below. Also, our key-evolving signature scheme (denoted IRKE-SIG) consists of a key-evolving protocol (containing key generation algorithm and key update algorithm) and an application to a signature scheme based on the discrete logarithm problem such as the Schnorr signature scheme. Thus, we assume that two hash functions $h_1 : \{0,1\}^* \rightarrow \{0,1\}^T$ and $h_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ are required. h_1 is used in defining a vector function B which is shown in IRKE.*Gen*. And h_2 is used in IRKE.*Sign* and IRKE.*Ver* as in the Schnorr signature scheme.

4.1 Intrusion-Resilient Key-Evolving Protocol

Our key-evolving protocol consists of two algorithms: key generation algorithm and key update algorithm.

4.1.1 Key Generation Algorithm

We first generate a N-bit *safe* prime $p : p = 2q + 1$ such that q is odd prime (such q, satisfying $2q + 1$ is prime, is known as *Sophie Germain prime*[7]). Then the signer actually selects the secret information $\mathcal{S} = (s_1, s_2, ..., s_T)$ at random, and computes the public key $PK = (g^{s_1}, g^{s_2}, ..., g^{s_T})$ using $\mathcal{S}$.

The signer randomly selects T polynomials used in sharing each element

```
algorithm IRKE.Gen$(1^N, T)$
Parameter :
    Generate $N$-bit prime $p \leftarrow 2q+1$ with that $q$ is a prime of at least
    160-bit long, $i.e.$ $q > 2^{160}$.
    Let $\mathbb{G}_q$ denote the subgroup of the quadratic residues modulo $p$
    and $g$ the generator of $\mathbb{G}_q$.
Setting :
    $\mathcal{S} \leftarrow (s_1, s_2, ..., s_T)$ in $\mathbb{G}_q$ (secret information)
    $PK \leftarrow (g^{s_1}, g^{s_2}, ..., g^{s_T})$ in $\mathbb{Z}_p^*$
    Randomly select $T$ $(T-1)^{th}$ degree polynomials
        $f_1(x) \equiv s_1 + \sum_{i=1}^{T-1} \alpha_{1,i} x^i \pmod q$
        $f_2(x) \equiv s_2 + \sum_{i=1}^{T-1} \alpha_{2,i} x^i \pmod q$
                    $\vdots$
        $f_T(x) \equiv s_T + \sum_{i=1}^{T-1} \alpha_{T,i} x^i \pmod q$
        ,where each $f_l$ is used to share $s_l$, $1 \le l \le T$, in $\mathcal{S}$.
Distributing shares :
    Let the signer and $Agent$s hold multiple share $(f_1(x_l), f_2(x_l), ..., f_T(x_l))$,
    for some random $x_l \in_{\mathcal{R}} \mathbb{Z}_q, 1 \le l \le n$.
Define :
    $B(x)$ := the binary representation of the hash function $h_1(x)$
    $i.e.$ $B(x) := (e_1, e_2, ..., e_T) \rightarrow T$ vector
    ,where each $e_i$, $1 \le i \le T$, is bit (0 or 1) and $h_1(x)$ is a hash
    function with $T$-bit output.
Return $(PK)$
```

Figure 4.1: Key Generation algorithm

s_l, $1 \leq l \leq T$, of secret information $\mathcal{S}$. Note that after sharing, the signer discards the secret information $\mathcal{S}$ and such T polynomials. The detailed method how the signer makes and distributes shares will be described at key update algorithm. Figure 4.1 describes key generation algorithm.

4.1.2 Key Update Algorithm

Key generation is immediately followed by key update. The signer first divides each element s_l of secret information $\mathcal{S}$ into n shares $f_l(x_1)$, $f_l(x_2), ..., f_l(x_n)$, $1 \leq l \leq T$, where each x_i, $1 \leq i \leq n$, is distinct and large enough so that the maximum time period never reaches them, and then makes n multiple shares as follows:

$$\begin{gathered}(f_1(x_1), f_2(x_1), ..., f_T(x_1))\\(f_1(x_2), f_2(x_2), ..., f_T(x_2))\\ \vdots \\(f_1(x_n), f_2(x_n), ..., f_T(x_n))\end{gathered}$$

Assume that there are j *Agent*s, $j < k$, *i.e.*, $Agent_1$, $Agent_2$,..., $Agent_j$, and each pair of the signer and *Agent*s share a private channel by which secret information can be passed between them. Thus, the signer gives j multiple shares $(f_1(x_{m_1}), f_2(x_{m_1}), ..., f_T(x_{m_1}))$, $1 \leq m_1 \leq j$, to all j *Agent*s individually and stores the remaining multiple shares $(f_1(x_{m_2}), f_2(x_{m_2}), ..., f_T(x_{m_2}))$, $j+1 \leq m_2 \leq n$. At this time, the following two inequalities must be satisfied:

$$n - j < k \quad \text{and} \quad n < 2k - 2$$

This means that only the signer as well as only *Agent*s cannot update secret key, *i.e.*, the signer must collude with some *Agent*s. At time period i, the signer holds SK_i. The signer and *Agent*s would like to compute SK_{i+1}, which shall be known to the signer only.

