私立東海大學資訊工程與科學系研究所

碩士論文

指導教授：林 祝 興 博士

(Dr. Chu-Hsing Lin)

植基於橢圓曲線的階層式金鑰管理之研究

A Study of Hierarchical Key Management Schemes Based on Elliptic Curves

研究生：李 家 豪

(Jia-Hao Lee)

中 華 民 國 九 十 二 年 六 月

摘要

在一個系統中，如果所有的使用者依照其權責可被分類為數個不同類別而形成一個階層式架構，而在這種階層式架構中，屬於比較高階層類別中的使用者將有存取比較低階層類別的使用者所有的訊息及文件之權力，反之則否。為了達成這種階層式架構的存取控制，Lin 在 1997 年於 Computer Communications 期刊上發表了一個嶄新的方法，可在可變動階層式組織架構中，達成動態的金鑰管理。在這個方法中，他設計了一個特別的關聯參數(related parameters)，使得金鑰管理更加容易及富有彈性。

然而這些關聯參數的產生需要進行許多模指數運算(modulo exponential computation)，對於高變動性的階層式架構將會有效能方面的隱憂，而且在 1998 年，Lee and Hwang 發現 Lin 的方法有若干安全問題存在。為了解決這些問題 Cho 發表了一個修補的方法。雖然 Cho 的方法可以解決原方法在安全上的問題，由於使用了額外的參數而且同樣使用模指數運算，使的整體的運算量比原方法來的多。所以在本論文中，我們提出一個利用橢圓曲線特性的新方法來解決原方法在安全方面問題並提升其效能，而且更進一步利用智慧卡的優點強化整個方法的安全性。

關鍵字：階層式金鑰管理；資料安全；橢圓曲線；分權化的階層架構；存取控制；智慧卡

Contents

List of Figures

Abstract

In a user hierarchy that users are authorized and classified into different privilege classes, a user belonging to higher-privileged class will have access right to messages created or owned by users in a lower-privileged class; while the opposite is not allowed. To meet the access control requirement in a user hierarchy, Lin proposed a dynamic key management scheme suitable for hierarchical control systems and that appeared in Computer Communications. It is useful in solving the key management problem for a privileged hierarchy using related parameters. However, the related parameters are generated by modulo exponential computation. This seems not efficient enough for a system with frequently changing requirements. Besides, Lee and Hwang proposed two comments to this scheme. To eliminate the possible drawbacks as discussed in their comments, Cho proposed modifying the scheme by adding a parameter. Although Cho's scheme is an improvement in security, it is not as efficient as Lin's scheme. In this thesis, we propose a novel scheme, based on the elliptic curves that eliminates the drawbacks and has better performance. Furthermore, we also use the advantages of smart card to enhance our proposed scheme.

Keywords: hierarchical key management; data security; elliptic curves; privileged hierarchy; access control; smart card

1. Introduction

Hierarchical structures are used in many access control methods. Much research has focused on controlling access to system resources securely and efficiently. Consider a computing system in which the users are classified into n disjoined privileged classes, namely $S = \{S_1, S_2, ... S_n\}$. A relation '$\leq$' is defined on the S. If we have $S_i \leq S_j$, it means that S_i has lower privilege than S_j.

Based upon cryptographic techniques, several schemes [20-23] have been proposed to solve the access control problem in a hierarchy. For cryptographic implementations, each class in S has a group key k_i, $i = 1,2,...n$. For the relation $S_i \leq S_j$, computing the group key k_i is easy by S_j's group key k_j. There is no way to derive k_j by knowing k_i.

In this thesis, we'll talk about Lin's scheme [5] for access control in a hierarchy. And we'll also discuss about the problems brought up by Lee and Hwang [4] and the improved scheme proposed by Cho [2]. Finally, we propose a novel scheme, based on the elliptic curves that eliminates the drawbacks and has better performance. Furthermore, we also use the advantages of smart card to enhance our proposed scheme.

2. Background

2.1 One-Way Hash Function

One-Way hash function acts an important role in modern cryptosystem. It may have several names in different situation when we use it. Its name could be called as compression function, contraction function, message digest, fingerprint, cryptographic checksum, and message integrity check. The most important characteristic of one-way hash function is that we can use it easily to generate a hash value for a message, but it is hard that we try to reverse the one-way hash function process in order to get original message from the hash value. We say a one-way hash function is collision-free if it is hard to find two messages with the same hash value.

In this thesis, we use the one-way hash function to generate the message digest and use it to make some secure value with some secret parameter.

2.2 Symmetric key cryptography

Symmetric key cryptography is a traditional form of cryptography. People can use a key to encrypt a message and decrypt the message with the same key. The advantage of symmetric key cryptography is that the encryption/decryption process is much faster than asymmetric key cryptography. However, it is an important issue in symmetric key cryptography that how to two ends shares an agreed secret key without anyone else getting it. Some techniques could solve the problem with eliminating the misgiving of eavesdropping. There are two basic types of symmetric key cryptography: block cipher (such as DES, 3DES, and AES) and stream cipher (such as A5 used in GSM).

2.3 Elliptic Curve Cryptography

Elliptic Curve Cryptosystem (ECC) was proposed by Neal Koblitz [10] and Victor Miller [11] in 1985. The security of elliptic curve cryptosystem is based on the difficulty of computing an elliptic curve discrete logarithm problem (ECDLP) [7] [8]. Due to numerous researchers have been done on its security and efficient implementation, ECC has accepted by standard organizations.

In this section we will give a quick introduction to elliptic curve. Let E be an elliptic curve over Z_p and G be a point on E of order n, i.e. $G \in E(Z_p)$. In general applications, p is typically a power of 2 or an odd prime number. Then we can choose a number "s" and let $Q = s \times G$, where $0 \le s \le n-1$ and Q is also a point on E. If n and p are large enough, it is hard to find s with knowing E, Q, and G. This is called the elliptic curve discrete logarithm problem (ECDLP).

An elliptic curve E over Z_p is defined as following equation

$$y^2 = x^3 + ax + b$$

where $a, b \in Z_p$ and $4a^3 + 27b^2 \neq 0 (\bmod p)$, and all the point ($x$, y), $x \in Z_p$, $y \in Z_p$, form the set of $E(Z_p)$ containing a point O called the point at infinity. When a point G on the elliptic curve E multiplied by a number s, it is equivalent to adding G to itself s times, and will yield another point on the curve. A rule, called chord-and-tangent rule, is utilized to add two points on an elliptic curve to get another point. We now describe the addition formula on the elliptic curve.

