Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 178 (2008) 230–243
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Access control in user hierarchy based on elliptic curve cryptosystem

Yu Fang Chung [a], Hsiu Hui Lee [b], Feipei Lai [b,c], Tzer Shyong Chen [d,*]

[a] *Information Management Department, Chaoyang University of Technology, Taiwan*
[b] *Computer Science and Information Engineering Department, National Taiwan University, Taiwan*
[c] *Electrical Engineering Department, National Taiwan University, Taiwan*
[d] *Information Management Department, Tunghai University, Taiwan*

Received 16 May 2006; received in revised form 30 July 2007; accepted 1 August 2007

Abstract

This work proposes a novel key management method based on elliptic curve cryptosystem and one-way hash function to solve dynamic access problems in a user hierarchy. The proposed scheme attempts to derive the secret key of successors efficiently and non-redundantly. It includes functions such as insertion and removal of classes, updating of their relationships, and changing of secret keys. The method utilizes a Central Authority, which enables a user to change the secret key at will conveniently. Since the proposed method uses the elliptic curve cryptosystem which has a low computational cost and small key size, its performance in terms of both security and efficiency is quite commendable. Therefore, it can be anticipated that its use will be extended to wireless communication in the future.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Key management; Elliptic curve cryptosystem; Access control; User hierarchy

1. Introduction

Computer cryptography and information security are of prior concern in the digital age. The accelerated growth of computer networks and technology favors environments that support multi-users in a hierarchy. Thus, sharing resources has become unavoidable, and both academic and industrial fields require approaches to protect information from unauthorized access.

Computer communication systems often employ user hierarchies to solve access control problems. A user hierarchy generally comprises disjointed security classes, to which users and user information are assigned and ranked. The security class of a user is known as his security clearance. Assume that $SC_1, SC_2, \ldots, SC_n$ are n disjointed security classes. Let $\geqslant$ denote a binary partially ordered relationship in a user set, $SC = \{SC_1, SC_2, \ldots, SC_n\}$. In the partially ordered set $(poset, SC, \geqslant)$, $SC_i \geqslant SC_j$ denotes that the security class SC_i have

* Corresponding author. Tel.: +886 923 696355; fax: +886 2 23504930.
E-mail address: arden@thu.edu.tw (T.S. Chen).

0020-0255/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2007.08.001

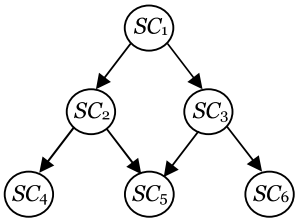

Fig. 1. *Poset* in a user hierarchy.

a security clearance higher than or equal to the security class SC_j. SC_i is classified as a predecessor of SC_j, and SC_j as a successor of SC_i. The predecessors SC_i have accessibility to information belonging to their successors SC_j, but not vice versa. If there is no security class SC_k between SC_i and SC_j, where $SC_i \geqslant SC_k \geqslant SC_j$, then SC_i is known as an immediate predecessor of SC_j, and SC_j is an immediate successor of SC_i. Fig. 1 illustrates an example of the partially ordered set in a user hierarchy. The arrows linking the connected paths in an accessible net indicate the direction of access. A user at the end of an arrow can access the user at the head of the same arrow.

For $SC_i \geqslant SC_j$, the data classified as SC_j is generally encrypted using the secret key sk_j, and SC_i derives sk_j to access the data that belongs to SC_j using sk_i. A predecessor needs to access the non-immediate successors by recursively storing the successors' secret keys level by level. Following a growing hierarchy, the users of higher-clearance security classes need larger storage space to accommodate the secret keys of all successors, resulting in key management problems. Additionally, large numbers of keys make security management difficult. Another solution to the problem is developed, which is, to assign each user a unique secret key through which a user can calculate all his successors' keys.

The cryptographic key assignment scheme is developed by Akl and Taylor [19]. In this model, each security class SC_i is given with a secret key sk_i corresponding to a public parameter T_i. For the relationship $SC_i \geqslant SC_j$, SC_i can derive the successor's secret key sk_j from his secret key sk_i and the successor's public parameter T_j. Simple key generation and derivation algorithms make the scheme superior to other solutions for dynamic access control problems. However, in practice, the number of security classes increases as the hierarchy expands, and so does the required storage space for maintaining public parameters. Therefore, the process of updating a key becomes complex and the procedure for altering secret keys becomes inconvenient.

This study presents a key management approach to overcome the above problems. The proposed method simplifies key generation and derivation algorithms, efficiently solves the dynamic access control problems, enables users to alter their secret keys at will for security reasons, and resists collusive attacks.