Figure 4.2 describes key update algorithm where the reason why we use inner product ($\bullet$) is that the signer and *Agent*s select some elements randomly from all elements of their multiple shares and execute the modular summation of that elements.

```
algorithm IRKE.Upd

Assume that $j$ $Agent$s $(j < k)$ $(k$ : threshold value$)$,
$i.e.$, $Agent_1, Agent_2, \dots, Agent_j$.

1. $Agent_{r_1} (1 \leq r_1 \leq j)$ computes
   $SKU_{r_1} \leftarrow (f_1(x_{r_1}), f_2(x_{r_1}), ..., f_T(x_{r_1})) \bullet B(i+1) \bmod q$
,where $\bullet$ means inner product here and is used throughout this paper,
and then sends each $SKU_{r_1}$ to the signer.

2. The signer selects $k - j$ multiple shares randomly and computes,
$(d_1 \leq r_2 \leq d_{k-j})$,
   $SKU_{r_2} \leftarrow (f_1(x_{r_2}), f_2(x_{r_2}), ..., f_T(x_{r_2})) \bullet B(i+1) \bmod q$

3. Finally, the signer computes
   $SK_{i+1} \leftarrow \sum_{r_1=1}^{j} SKU_{r_1} \cdot (\prod_{t_1 \leq I \neq r_1 \leq t_k} \frac{x_I}{x_I - x_{r_1}}) \;+$
                 $\sum_{r_2=d_1}^{d_{k-j}} SKU_{r_2} \cdot (\prod_{t_1 \leq I \neq r_2 \leq t_k} \frac{x_I}{x_I - x_{r_2}}) \bmod q$

Return $(SK_{i+1})$
```

Figure 4.2: Key Update algorithm

Each result $(SKU_{r_1}, 1 \leq r_1 \leq j)$ of that modular summation is sent to the signer. Finally he calculates the secret key of the next time period by the Lagrange interpolation method.

We can make the computation verifiable by letting each $Agent_l$ publish $g^{f_1(x_l)}, g^{f_2(x_l)}, \dots, g^{f_T(x_l)}$. The signer then verifies whether he receives the right share from $Agent_l, 1 \leq l \leq j$, by checking

$$g^{SKU_l} \equiv g^{(f_1(x_l), f_2(x_l), \dots, f_T(x_l)) \bullet B(i)} \pmod p$$

,where $g^{(f_1(x_l), f_2(x_l), \dots, f_T(x_l)) \bullet B(i)}$ means the random multiplication of each $Agent_l$'s published values based on the hash value of time period i.

4.2 Application to Schnorr Signature Scheme

Our key-evolving protocol is applied to the Schnorr signature scheme [24, 25] which we review here. Let p and q be primes such that $p = 2q + 1$ and let $\mathcal{G}$ be the subgroup of $\mathbb{Z}_q^*$ of order q. Fix generator $g \in \mathcal{G}$. A public key is generated by choosing a secret $s \in_{\mathcal{R}} \mathbb{Z}_q$ and setting $I = g^s$. To sign message M, a user chooses random $k \in_{\mathcal{R}} \mathbb{Z}_q$ and computes $r = g^k$. Using a hash function H (modeled as a random oracle), the user then computes $e = H(M, r)$, where e is interpreted as an element of $\mathbb{Z}_q$. The signature is: $(e, z = k + xe, M)$. A signature (e, z, M) on message M is verified by computing $r \stackrel{?}{=} g^z I^{-e} \mod p$. Our constructions based on the Schnorr signature scheme are illustrated in Figures 4.3 and 4.4. We stress that the the scheme achieves *strong* security without additional modifications, yet the time required for signing and verifying is essentially the same as in the basic Schnorr scheme. We describe it in detail in the following subsections.

4.2.1 Signature Generation Algorithm

The signing procedure is straightforward, as defined by the Schnorr signature scheme [24, 25], shown in Figure 4.3.

```
algorithm IRKE.Sign$(M, SK_i)$

$k \overset{R}{\leftarrow} \mathbb{Z}_q^*$
$r \leftarrow g^k \bmod p$
$e \leftarrow h_2(M, r)$
$z \leftarrow (SK_i) \cdot e + k \bmod q$

Return $(z, e, i)$
```

Figure 4.3: Signing algorithm

4.2.2 Signature Verification Algorithm

Our verification algorithm, described in Figure 4.4, is also exactly the same as in the Schnorr signature scheme [24, 25], except it includes the time period as input — if the time period is changed, the signature becomes invalid.

In Figure 4.4, we also have a pre-computation process, *i.e.*, $g^{(s_1,s_2,\ldots,s_T)\bullet B(i)}$. $g^{(s_1,s_2,\ldots,s_T)\bullet B(i)}$ means the random multiplication of PK's elements based on the hash value of time period i ,*i.e.*, the multiplication of elements with weight 1 among all elements of PK based on the hash value of the time period i.

```
algorithm IRKE.Ver$(M, PK, (z, e, i))$

Let $PK = (g^{s_1}, g^{s_2}, ..., g^{s_T})$
$v \leftarrow g^z \cdot (g^{(s_1,s_2,\ldots,s_T)\bullet B(i)})^{-e} \bmod p$
$e' \leftarrow h_2(M, v)$
If $e = e'$, then return 1 else 0
```

Figure 4.4: Verifying algorithm

Chapter 5 Security Analysis

We now discuss the correctness, complexity and security proofs of our proposed scheme. Afterwards, T means the total time period.