If $G = (x, y) \in E(Z_p)$ then $G + O = O + G = G$ and $G + (-G) = O$, where

$-G = (x, -y)$ called the negative of G.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E, i.e. $P, Q \in E(Z_p)$. The formula for adding P and Q are describe as follows:

$R = P + Q = (x_3, y_3)$, where $P \neq -Q$, and

$$\begin{cases} x^3 = \delta^2 - x_1 - x_2 \\ y^3 = \delta(x_1 - x_3) - y_1 \end{cases} \quad \text{and} \quad \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & if \quad P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & if \quad P = Q \end{cases}$$

Figure 1 and 2 are the geometric description of the addition rule. In Figure 1 $P \neq Q$, drawing a line through P and Q, it will intersect the elliptic curve in third point. Then R is the reflection of the third point in x-axis.

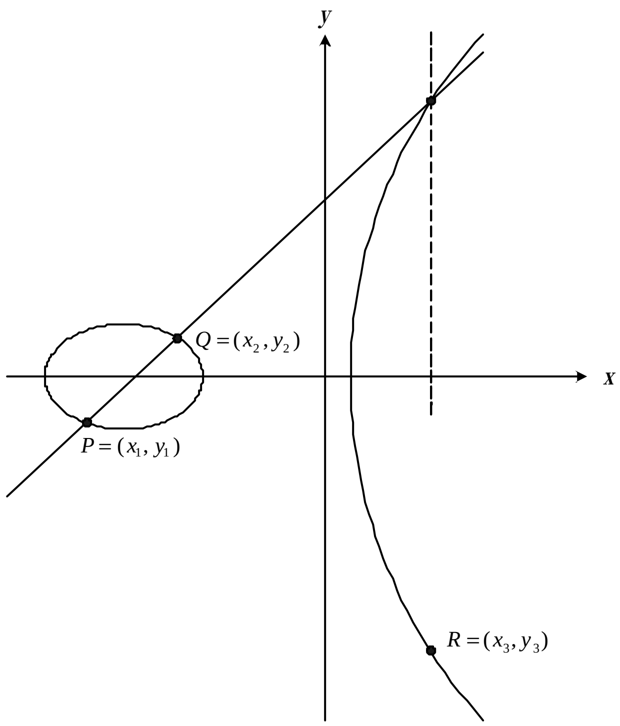

Figure 1 the addition of two distinct points R=P+Q

In figure 2, $P = Q$, drawing a tangent line to the elliptic curve at P, it will intersect the elliptic curve in second point. Then R is the reflection of the third point in x-axis.

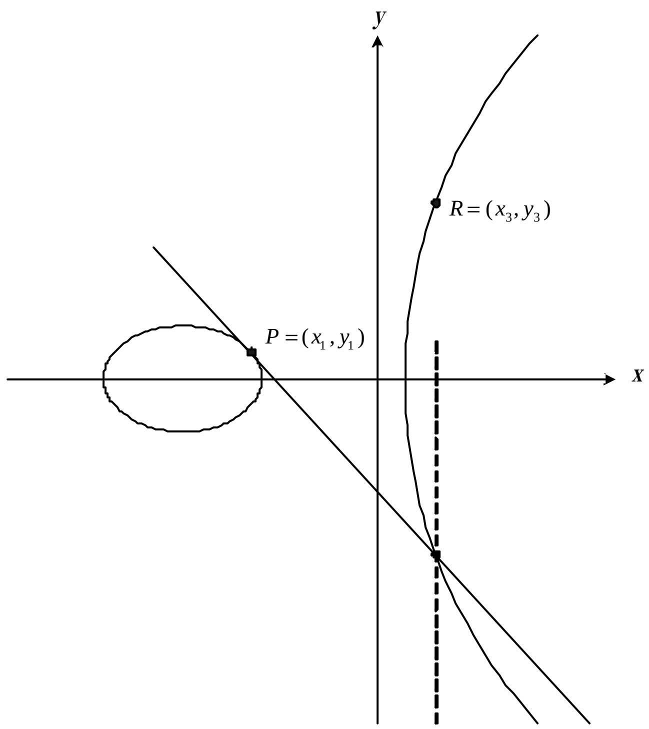

Figure 2 doubling of a point *R*=*P*+*P*

2.4 Smart Card

The smart card, an intelligent token, is a credit card sized plastic card embedded with an integrated circuit chip. It provides not only memory capacity, but computational capability as well. Nowadays, the size of storage and ability of computation of smart card continue to increase. Besides, the chip on smart card also allows the implementation of cryptographic and authentication scheme. Hence, in this thesis, we propose two cryptographic protocols based smart card. The Figure 3 shows the physical appearance of smart card.