5.1 Correctness

Theorem 5.1.1 *Let* IRKE.*Upd take output* SK_{i+1} *for* $1 \leq i < T$. *Then,* $SK_{i+1} \equiv (s_1, s_2, \ldots, s_T) \bullet B(i+1) \pmod{q}$.

Proof: By the Lagrange interpolation method, the following equations are satisfied.

$$\begin{aligned}
SK_{i+1} &\equiv \sum_{r_1=1}^{j} SKU_{r_1} \cdot \left(\prod_{t_1 \leq I \neq r_1 \leq t_k} \frac{x_I}{x_I - x_{r_1}} \right) + \sum_{r_2=d_1}^{d_{k-j}} SKU_{r_2} \cdot \left(\prod_{t_1 \leq I \neq r_2 \leq t_k} \frac{x_I}{x_I - x_{r_2}} \right) \pmod{q} \\
&\equiv \left[\sum_{r_1=1}^{j} \{ (f_1(x_{r_1}), f_2(x_{r_1}), ..., f_T(x_{r_1})) \bullet B(i+1) \} \cdot \left(\prod_{t_1 \leq I \neq r_1 \leq t_k} \frac{x_I}{x_I - x_{r_1}} \right) \right] + \\
&\quad \left[\sum_{r_2=d_1}^{d_{k-j}} \{ (f_1(x_{r_2}), f_2(x_{r_2}), ..., f_T(x_{r_2})) \bullet B(i+1) \} \cdot \left(\prod_{t_1 \leq I \neq r_2 \leq t_k} \frac{x_I}{x_I - x_{r_2}} \right) \right] \pmod{q} \\
&\equiv \left[\sum_{r=t_1}^{t_k} \{ f_1(x_r), f_2(x_r), ..., f_T(x_r) \} \bullet B(i+1) \right] \cdot \left(\prod_{t_1 \leq I \neq r \leq t_k} \frac{x_I}{x_I - x_r} \right) \pmod{q} \\
&\equiv \left(\sum_{r=t_1}^{t_k} f_1(x_r) \cdot \left(\prod_{t_1 \leq I \neq r \leq t_k} \frac{x_I}{x_I - x_r} \right), \sum_{r=t_1}^{t_k} f_2(x_r) \cdot \left(\prod_{t_1 \leq I \neq r \leq t_k} \frac{x_I}{x_I - x_r} \right), \right. \\
&\quad \left. \ldots, \sum_{r=t_1}^{t_k} f_T(x_r) \cdot \left(\prod_{t_1 \leq I \neq r \leq t_k} \frac{x_I}{x_I - x_r} \right) \right) \bullet B(i+1) \pmod{q} \\
&\equiv (s_1, s_2, ..., s_T) \bullet B(i+1) \pmod{q}
\end{aligned}$$

(Notation : $(t_1, t_2, \ldots, t_k) = (1, .., j, d_1, ..., d_{k-j})$)

as desired. ■

We now check that the verification is performed correctly.

Theorem 5.1.2 *Let $PK = (g^{s_1}, g^{s_2}, ..., g^{s_T})$ be a public key generated by the above key generation algorithm. Let $(M, (z, e, i))$ be an output of* IRKE.*Sign* (M, SK_i). *Then* IRKE.*Ver* $(M, PK, (z, e, i)) = 1$.

Proof: We will show that $v \equiv r(= g^k) \bmod q$. The process is the same as one of the Schnorr signature scheme.

$$\begin{aligned} v &\equiv g^z \cdot (g^{(s_1, s_2, \ldots, s_T) \bullet B(i)})^{-e} \pmod p \\ &\equiv g^{(SK_i) \cdot e + k} \cdot (g^{SK_i})^{-e} \pmod p \\ &\equiv g^{(SK_i) \cdot e + k} \cdot g^{-(SK_i) \cdot e} \pmod p \\ &\equiv g^k \pmod p \end{aligned}$$

Hence $h_2(M, v) = h_2(M, r)$, *i.e.*, $e = e'$, always holds. This means that the verification succeeds. ■

5.2 Complexity and Efficiency

Our scheme has public key and secret information of size $O(T)$, where T denotes the total time period.

Key Generation. In order to complete the key generation algorithm, we first have to generate secret information $\mathcal{S}$ and calculate the public key PK using it. It requires T modular exponentiations.

Key Update. Key update algorithm consists of addition and multiplication. Since we use Shamir's (k, n) threshold scheme, key update algorithm requires $(k \cdot \frac{T}{2})$ modular additions and k modular multiplications. $\frac{T}{2}$ means that ideal hash function (h_1) generates $\frac{T}{2}$'s '1' bits on the average.

Signing and Verifying. Our scheme has the same signature and verification algorithm as Schnorr one. So it has same efficiency and complexity except for another pre-computation process in our verification algorithm, which takes only $\frac{T}{2}$ modular multiplications. Also $\frac{T}{2}$ has same meaning as above.