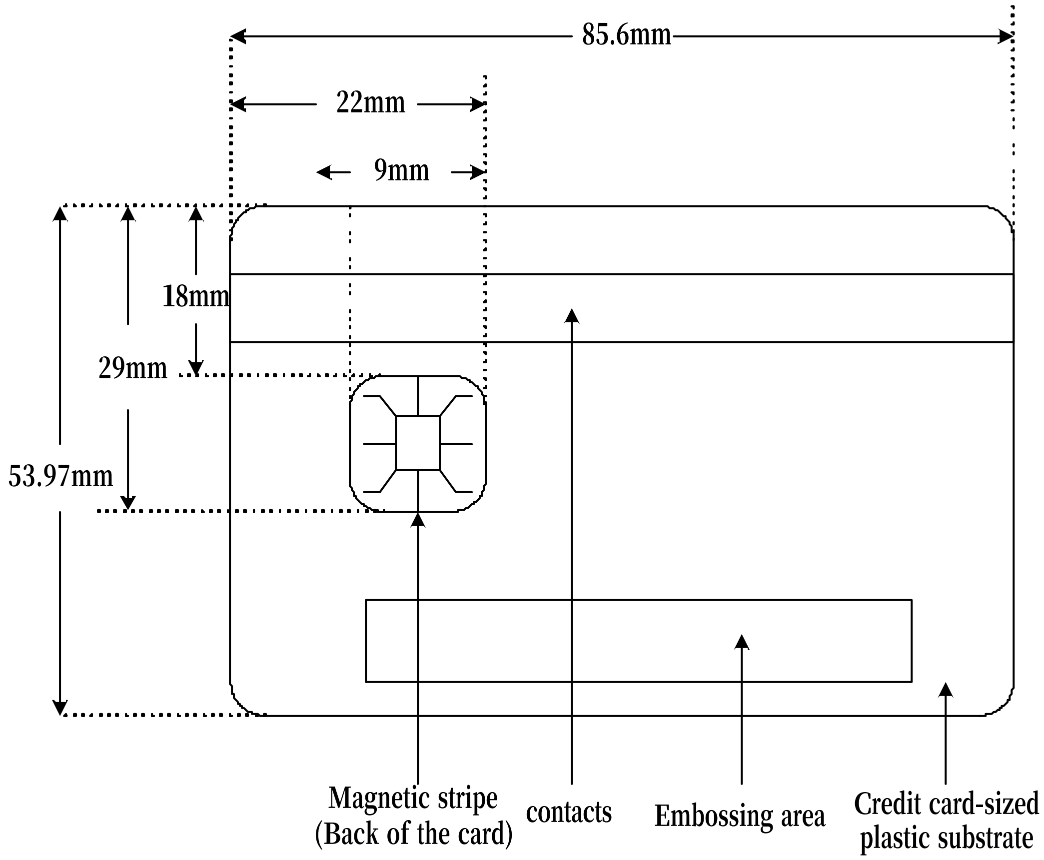

Figure 3 smart card physical appearance

In general, smart card should have the ability of tamper resistance to prevent some malicious explore to the data in the smart card. There are several types of smart card:

- Memory cards
- Processor cards
- Electronic purse cards
- Security cards
- Java Card

With the development of new technology, there are many smart card related standards. We describe these standards below.

Horizontal standards

- ISO 7816 – This describes the lowest-level interface to a smart card. It is at this that data bytes are transferred between card readers and card and it is the most important standard defining the characteristic of chop cards that have electrical contacts. ISO 7816[16] covers various aspects of smart cards:
  - Part 1 – physical characteristics
  - Part 2 – dimensions and location of the contacts
  - Part 3 – electronic signals and transmission protocols
  - Part 4 – interindustry commands for interchange
  - Part 5 – application identifiers
  - Part 6 – interindustry commands for SCQL
- PC/SC – It is the standard for communication with smart cards connected to personal computer system. [17]
- PKCS #11 – This is an interface between application and all kinds of portable cryptographic devices. [18]
- OCF – OCF is an all-Java interface for communicating with smart cards from a Java environment. [19]
- Java card – It describe the Java Card and what it supports. [20]
- Multos – It is a multi-application operation system for smart cards. [21]

Vertical standards

- Mondex – A kind of digital cash that uses smart cards only. The Mondex approach does not allow cash to exist outside of the card. [22]
- CEPS – The main purpose of the common electronic purse specifications is to define requirements for all components needed by an organization to implement

a globally interoperable electronic purse program and to maintain full accountability and auditability. [23]

- MPCOS-EMV – This is general-purpose card that lets you implant your own type of currency or token. [24]

3. Dynamic key management schemes for access control in a hierarchy

Based upon cryptographic techniques, several schemes [6] [9] [12] [13] have been proposed to solve the access control problem in a hierarchy. In these schemes, a trust agent (TA) is assumed to existing for generating and distributing the key and parameters. But, there are some drawbacks in these schemes. Firstly, if there is any one new class joining or leaving the hierarchy, the whole keys and parameters have to be recomputed. Secondly, when the number of classes has been inserted, the size of storage which stores the parameters will grows dramatically.

In order to solve the problem, Lin proposed a dynamic key management scheme to fit the above requirements. By using designed public information, namely the related parameters, between any two classes, Lin’s method make the key management more flexible and decreases the parameters required in a hierarchical structure with keys that change frequently.

However, there are two potential drawbacks that were addressed by Lee and Hwang. First, if an attacker had the group key for some class, he (or she) could easily acquire a new group key when that class changed to a new key. Second, the sibling attack would be feasible using the differences in IDs between two sibling classes. Although Cho proposed a scheme to eliminate these drawbacks, Cho’ s scheme is not as efficient as Lin’ s original scheme.

3.1 Lin's scheme for access control in a hierarchy

In Lin's scheme, we allow each class in a hierarchy to choose or change its own group key without affecting the other keys. The scheme can be divided into the several procedures.

- **Initial**

A trusted CA (Central Authority) is assumed to exist in this system. And the users of the system can be classified into *n* privilege classes. We denote the classes as $S = \{S_1, S_2, \cdots, S_i, \cdots, S_n\}$, where S is a partially ordered set under the binary relation denoted by "$\leq$" and each class has a corresponding ID which be denoted $ID_1, ID_2, \cdots, ID_i, \cdots, ID_n$. In figure, we assume that there are 7 classes in the system. We represent the relationship of classes as a directed tree.

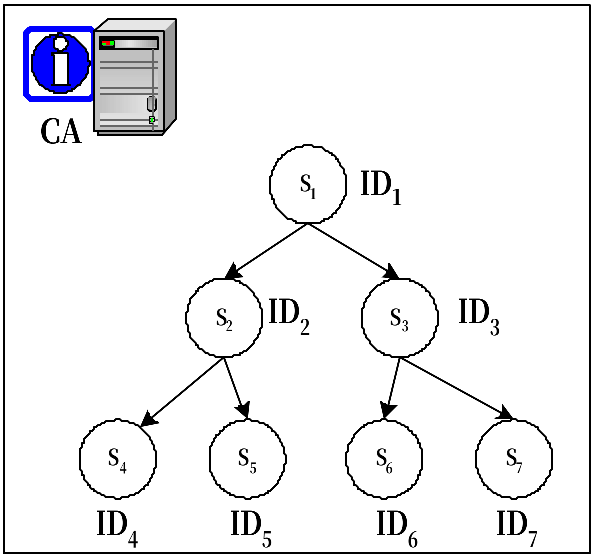

Figure 4 the sample privileged hierarchy

- **Group key generation and parameter calculation**

The CA chooses a large prime P and a primitive element $Z \in GF(P)$ as the public parameters. Then CA uses the public key scheme such as RSA to generate his own public key PK and secret key SK. After each class S_i randomly generates its own group key k_i, the classes encrypt their own group keys using CA's public key PK and sends it to the CA. Once the keys of all classes are collected, CA computes the related parameter r_{ji} for two classes with a branch according to the privileged hierarchy using the following equation:

$$r_{ji} = (Z^{k_j \oplus ID_i} \bmod P) \oplus k_i \tag{1}$$

When all related parameters have been generated, the parameters will be published or maintained by CA. Figure 5 represents an example of the procedure.