5.3 Security Proof

5.3.1 Robustness

The robustness of our intrusion-resilient key-evolving protocol is shown by the following theorem:

Theorem 5.3.1 *The intrusion-resilient key-evolving protocol* (IRKE) *is robust for an adversary who can corrupt $k-1$ multiple shares among n multiple shares such that $n < 2k-2$.*

Proof: There are n multiple shares in IRKE.*Upd*. Every $Agent_l (1 \le l \le j)$ contains a multiple share MS_l. Also the signer has $n-j$ multiple shares $MS_{l'} (j+1 \le l' \le n)$. Thus, even if there exists an adversary who can corrupt up to $k-1$ multiple shares among n multiple shares, any subset of k multiple shares constructs the unique secret key SK_i uniformly distributed in $\mathbb{G}_q$ at the time period i. That means IRKE.*Upd* completes successfully in event that at most $k-1$ multiple shares are corrupted.

In this key update algorithm (IRKE.*Upd*), each partial secret key SKU_l from every $Agent_l (1 \le l \le j)$ is verified by using the random multiplication of $Agent_l$'s correspondent published values $g^{f_1(x_l)}, g^{f_2(x_l)}, \dots, g^{f_T(x_l)}$ based on the hash value of the time period i as follows:

$$g^{SKU_l} \equiv g^{(f_1(x_l), f_2(x_l), ..., f_T(x_l)) \bullet B(i)} \pmod p$$

Even at most $k-1$ multiple shares can be corrupted, the adversary needs partial secret keys from other multiple shares to form k valid partial secret keys. With k valid partial secret keys, the secret SK_i can be produced by using the Lagrange interpolation, and its correctness was shown in Section 5.1. Therefore, IRKE.*Upd* algorithm completes successfully. These show that IRKE is robust. ■

5.3.2 Key-Independence

Theorem 5.3.2 *In the random oracle model, our key-evolving protocol is key-independent if the Discrete Logarithm Problem is hard.*

Proof: We will show how to solve the Discrete Logarithm Problem (DLP) if successful attacker breaks the key-independent property. First, we will build a DL oracle using the successful attacker. Next, we'll try to solve DLP using this DL oracle.

Given y in the maximal cyclic group of $\mathbb{Z}_p^*$, we will use the successful attacker to compute $x = \log_g y$. First, we will generate a set of random numbers $\{r_i | i = 1, 2, ..., T+1\}$. And we control the random oracle to output $\{r_1, r_2, ..., r_{k-1}\}$ at the first $k-1$ queries. Thus, the following relations hold.

$$\begin{aligned} r_1 &= \mathcal{S} \bullet B(1) = (s_1, s_2, ..., s_T) \bullet B(1), \quad & Q_1 &= g^{r_1} \\ r_2 &= \mathcal{S} \bullet B(2) = (s_1, s_2, ..., s_T) \bullet B(2), \quad & Q_2 &= g^{r_2} \\ &\vdots & &\vdots \\ r_{k-1} &= \mathcal{S} \bullet B(k-1) = (s_1, s_2, ..., s_T) \bullet B(k-1), \quad & Q_{k-1} &= g^{r_{k-1}} \end{aligned}$$

At the beginning of the k periods, we force the random oracle to output $y \cdot g^a$, *i.e.*

$$y \cdot g^a = Q_k = g^{r_k} = g^{\mathcal{S} \bullet B(k)} = g^{SK_k}$$

For periods from $k+1$ to $T+1$, the random oracle proceeds as usual.

Second, we give (Q_i, SK_i) and the above public key set to the attacker A. If he succeeds in breaking key-independent, A would return (Q_j, SK_j) for some time period $j \neq i$.

The probability that $j = k$ is $\frac{1}{T}$. If this happens, we have $SK_j = SK_k = \log_g(y \cdot g^a) = \log_g y + a$. Therefore, the discrete logarithm of y can be computed as follows :

$$x = SK_j - a$$

Given this DL oracle, we can solve DLP. Therefore, if solving DLP is hard, this key-evolving protocol is key-independent in the random oracle model. ■

5.3.3 Intrusion-Resilience

Theorem 5.3.3 *In the random oracle model, our key-evolving protocol is intrusion-resilient protocol.*

Proof: Attacker can consider each element of secret information $\mathcal{S}$ as an unknown variable and each secret key as a linear equation. Since the number of all elements in secret information $\mathcal{S}$ is T, which represents the total time period, we can consider our key-evolving protocol as a T linear system of equations with T unknown variables. Thus, attacker must know T secret keys to compromise a secret key of non-exposed (past or future) time period by the property of a linear system of equations, *i.e.*, he must know secret keys of all periods. Therefore, if secret keys of all time periods are not compromised, it is not possible to forge signatures relating to non-exposed secret keys. This means that our key-evolving protocol is $(T-1)$-resilient, *i.e.*, intrusion-resilient. ■

Also, we prove that intrusion-resilient property can reduce to key-independent property in the following theorem.

Theorem 5.3.4 *In the random oracle model, if our key-evolving protocol is intrusion-resilient, our key-evolving protocol is key-independent.*

Proof: A proof of this theorem will be nearly the same as one of theorem 5.3.2. Let *A-KI* be an attacker against key-independent property. We build an attacker *A-IR* against intrusion-resilient property as follows. *A-IR* now runs the attacker *A-KI* against key-independent property. First, we will generate a set of random numbers $\{r_i | i = 1, 2, ..., T+1\}$. And we control the random oracle to output $\{r_1, r_2, ..., r_{k-1}\}$ at the first $k-1$ queries. Thus, the following relations hold.

$$\begin{array}{rl} r_1 = \mathcal{S} \bullet B(1) = (s_1, s_2, ..., s_T) \bullet B(1), & Q_1 = g^{r_1} \\ r_2 = \mathcal{S} \bullet B(2) = (s_1, s_2, ..., s_T) \bullet B(2), & Q_2 = g^{r_2} \\ \vdots & \vdots \\ r_{k-1} = \mathcal{S} \bullet B(k-1) = (s_1, s_2, ..., s_T) \bullet B(k-1), & Q_{k-1} = g^{r_{k-1}} \end{array}$$

At the beginning of the k periods, we force the random oracle to output SK_k, *i.e.*

$$SK_k = \mathcal{S} \bullet B(k) = (s_1, s_2, ..., s_T) \bullet B(k)$$

For periods from $k+1$ to $T+1$, the random oracle proceeds as usual.

Second, we give (Q_i, SK_i) and the above public key set to the attacker *A-KI*. If he succeeds in breaking key-independent, *A-KI* would return (Q_j, SK_j) for some time period $j \neq i$. Then, *A-IR* outputs the same result as *A-KI* did.

The probability that $j = k$ is $\frac{1}{T}$. If this happens, we have $SK_j = SK_k$. This means that *A-IR* has gotten the secret key SK_j without the knowledge of $(T-1)$ secret keys. In the end, we can build the attacker *A-IR* which would break the $(T-1)$-resilient property, *i.e.*, intrusion-resilient property. Therefore, if our key-evolving protocol is intrusion-resilient, our key-evolving protocol is key-independent in the random oracle model. ■

5.3.4 Unforgeability

The notion of unforgeability that we use for the regular underlying scheme is the strong notion of security for digital signature as formalized in [13] (security against existential forgery under adaptive chosen message attack). This notion can be extended in a natural way to capture also intrusion-resilience of signatures. Our proof of the following theorem follows one of Theorem 1 in [17].

Theorem 5.3.5 *Let* SIG = (*Gen, Sign, Ver*) *be an unforgeable signature scheme and* IRKE-SIG = (IRKE.*Gen,* IRKE.*Upd,* IRKE.*Sign,* IRKE.*Ver*) *be an intrusion-resilient key-evolving signature scheme, then the scheme* IRKE-SIG *constructed above is an unforgeable intrusion-resilient signature scheme.*

Proof: Let *F-IRKE* be a forger against scheme IRKE-SIG that succeeds with probability ε. We build a forger F against the underlying scheme SIG as follows. Let (p, s) be a pair of public/private keys for SIG against which we want to produce forgeries. Forger F is given an oracle $\mathcal{O}_{sig}$ that given a message returns a signature on that message under the pair (p, s). Forger F starts by generating information corresponding to T periods of a IRKE-SIG scheme. F first chooses a period number t_0 at random 1 and T. Then it chooses a random seed for IRKE.*Gen* and generates out of it $T - t_0$ private SIG keys following the specification of the key generation algorithm IRKE.*Gen*. These keys are set as the IRKE-SIG keys for periods $t_0 + 1, t_0 + 2, \ldots, T$. In addition, F generates $t_0 - 1$ random and independent private keys that it sets as the IRKE-SIG keys for periods 1 to $t_0 - 1$. For period t_0 it sets the public key to be p.

Algorithm F now runs the forger *F-IRKE* against the IRKE-SIG scheme defined above. We let *F-IRKE* query for signatures corresponding to any period of its choice except for the following restriction. Whenever *F-IRKE* asks for a signature corresponding to a period i, it cannot later ask for a signature corresponding to any period different from a period i. Each time *F-IRKE* requests a signature (on a message of its choice) corresponding to any period different than t_0, then F provides the requested signature using its knowledge of the signature keys for those periods (these keys were chosen by F). When *F-IRKE* asks to issue signatures for period t_0, then F goes to its oracle $\mathcal{O}_{sig}$ to get the corresponding signatures under (p, s). When *F-IRKE* decides to query the secret information for some t'-th period then F does the following. If $t' = t_0$, then it aborts its run (*i.e.,* in this case F fails to forge). Otherwise if $t' \neq t_0$, then F provides *F-IRKE* with the secret information for that period (F knows it). F keeps running *F-IRKE* as before and responds to signature requests as before. If at some point *F-IRKE* outputs a forgery against a period $t'' \neq t'$ then F acts as follows. If $t'' \neq t_0$, F aborts its run failing to forge. Otherwise if $t'' = t_0$, F outputs the same forgery as *F-IRKE* did and stops. (Note that in order for *F-IRKE*'s output to be considered a forgery it must be that *F-IRKE* did not ask for the forged message to be signed during period t_0, so in particular F did not ask for that signature from $\mathcal{O}_{sig}$ meaning that this is a valid forgery for F too.)