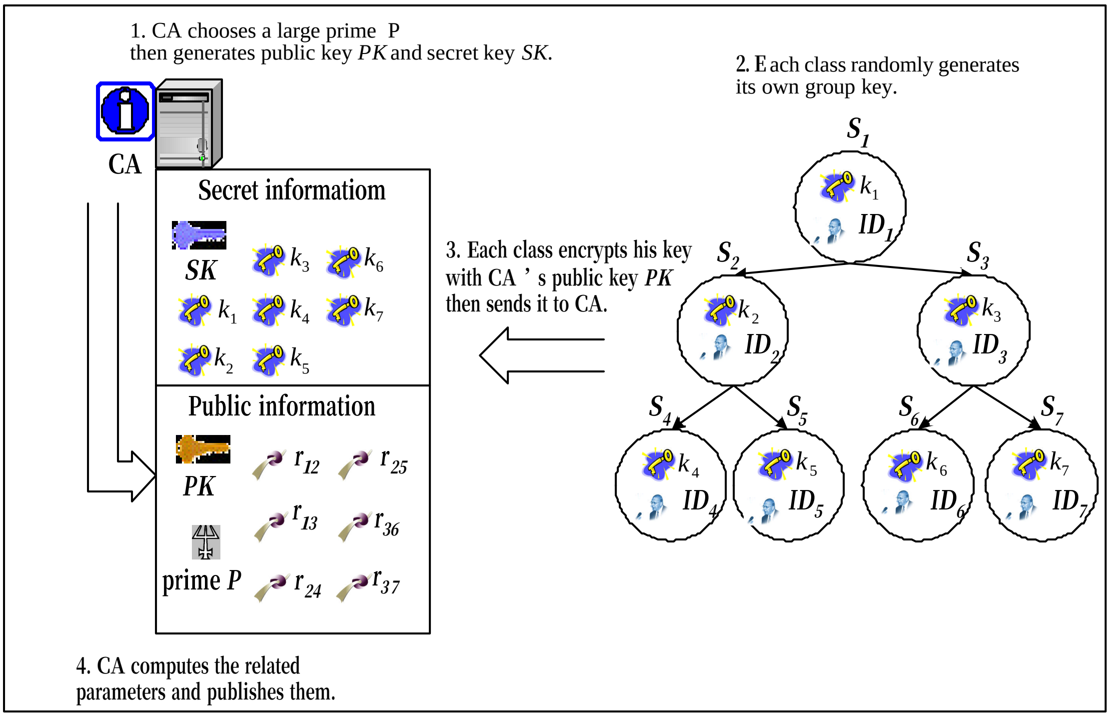

Figure 5 group key generation and parameter calculation in Lin's scheme

- **Group key derivation**

Group key derivation procedure is very sample in Lin's scheme. We assume that class S_j has higher privilege than class S_i. And one member of class S_j wants to get the key of class S_i. He can simply follow these steps:

Step1. He has to gets the related parameters r_{ji} which associate with class S_j and S_i from public board or sends a request to CA. He also has to know the class ID of target class.

Step2. Then he use his own key k_j, related parameter r_{ji} and the target class ID ID_i. He will obtain the key k_i by equation (2).

$$k_i = (Z^{k_j \oplus ID_i} \bmod P) \oplus r_{ji} \tag{2}$$

- **Inserting a new class into hierarchy or removing an existing class**

For inserting or removing classes into existing system, the CA just has to update several of related parameters which are corresponding to the classes. For example in Figure 6, if the class S_8 has been inserted, S_8 generates his own key k_8 and sends it to CA. Then CA computes a new related parameter r_{38} and makes it public or maintains it. The procedure of removing a class is similar as inserting.

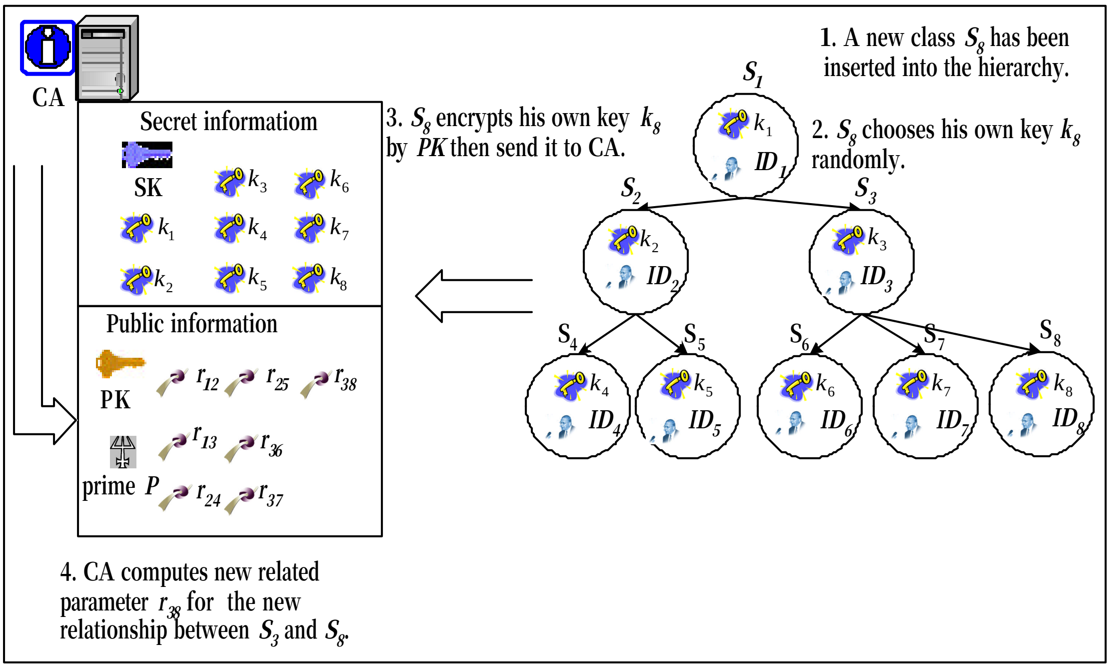

Figure 6 insert a new class into privileged hierarchy

- **Modify an existing group key**

When class S_i changes its group key k_i to a new key k_i^*, S_i just returns its new key to the CA and CA computes new related parameters r_{ji}^* for the relationship $S_i \leq S_j$ by equation (3) and r_{ik}^* for the relationship $S_l \leq S_i$ by equation (4).

$$r_{ji}^* = (Z^{k_j \oplus ID_i} \bmod P) \oplus k_i^* \tag{3}$$

$$r_{il}^* = (Z^{k_i^* \oplus ID_i} \bmod P) \oplus k_l \tag{4}$$

Figure 7 shows an example that class S_2 changes his key k_2 to new key k_2^*.

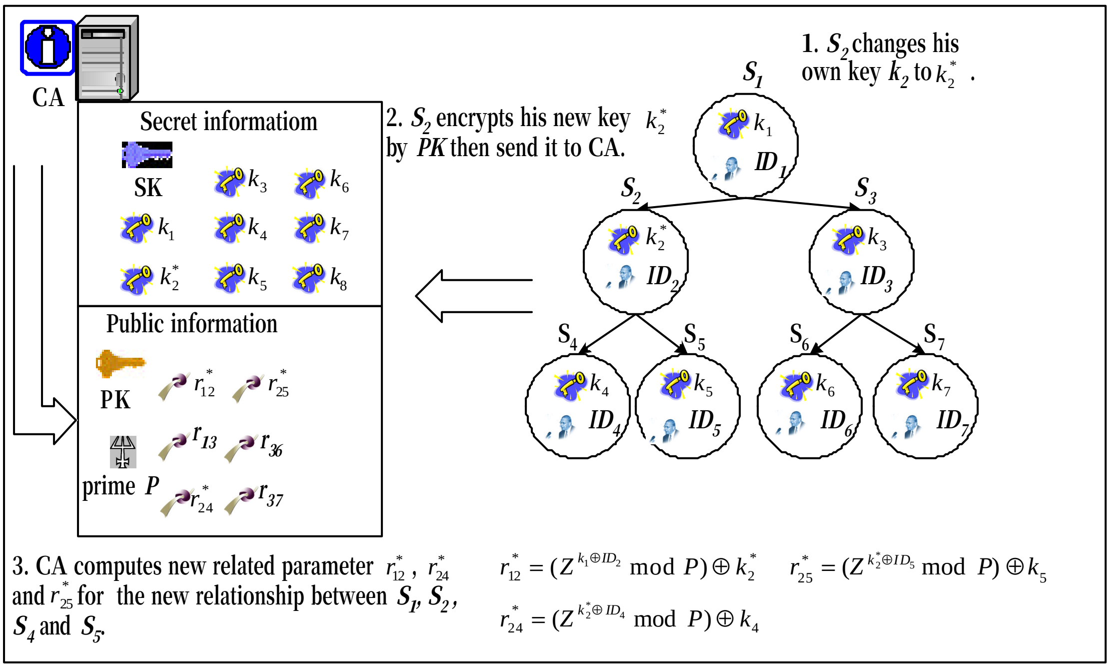

Figure 7 group key modification of class S_2

3.2 Lee and Hwang's Comment on Lin's scheme

Lee and Hwang pointed out two potential weaknesses and made two comments on Lin's scheme:

Comment 1: weakness in case of group key modification procedure

If an attacker has the old key k_i and the related parameter r_{ji} he can compute:

$$Z^{k_j \oplus ID_i} \bmod P = r_{ji} \oplus k_i \tag{5}$$

This value will be used in generating the new related parameter:

$$r_{ji}^{*} = (Z^{k_j \oplus ID_i} \bmod P) \oplus k_i^{*} \tag{6}$$

An attacker could easily obtain the new group key k_i^{*} if he can also get the new r_{ji}^{*}.

Comment 2: the sibling attack may be feasible

If S_i and S_l are within the same parent node S_j, and the attacker is a member of the class S_i. He could crack the S_l's group key k_l using the relationship between the following equations:

$$Z^{k_j \oplus ID_i} \bmod P = r_{ji} \oplus k_i \tag{7}$$

$$Z^{k_j \oplus ID_l} \bmod P = r_{jl} \oplus k_l \tag{8}$$

The values, $Z^{K_j \oplus ID_i} \bmod P$, ID_i, ID_j are known to him. He could easily determine the difference in bits between ID_i and ID_j. If he accomplishes this, he could acquire the information

$$(k_j \oplus ID_i) \oplus (k_j \oplus ID_l) = ID_i \oplus ID_l \tag{9}$$

If there are t different bits between ID_i and ID_l, then there are 2^t possible values for $Z^{k_j \oplus ID_l} \bmod P$. These values can be derived from $Z^{k_j \oplus ID_i} \bmod P$.

3.3 An improved scheme proposed by Cho

To eliminate the potential drawbacks pointed out by Lee and Hwang, at The Second Information Security Application Workshop, Cho proposed an improved scheme based on Lin's scheme by adding the parameter SG

$$SG = hash(k_j + ID_i) \cdot k_j \bmod f(P) \ , f(P) = P - 1 \tag{10}$$

$$r_{ji} = (Z^{SG} \bmod P) \oplus (k_i \cdot SG^{-1} \bmod P) \tag{11}$$

where SG^{-1} is the multiplicative inverse of SG with modulo P.

This scheme can eliminate the drawbacks, but requires more computation than Lin's scheme.

4. Proposed scheme

4.1 An Efficient Hierarchical Key Management Scheme Based on Elliptic Curves

In this section, we describe our proposed scheme. Based on Lin's scheme, we assumed that there is a trust agent, TA. TA has the same functionalities as CA in Lin's scheme. The major procedures are described in the following three subsections.

- **Group key generation**

In this stage, each class will generate its own group key independently. The related parameters are then generated.

Step 1： TA selects an elliptic curve E over Z_p.

Step 2： TA selects a generation point $G \in E(Z_p)$, and finds a large prime q such that $q \times G = O$. CA also selects his private key K and computes $Y = K \times G$ as CA's public key.

Step 3： Each class S_i, $1 \le i \le n$, selects a random number k_i where $k_i \in [1, q-1]$ is the group private key. $p_i = k_i \times G = (x_i, y_i)$ is then computed as the corresponding group public key.

Step 4： Each class encrypts its' private and public keys using TA's public key Y and sends them to TA. TA then decrypts the keys using his own private key K.

Step 5： For each relation $S_i \le S_j$, CA computes a related parameter r_{ji} using

Eq.(11)

$$r_{ji} = \mathrm{X}((k_j \oplus h(x_i \parallel y_i)) \times G) \oplus k_i \tag{12}$$

Here $h()$ denotes a one-way hash function such as SHA-1 or SHA-2 [10] and $\mathrm{X}()$ is defined as Eq.(13).

$$\mathrm{X}(p_i) = x_i \oplus y_i \text{, where } p_i(x_i, y_i) \text{ is a point in } E(Z_p) \tag{13}$$

- **Group key derivation**

For the relation $S_i \le S_j$, k_i can be derived by k_j using the following equation:

$$k_i = \mathrm{X}((k_j \oplus h(x_i \parallel y_i)) \times G) \oplus r_{ji} \tag{14}$$

Sometimes, we may find that there are many ways that class S_j can derivate k_i. We shall choose a simplest way to derivate the key for class S_j. We take an example like figure 8. We assume that class S_1 wants to get the key of class S_7. Obviously, we can see that there are four paths for derivation. But we have to choose a shortest path for derivation. For solving this problem, we just let the transversal cost of each edge is 1. And we apply the signal source shortest path algorithm [14]. By the algorithm, we can simply find a shortest path which starts at class S_1 and passes through class S_2 to class S_7. Then we can reduce computation required in derivation.