What is the probability of F to succeed in forging? If *F-IRKE* succeeds with probability ε then F succeeds at least with probability roughly ε/T. This argument is outlined as follows. First, the view of IRKE-SIG that F produces for *F-IRKE* is computationally indistinguishable from the view of *F-IRKE* under a real run of IRKE-SIG (where all keys are produced out of a single initial seed for IRKE.Gen). Indeed, using standard techniques it is straightforward to show that if a distinguisher exists for these two views of IRKE.Gen. Next, conditioned on F choosing the value of t_0 as the period for which *F-IRKE* will eventually output a forgery, we have that the probability that F outputs a forgery against (p, s) is the same probability that *F-IRKE* succeeds in forging, *i.e.,* probability ε. Since choosing the "right" t_0 happens with probability $1/T$ we get that ε/T is an approximate lower bound on the forging probability of F. (The "approximate" comes from the negligible probability with which the above mentioned views of *F-IRKE* can be successfully distinguished.) ■

We show that if a forger for the scheme IRKE-SIG exist then we can construct out of it a forger for the scheme SIG, thus reaching a contradiction. We note that a basic difference between a forger against SIG and the one against IRKE-SIG is that the former is never given the signature key, while the latter is provided with the signature key for a period i and it is considered successful if it finds a forgery for a signature corresponding to a time period $i' \neq i$.

Pointcheval *et al.* [21] proved that Schnorr signature scheme is UF-CMA (Unforgeable Against Chosen Message Attack). Therefore, we can reach the following Proposition.

Proposition 5.3.6 *In the random oracle model, our intrusion-resilient key-evolving signature scheme* (IRKE-SIG) *is unforgeable.*

Proof: The proof of this proposition comes immediately from Theorem 5.3.5. ■

Chapter 6 Conclusions

In this thesis, we have studied the design and the analysis of intrusion-resilient key-evolving protocol, in particular, running together with Schnorr signature scheme and Shamir's (k, n) threshold scheme. We have reviewed previous works and presented a new construction.

We have proposed a new key-evolving protocol which combines Shamir's (k, n) threshold scheme and a discrete logarithm based scheme (*e.g.* Schnorr signature scheme). Shamir's (k, n) threshold scheme distributes possession of the secret information of the signer to a group of *Agent*s. Our construction is working on an the discrete logarithm problem over a finite field.

To guarantee sound security of our construction, we have established appropriate security model to prove security of our construction. Especially, we defined the concept of intrusion-resilience which is different from Itkis *et al.*'s intrusion-resilience [15]. In our protocol, even if there are several entities — *Agent*s — as well as the signer, all *Agent*s are required to execute much simpler computational capability than the signer is unlike Itkis *et al.*'s case. To get intrusion-resilience, we have used the properties of a linear system of equations. That is, when we consider our key-evolving protocol as a linear system of equations with T unknown variables and T linear equations, we have showed that if secret keys of all time periods are not compromised, it is not possible to forge signatures relating to non-exposed secret keys. We believe that our intrusion-resilience has the best strength against key exposure problem.

We also have used the random oracle model as a tool to show that any attacker, who breaks the intrusion-resilient key-evolving protocol, can be transformed into an efficient algorithm to solve the underlying problem, namely "the Discrete Logarithm Problem". Finally, we have presented the security proof for all well-defined security requirements.

Our construction has also achieved an efficiency in key-evolving protocol. Actually our key-evolving protocol requires mainly summation operation as a simple one.

Our proposed protocol can be applied in any schemes such as the ElGamal scheme (encryption and signature), the Digital Signature Algorithm (DSA) and so on based on the discrete logarithm problem and the secret information should be distributed to enhance security. The typical example for our protocol is a smart card with secret information which is distributed by several *Agent*s including the signer for validation purpose.

In future work, since there are no previous formal proofs for the schemes related to our proposed one in the literature, we can add provably secure proof to our scheme. This would be very meaningful job. Also the main drawback of our scheme is that key update needs help from *Agent*s. Thus, it would be interesting to find a key-evolving protocol in which key update can be done by the signer alone.

안전한 비밀키 갱신이 가능한 프로토콜에 대한 연구

김중만

공개키 기반 구조에서 개인키의 분실은 가장 심각한 문제 중 하나이며 피할 수 없는 문제로 여겨진다. 이것을 키 노출 문제(key exposure problem)라고 한다. 현재 사용되고 있는 전자서명 기법에서 서명자의 개인키가 노출이 되면, 이러한 사실을 인지하지 못한다 하더라도, 마치 서명자가 서명한 문서를 위조 할 수 있다. 이를 방지하기 위하여 전방 보안(forward-secure) 서명 기법, 문턱 (threshold) 서명 기법, key-insulated 서명 기법, intrusion-resilient 서명 기법 등의 연구들이 이루어져 왔다. 또한 이에 맞춰 키 전개(key-evolving) 프로토콜이 개발 되었는데, 이 모든 기법들은 프로토콜 전 구간을 부분적인 시간 구간으로 나눈 뒤 각 시간 구간들이 진행 되어 갈 때마다 개인키가 갱신된다. 따라서 공격자가 어느 특정한 시간 구간안의 개인키를 획득하더라도 다른 시간 구간의 개인키는 보호되어 질 수 있다. 물론 보호된 개인키로 서명한 문서 또한 위조 할 수 없을 것이다. 이렇게 시간 구간이 전개해 나가면서 개인키 또한 갱신되고 어느 시점에서도 누구나 서명 문서를 검증 할 수 있어야 하므로, 프로토콜 전 구간 동안 검증에 필요한 공개키는 하나의 값으로 고정될 것이다. 이렇듯 위의 방법은 한 사용자의 개인키가 분실 되더라도 그 분실에서 올 수 있는 심각한 문제를 줄일 수 있는 방법이라고 할 수 있다.