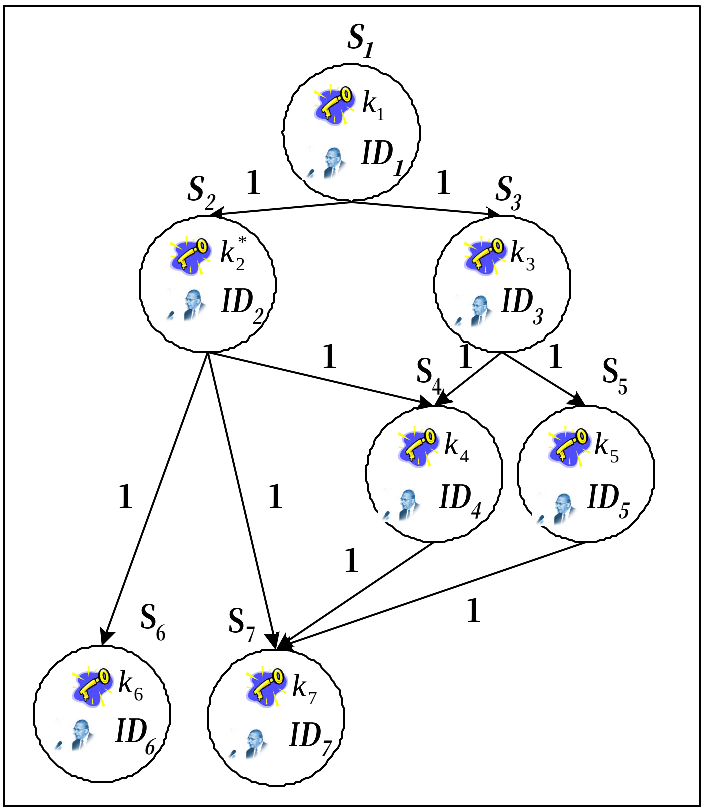

Figure 8 find a shortest path for derivation

- **Modifying a group key**

If class S_i wants to modify its group key k_i, S_i generates a new key k_i^* and sends it back to TA. TA will compute the new related parameter r_{ji}^*.

$$r_{ji}^* = \mathrm{X}((k_j \oplus h(x_i^* \,\|\, y_i^*)) \times G) \oplus k_i^* \tag{15}$$

4.1.1Security Analysis

The following assumptions are made.

✧ The chosen elliptical curve $E(Z_p)$ has a point $P \in E(Z_p)$ whose order is 160-bits prime. The private keys, k_i which $1 \le i \le n$, will be a 160-bit long integer.

✧ The chosen hash function has 160-bit long output results.

Note that the XOR is indicated as $a \oplus b$, where a and b are represented by a binary string with a 160 bit of length. If a or b is longer than 160 bits, it will be truncated from the LSB bits. If a or b is shorter than 160 bits, padding bits are added to the LSB such that it becomes 160-bits long.

- **Attack 1：contrary attacks**

For a lower privileged class S_i with a parent class S_j, a member of class S_i that wants to crack the key k_j of S_j. He has two ways to try. First, he can try to crack k_j using his key k_i and the related parameter r_{ji}. He will find that he can't get any information about k_j from equation $r_{ji} = \mathrm{X}((k_j \oplus h(x_i \parallel y_i)) \times G) \oplus k_i$. Secondly, he may try to crack the equation $p_j = k_j \times G$. He will face a difficult problem of ECDLP (Elliptic Curve Discrete Logarithm Problem). [8] Therefore cracking a parent key is infeasible for a lower privileged class.

- **Attack 2：interior collecting attacks**

If there is a lower privileged class S_i that has m parent nodes, namely S_j, S_{j+1} …and S_{j+m}. The attacker is a member of class S_i. He could try to collect the related parameters r_{ji}, $r_{(j+1)i}$, …and $r_{(j+m)i}$ associated with his parents. He then might try to find the relationships between the values computed using Eq.(15) to crack any parent group key.

$$\mathrm{X}((k_{j+v} \oplus h(x_i \parallel y_i)) \times G),\ v = 1,\dots,m \tag{15}$$

The point $(k_{j+v} \oplus h(x_i \| y_i)) \times G$ is on the elliptical curve, it is difficult to solve $k_{j+v} \oplus h(x_i \| y_i)$ knowing $(k_{j+v} \oplus h(x_i \| y_i)) \times G$ and G. And according to the $X()$ function, the attacker cannot determine the point value of $(k_{j+v} \oplus h(x_i \| y_i)) \times G$. It is infeasible to determine any relationship in these values.

- **Attack 3：exterior collection attacks**

The attacker is outside the system and wants to perform the same process in Attack 2. It is infeasible for the attacker to crack any group key. Because he does not have the group key k_i, cracking the code will be more difficult than Attack 2 for him.

- **Attack 4：collaborative attacks**

If there is a higher privileged class S_j that has two child nodes S_i and S_l. Classes S_i and S_l attempt to crack the group key k_j collaboratively using k_i and k_l. They can obtain two equations:

$$r_{ji} = X((k_j \oplus h(x_i \| y_i)) \times G) \oplus k_i$$

$$r_{jl} = X((k_j \oplus h(x_l \| y_l)) \times G) \oplus k_l$$

Similarly, they face the same problem that occurs in Attack 2.

- **Attack 5：sibling attack**

Consider a case in which a class S_i with a parent node S_j wants to crack the group key of class S_l that has the same parent S_j. He only holds the equation

$r_{ji} = X((k_j \oplus h(x_i \parallel y_i)) \times G) \oplus k_i$. Cracking the code will be more difficult than Attack 4. This kind of attack will not work.

● **Attack 6：Lee and Hwang's comments on Lin's scheme**

Comment 1： knowing the old key information when changing the key value

If a class S_i with a parent node S_j wants to change his group key. The main weakness in Lin's scheme is the value, $(Z^{K_j \oplus ID_i} \bmod P)$, which is unchanged after the new key is generated. Therefore, the attacker may be able to easily steal the new key. In our scheme, we use the elliptic curve crypto-scheme to blind the parent's group key information, and also add a computation with the S_i's public key p_i to make sure that r_{ji}, $X((k_j \oplus h(x_i \parallel y_i)) \times G)$ and k_i will be changed in each process. we have eliminated this weakness.

Comment 2： sibling attack by knowing the bit difference between the IDs of two sibling classes

Our equation, $r_{ji} = X((k_j \oplus h(x_i \parallel y_i)) \times G) \oplus k_i$, does not use any user's "ID" values in computation. This weakness is therefore eliminated.