본 논문에서는 암호학적으로 어려운 문제인 이산로그의 어려운 문제(discrete logarithm problem) 에 기반 한 키 전개 프로토콜을 설계하고 이것의 안전성에 대한 정량적 근거를 제시한다. 키 전개 프로토콜이 가능한 공격에 대해서 안전하다는 것을 보장하기 위해 키 전개 프로토콜이 얼마나 안전한 가에 대한 정형적이고 엄격한 수학적 증명이 수반되어야 할 것이다. 키 전개 프로토콜에 대한 안전성이 증명되면 이를 서명 기법에 적용한다.

키 전개 프로토콜이 이산로그의 어려운 문제에 기반하고 있기 때문에 같은 문제에 기반한 모든 서명 기법들에 적용될 수 있다. 따라서 본 논문은 안전한 비밀키 갱신이 가능한 키 전개 서명 기법을 제시하고자 한다.

References

1. R. Anderson, “Two Remarks on Public-Key Cryptology”, Invited lecture, Fourth Annual Conference on Computer and Communications Security, ACM, 1997, Availabe at **http://www.cl.cam.ac.uk/users/rja14/**.

2. M. Abdalla, S. Miner, and C. Namprempre, “Forward-Secure Threshold Singature Schemes”, In David Naccache, editor, Progress in Cryptology-CT-RSA 2001, LNCS 2020, 2001.

3. M. Bellare and S. Miner, “A Forward-Secure Digital Signature Scheme”, In Michael Wiener, editor, Advances in Cryptology - Crypto ’99, Springer-Verlag, 15-19 August 1999. Revised version is avaiable from **http://www.cs.ucsd.edu/∼mihir/**.

4. M. Bellare and P. Rogaway, “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”, Proceedings of the First ACM Conference on Computer and Communications Security, pp. 62-73, 1993.

5. M. Bellare and P. Rogaway, “The Exact Security of Digital Signatures - How to Sign with RSA and Rabin”, Advances in Cryptology - Eurocrypt 1996, Springer-Verlag, LNCS 1070, pp 399-416, Berlin, 1996.

6. R. Canetti, O. Goldreich, and S. Halevi, “The Random Oracle Methodology Revisited”, Proceedings of the 30th ACM Annual Symposium on Theory of Computing, pp. 209-218, 1998.

7. R. Cramer and V. Shoup, “Signature Schemes Based on The Strong RSA Assumption”, ACM Transactions on Information and System Security, 3(3):161-185, 2000.

8. Y. Desmedt and Y. Frankel, “Threshold Cryptosystems”, In G.Brassard, editor, Advances in Cryptology - Crypto ’89, Springer-Verlag, LNCS 435, pp. 307-315, 1990.

9. Y. Dodis, J. Katz, S. Xu, and M. Yung, “Key-Insulated Public-Key Cryptosystems”, In Lars Knudsen, editor, Advances in Cryptology - Eurocrypt 2002, Springer-Verlag, LNCS, 28 April-2 May 2002.

10. NIST, “Digital Signature Standard (DSS)”, Federal Information Processing Standards Publication 186, November 1994.

11. T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, IEEE Transactions on Information Theory, IT-31(4):469-472, 1985.

12. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, “Robust Threhold DSS Signatures”, In Ueli Maurer, editor, Advances in Cryptology - Eurocrypt ’96, Springer-Verlag, LNCS 1070, pp. 354-371, 12-16 May 1996.

13. S. Goldwasser, S. Micali, and R. L. Rivest, “A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks”, SIAM J. Computing, 17(2):281-308, April 1988.

14. A. Herzberg, M. Jackobsson, S. Jarecki, H. Krawczyk, and M. Yung, “Proactive Public Key and Signature Systems”, In Fourth ACM Conference on Computer and Communication Security, pp. 100-110, 1997.

15. G. Itkis and L. Reyzin, “SiBIR: Signer-Base Intrusion-Resilient Signatures”, In Moti Yung, editor, Advances in Cryptology - Crypto 2002, Springer-Verlag, LNCS, 18-22 August 2002, Available from **http://eprint.iacr.org/2002/054/**.

16. KCDSA Task Force Team, “The Korean Certificate-based Digital Signature Algorithm”, IEEE P1363a Submission, August 1998.

17. Hugo Krawczyk, "Simple Forward-Secure Singatures from Any Signature Scheme", In Seventh ACM Conference on Computer and Communication Security, November 1-4 2000.

18. C. -F. Lu and S. W. Shieh, "Secure Key-Evolving Protocols for Discrete Logarithm Schemes", In Proceedings of The Cryptographer's Track at the RSA Conference 2002., Springer-Verlag 2002, LNCS 2271, pp. 300-310.

19. R. Ostrovsky and M. Yung, "How To Withstand Mobile Virus Attacks", In 10-th Annual ACM Symp. on Principles of Distributed Computing, pp. 51-59, 1991.

20. K. Ohta and T. Okamoto, "On Concrete Security Treatment of Signatures Derived from Identification", Advances in Cryptology - Crypto 1998, Springer-Verlag, LNCS 1462, pp. 354-369, Berlin, 1998.