4.1.2Time Complexity

In this section, we will compare the performance of the proposed scheme with Lin's scheme. Lin's scheme is based on the DLP difficulty (Discrete Logarithm Problem). For practical implementation, we often choose a 1024-bit large prime as the modulus to ensure that solving DLP will be difficult. An elliptical curve $E(Z_p)$ with a point $P \in E(Z_p)$ whose order is 160-bits prime offers approximately the same level of security as DLP with 1024-bits modulus. The following assumptions are made:

- There are n disjointed classes in the system.
- The group keys in both Lin's and our schemes are 160-bit random integers.
- In Lin's scheme, we assume that $Z^{k_j \oplus ID_i} \bmod P$ with P a 1024-bit prime and $k_j \oplus ID_i$ is 160-bits long.

1. In our scheme, we assumed that an elliptical curve is chosen $E(Z_p)$ with $p \approx 2^{160}$.

In our scheme, the one-way hash function has 160-bit output results, like SHA-1.

Some notations are defined as follows:

T_{MUL}: the time needed for a 1024-bits modular multiplication.

T_{EXP}: the time needed for the modular exponentiation with 1024-bits modulus.

T_{EC_MUL}: the time needed for an elliptic-curve multiplication with 160-bits multiplier.

According to [3], we will know the relationship:

$T_{EXP} \approx 240 T_{MUL}$; $T_{EC\_MUL} \approx 29 T_{MUL}$

In the group key generation phase, Lin's scheme generates n keys and n related parameters. The time for key generation in Lin's scheme can be ignored because the key is a randomly chosen 160-bit integer. The time for generating n

related parameters will take n times the 1024-bit modular multiplication and two XOR operations.

Our scheme takes $n+1$(CA's keys included) key-pair generations and n related parameter generations. Each key pair and one related parameter take a single elliptic-curve multiplication with a 160-bit multiplier. Our scheme uses $2n+1$ times the elliptic-curve multiplication with a 160-bit multiplier.

Both schemes take n encryption/decryption processes when key sending and receiving. We presume that the time for the encryption/decryption processes will not be included in the comparison.

The equations in Lin's scheme in the second phase require the same number of computations. Each equation takes one 1024-bit modular multiplication and two XOR operations. In our scheme, both phases take one concatenation, two XOR and one elliptic-curve multiplication with a 160-bit multiplier.

We consider one-way hash function operations to be much faster than modular exponentiation and elliptic-curve multiplication. The time needed for hash function operation will therefore be ignored. The time for XOR and concatenation '||' are also ignored. Using the above assumptions, the computation time for our scheme against Lin's scheme is summarized in Table 1.

	Lin's scheme	The proposed scheme
	Time complexity	Time complexity
Group key generation	$nT_{EXP} \approx 240nT_{MUL}$	$(2n+1)T_{EC\_MUL} \approx 29(2n+1)T_{MUL} = (58n+29)T_{MUL}$
Group key derivation	$T_{EXP} \approx 240T_{MUL}$	$T_{EC\_MUL} \approx 29T_{MUL}$
Modifying group key	$T_{EXP} \approx 240T_{MUL}$	$T_{EC\_MUL} \approx 29T_{MUL}$

Table 1. Time complexity comparison

It is obvious that our scheme is more efficient than Lin's scheme.

4.2 Hierarchical key management using smart card based on elliptic curve

In hierarchical key management, we often assume that there is a TA in the system. TA collects all keys of users and generates the related parameters according to system hierarchy. Although the related parameters will be generated by some special methods to assure that any illegal user can't get any information from these parameters. But these public parameters may have some potential weakness that we can't nose out easily. The illegal users may collect parameters and try to crack them. If we can make these parameters harder to get, the system will be more secure. For hiding the information of parameters, it is a nice solution that using temper-resistant smart cards. In this section, we will describe our second scheme. Based on the first proposed scheme, we use smart card to store parameters and to compute secret information.

As the previous scheme, the scheme also can be divided into following procedures:

- **Initial**

We also assume that there is a TA existed in our system, and there are n classes denoted as $S = \{S_1, S_2, \cdots, S_i, \cdots S_n\}$, where S is a partially ordered set in our system. And each class has a corresponding ID which be denoted $ID_1, ID_2, \cdots, ID_i, \cdots, ID_n$.

- **Group key generation and smartcard registration phase**

This phase is similar with the group key generation phase in previous scheme.

TA will choose and generate the parameters needed in the system. But there is some difference between the scheme and previous one. The keys of all classes are generated by TA when the system starts up first time.

Step1. TA chooses an Elliptic curve over Z_p, then selects a generation point $G \in Z_p$ and finds a large prime q which it is satisfied the equation $q \times G = O$.

Step2. TA chooses a key *SK*. This key should be kept secretly.

Step3. TA generates the secret keys k_i $(1 \le i \le n)$ of all classes and calculates the corresponding public keys p_i ($p_i = k_i \times G$).

Step4. For each relationship $S_i \le S_j$, CA generates the related parameters r_{ji} by the equation:

$$r_{ji} = X(k_j \oplus h(x_i \| y_i) \times G) \oplus k_i .$$

$X(\bullet)$ is function that we define as:

$$X(P_i) = X(x_i, y_i) = x_i \oplus y_i$$

$h(\bullet)$ is a hash function like SHA-1.

Step4. TA use *SK* to encrypt each r_{ji}. We denote the encrypted r_{ji} as $E_{SK}(r_{ji})$. $E_{SK}(r_{ji})$ represents that we use a symmetric cryptographic function like AES with key *SK* to encrypted the message r_{ji}. After encryption, TA separates $E_{SK}(r_{ji})$ into two half parts, $E_{SK}(r_{ji})\_1$ and $E_{SK}(r_{ji})\_2$.

Step5. TA puts ID_i, the key k_i, the key SK and all second part of all encrypted related parameters into smart card of class S_i. Then TA puts all first part of encrypted related parameters in public or maintains them.

- **Group key derivation**

For the relationship $S_i \le S_j$, if there is a member of class S_j wants to get the key of class S_i. He should follow the under steps.

Step1. He inserts his smart card into smart card reader. He has to enter the right PIN code to assure that he is a legal user.

Step2. The terminal software will automatically communicate with TA to determine any update required. If there are updates needed, the software will put updates into smart card and replace old one. (This part will be discussed in group key modification procedure.)

Step3. The terminal software will help him to find the public information about class S_i: the public key of class S_i, p_i, and the half of encrypted related parameter associated with class S_i, $E_{SK}(r_{ji})\_1$. Then the terminal software sends them into smart card.

Step4. The smart card combines two part of encrypted related parameter, $E_{SK}(r_{ji})\_1$ and $E_{SK}(r_{ji})\_2$ as $E_{SK}(r_{ji})$. Then smart card decrypts r_{ji} and uses the parameters $p_i = (x_i, y_i)$, r_{ji} and his own key k_j to computes the key k_i by following equation:

$k_i = X((k_j \oplus h(x_i||y_i)) \times G) \oplus r_{ji}$

- **Modifying a group key**

For security reasons, each class may change its own key for a period of time. If there is a class S_i wants to change the key k_i to k_i^*, the class has to follow the under steps:

Step 1. By the helping of terminal software, the smart card of class S_i will generate the new group key k_i^* and encrypt it as $E_{SK}(k_i^*)$.

Step2. The message, $E_{SK}(k_i^*)$, will send back to TA. Then TA will recompute all related parameters associated with class S_i and encrypt these parameters with key *SK*, then TA separates all new encrypted related parameters into two half parts.

Step3. TA notices the smart card what parameters it need to update and transmit these parameters to smart card.

Step4. The smart card confirms the updates and replaces the old parameters with new parameters.

4.2.1 Security analysis

- **Hardware cracking**

For any user which has harmful intensions to system, it is the simplest way for cracking that he tries to get keys or information from smart cards. Because the smart cards are temper-resistant, we can assure that any illegal user can't obtain information from smart card.

- **interior collection attacks**

For a legal user of class S_i, he can get all of the half encrypted related parameters from TA. He sends them into his smart card and he can obtain all related parameters. If the class S_i that has m parent nodes, namely S_j, S_{j+1}…and S_{j+m}. The user may pick up related parameters r_{ji}, $r_{(j+1)i}$, …and $r_{(j+m)i}$ associated with the parents. He may perform the same action discussed in interior collection attacks of security analysis 4.1.1. Obviously, the user can obtain any secret information from these parameters.

- **exterior collection attacks**

The information which can be collected is half of encrypted related parameters like $E_{SK}(r_{ji})\_1$. Because the illegal users don't have any smart card and the key, SK and all half of related parameters like $E_{SK}(r_{ji})\_2$ are stored in smart card, he can't obtain any information for cracking. If he can get the related parameters in other ways, he still has to face harder problem than interior collection attacks discussed above.

The other attacks like collaborative attacks and sibling attacks are face the same problems discussed in security analysis 4.1.1.

5. Conclusion

In this thesis, we propose two practical hierarchy key management schemes. The first scheme is based on Lin's scheme using the advantages of elliptical curves. This scheme is successful for eliminating the drawbacks in Lin's scheme and has better performance than Lin's and Cho's schemes. The advantageous properties for dynamic key management in a hierarchy are also preserved. The two drawbacks in Lin's scheme, one is caused by that a part value of related parameter is unchanged before and after group key modification procedure. The other is caused by class ID. In our scheme, we use the properties of elliptical curves to let each class have a key pair, k_i and p_i. When TA computes the related parameters, the public keys of classes will participate in computation to make sure that each part of related parameter will change before and after group key modification procedure.

Furthermore, the public information may leak some information for cracking. We use the advantage of smart card to store and compute secret information. We add a key *SK* shared by TA and all smart card. All related parameters will be encrypted by key *SK* and be separated into two parts. One is store in smart card, the other is public. Because the related parameters are encrypted and separated, cracking parameters will be harder for any illegal user. It will make whole system be more secure.

By these two protocols, we make the key management in a user hierarchy more secure and more efficient.

Bibliography

[1] C. H. Lin and C. L. Lee: "Elliptic-curve Undeniable Signature Schemes," *Proceedings of 11th National Conference on Information Security,* May 3-4, 2001, pp.331-338.

[2] H. H. Cho, Y. H. Park, J. S. Lee, H. S. Jang, and K. H. Rhee: "A Proposal of Secure Efficient Dynamic Hierarchical Key Management Structure," *The Second Workshop on Information Security Application*, Korea, 2001, pp.357-362.

[3] N. Koblitz, A. Menezes and S. Vanstone: "The State of Elliptic curve Cryptography," *Design, Codes and Cryptography*, 19, 2000, pp.173-193.

[4] N. Y. Lee and T. Hwang: "Research Note Comments on 'Dynamic Key Management Schemes for Access Control in a Hierarchy'," *Computer Communications*, 22, 1999, pp.87-89.

[5] C. H. Lin: "Dynamic Key Management Schemes for Access Control in a Hierarchy," *Computer Communications*, 20, 1997, pp.1381-1385.

[6] R.S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," *Computer Communication* 20, 1997, pp.1381-1385.

[7] N. Koblitz. "A Course in Number Theory and Cryptography," New York, *NY:Spring-Verlag*, Second edition, 1994.

[8] A. Menezes: "Elliptic curve Public Key Cryptosystems," *Kluwer Academic Publishers*, 1993.

[9] L. Harn, L.Y. Lin,"A Cryptographic Key Generation Scheme for Multi-Level Data Security," *Computer and Security* 9, 1990, pp.539-546.

[10] N. Kobliz: "Elliptic Curve Cryptosystems," *Mathematics of Computation*, 48, 1987, pp.203-209.

[11] V. Miller: "Uses of Elliptic Curves in Cryptography," *Advances in Cryptology - Crypto' 85, Proceedings, Lecture Notes in Compute Science*, No. 218, Springer-Verlag, New York, 1985, pp.417-426.

[12] S.J. MacKinnon, P.D. Taylor, H. Meijer, S.G. Akl,"A Optimal Algorithm for Assigning Cryptographic Keys Access Control in a Hierarchy," *IEEE transaction on Computers* C34 (9), 1985, pp.797-802.

[13] S.G. Akl, P.D. Taylor,"Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. On Computer System* 1 (3), 1983, pp.239-247.

[14] E. Horowitz, S. Sahni, S. Rajasekaran, "Computer Algorithms," *Computer Science Press*, 1997

[15] Draft FIPS 180-2, Secure Hash Standard (SHS), U.S. Doc/NIST, May 30, 2001.

[16] International Organization for standards, http://www.iso.ch/

[17] PC/SC Workgroup, http://www.pcscworkgroup.com/

[18] RSA security Inc., http://www.rsasecurity.com/

[19] Open Card consortium, http://www.opencard.org/

[20] Java Card Forum, http://www.javacardforum.org/

[21] MAOSCO Ltd, http://www.multos.com/

[22] Modex web site, http://www.modex.com/

[23] SEPSCO, http://www.cepsco.com/

[24] EMVCO, http://www.emvco.com/