21. D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes",In Ueli Maurer, editor, Advances in Cryptology - Eurocrypt '96, Springer-Verlag, LNCS 1070, pp. 387-398, 12-16 May 1996.

22. D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures", Journal of Cryptlogy, 1999, Available from **http://www.di.ens.fr/∼pointche**.

23. R. Rivest, "The MD5 Message-Digest Algorithm", FRC 1321, The Internet Engineering Task Force, April 1992.

24. C. P. Schnorr, "Efficient Identification and Signatures for Smart Card", Advances in Cryptology - Eurocrypt '89, Springer-Verlag, pp. 235-251.

25. C. P. Schnorr, "Efficient Signature Generation by Smart Cards", Journal of Cryptology, 4(3):161-174, 1991.

26. A. Shamir, “How To Share A Secret”, Communications of the ACM, 22(11), pp. 612-613, 1979.

27. NIST, “Secure Hash Standard (SHS)”, Federal Information Processing Standards Publication 180-1, April 1995.

28. W. -G. Tzeng and Z. -J. Tzeng, “Robust Key-Evolving Public-Key Encryption Schemes”, Record 2001/009, Cryptology ePrint Archive, 2001.

Acknowledgements

First, I would like to express my gratitude to Prof. Kwangjo Kim, my thesis advisor, for his constant direction and support. Without his guidance, I could never have carried out my research. Throughout my days in ICU, he gave me many lessons about how to work and to study. Special thanks are also due to Prof. Jae Choon Cha and Dr. Dae Sung Kwon for their generosity and agreeing to serve as advisor committee members.

I also would like to thanks to all members of Cryptology and Information Security laboratory: Byungkon Kim, Songwon Lee, Hwasun Chang, Chuljoon Choi, Sungjoon Min, Sangwon Lee, Jaehyrk Park, Soogil Choi, Kyusuk Han, Zeen Kim, Jeongkyu Yang, Seokkyu Kang, Vo Duc Liem from Vietnam, Yan Xie, Wang Ping, Xiaofeng Chen, Ren Kui, and Jiqiang Lv from China, and Divyan from India, for giving me lots of interests and good advice during the course of my study. I also thanks Jeongmi Choi for her helpful support as a staff member.

In addition, I appreciate to the graduates, Myungsun Kim, Jongseong Kim, Hyunrok Lee, and Wooseok Ham, and also give my thanks Jungyeon Lee, Junbaek Ki, Sangbae Park, and Yunkyoung Jeong of Applied Crypto laboratory. I want to present my sincere gratitude to my fellow student Yutaek Lim, hoping to overcome his difficulties as fast as possible.

I always hope God bless my oldest friend, Yeonju Song, and his family. He has been a best friend to me in adversity or in prosperity.

My love and thanks go to my parents for endless and profound affection and their devotion. Especially, my mother has devoted her lifetime in supporting me and my brother, and my father has supported me for expanding wings of my study and has given his whole life to the family. My brother

Joongsun Kim also has given me warmhearted concerns behind me quietly. I cannot forget their trust and encouragement on me.

Last, but not least, I greatly appreciate my lovely sweetheart, Yunsuk Kim, for her belief and constant encouragement through very difficult situation.

Finally, I will always remember my life of ICU. It made me a grown-up person.

Curriculum Vitae

Name : Joong Man Kim

Date of Birth : May. 11. 1975

Sex : Male

Nationality : Korean

Education

1994.3–2002.2	Mathematics Korea Advanced Institute of Science and Technology (KAIST) (B.S.)
2002.3–2004.2	School of Engineering Information and Communications University (M.S.)

Career

2003.7–	Graduate Research Assistant Research on Secure DRM in Ubiquitous Environment The Ministry of Science and Technology (MOST)

2003.6–2003.8	Apprentice Researcher Cooperative Computing Research Group, Nippon Telegraph and Telephone (NTT)
2003.3–2003.6	Undergraduate Teaching Assistant ICE 0100 University Mathematics School of Engineering, ICU
2002.4–2002.12	Graduate Research Assistant Development of Secure Electronic Trading System for Online Game Items Cemtlo Media Corporation
2002.3–2002.11	Graduate Research Assistant Research on ID-Based Authentication Model and Application Service Electronics and Telecommunications Research Institute (ETRI)
2002.3–	Graduate Research Assistant Research on Cryptographic Primitive using Bilinear Paring The Ministry of Information and Communications (MIC)
2002.3–	Graduate Research Assistant Cultivation of Top-Level IT Security Manpower The Ministry of Information and Communications (MIC)

Publications

(1) 2003.12	김중만, 김광조, "안전한 비밀키 갱신이 가능한 Schnorr 형 서명 기법", 2003년도 한국정보보호학회 동계정보보호학술대회, pp.422-427, 한양대, 2003.12.6
(2) 2003.10	JoongMan Kim and Kwangjo Kim, "Intrusion-Resilient Key-Evolving Schnorr Signature", Proc. of CSS2003, pp.379-384, Oct. 29∼31,2003 Kitakyushu, Japan, IPSJ
(3) 2002.7	J.M. Kim and J. Cheon, "A Cryptographic Protocol for On-line Test," Proceedings of KIISC Conference Region Chung-Cheong, pp.145-152 (2002)