The rest of this paper is organized as follows. Section 2 briefly reviews previous studies on access control problems. Section 3 presents the proposed key generation and derivation algorithms. Section 4 describes dynamic key management. Section 5 analyzes the secure tolerance under such a key management scheme. Section 6 gives the analysis of performance. Conclusions are finally drawn in Section 7.

2. Review of previous research

In AT's model [19], each security class SC_i is assigned a public parameter T_i and a secret key sk_i. The secret key is created as follows:

$$sk_i = sk_0^{T_i} \pmod{M}$$

where sk_0 denotes the secret key of the Central Authority (called CA for brevity hereafter). M represents the product of a pair of secret large prime numbers, and T_i rises with the expansion of security classes. If $SC_i \geqslant SC_j$, then T_j/T_i will be an integer so that a predecessor SC_i can derive sk_j, as follows:

$$sk_j = sk_0^{T_j} = sk_0^{T_i*(T_j/T_i)} = sk_i^{(T_j/T_i)} \pmod{M}$$

If SC_j (not $\leqslant$) SC_i, then T_j/T_i will not be an integer, and the key derivation fails.

For the case where T_i rises as the security classes expand, MacKinnon et al. [20] presented a canonical assignment scheme to lower the value of T_i. Both AT's and MacKinnon's schemes utilize top-down traversal.

Harn and Lin [15] developed a bottom-up key generation method. Although these approaches succeed in decreasing the value of T_i, they have to update all existing secret keys to maintain security whenever the hierarchy changes.

Several key management schemes have been presented; for instance, the methods in [6,14,21] construct and derive the secure keys of all classes on the basis of discrete logarithm problems; the model in [17] uses the integer factorization problem to complete the construction and derivation of key. Attempts to improve dynamic access control problems were made in these schemes using different methods. These methods deal with the problems of inserting and removing security classes in a user hierarchy and the reduction of the size of public parameters. In these models, users with high security clearance apply repetitive key derivation processes to obtain the secret keys of non-immediate successors. Other methods [12,22] attempt to enhance AT's scheme, and explore other possible approaches that can enable a user in a hierarchy to modify the secret key as necessary. Accordingly, a predecessor can directly and efficiently derive the secret keys of its successors.

Lin [7] found that deriving one key from another key might compromise the new key due to the disclosure of the old secret key. Furthermore, if the identities of two users belonging to two different security classes in the hierarchy are only slightly different, then one user can probably guess the key of the other.

Kuo et al. later developed a method [10] that employs the public key to encrypt the secret key. Their model has a straightforward key assignment algorithm, and small storage space requirement. It utilizes a one-way hash function $H(X)$, where X denotes an arbitrary-length input, and $H(X)$ is a fixed-length output. The hash function is the fingerprint of a file, a message, or other data blocks, and has the following attributes [8].

(1) X can be applied to a data block of all sizes.
(2) For any given variable X, $H(X)$ is easy to operate, enabling easy implementation in software and hardware.
(3) The output length of $H(X)$ is fixed.
(4) Deriving X from the given value h and the given hash function $H(X)$ is computationally infeasible.
(5) For any given variable X, finding any $Y \neq X$ so that $H(Y) = H(X)$ is computationally infeasible.
(6) Finding an input (X, Y) so that $H(X) = H(Y)$ is computationally infeasible.

The trade-off between security and efficiency in performance means that $H(X)$ can help obtain message digest.

Based on these related works [3–5,9,13,17], this work develops a security model that provides protection against external and internal attacks. The model can provide a simple and efficient solution to overcome the collision and the security leaks.

3. Proposed scheme

The proposed method has three sequential phases, namely the relationship building phase, the key generation phase, and the key derivation phase, all of which are described below.

3.1. Elliptic curve cryptosystem

To ensure high security and efficiency, the proposed method is established based on an elliptic curve cryptosystem. An elliptic curve cryptosystem can achieve security of equal level to the RSA or DSA in the discrete logarithm problems, and it has lower computation overhead and smaller key size. The mathematic background of elliptic curve cryptosystem [1,23] is defined below.

The elliptic curve cryptosystem employs the use of elliptic curves. The variables and coefficients of elliptic curves are all restricted to elements of a finite field, offering added efficiency in the operation of ECC. Two families of elliptic curves, *prime curves* defined over Z_p and *binary curves* constructed over $GF(2^n)$, are used in cryptographic applications. Fernandes [2] once pointed out, "prime curves are best for software applications because the extended bit-fiddling operations needed by binary curves are not required; and that binary curves are best for hardware application, where it takes remarkably few logic gates to create a powerful, fast cryptosystem".

In this study, the applied elliptic curve over Z_p, defined modulo a prime$p$, is the set of solutions (x, y) to the equation, $E_p(a, b)$: $y^2 = x^3 + ax + b \pmod{p}$ where $a, b \in Z_p$, and $4a^3 + 27b^2 \bmod p \neq 0$. The condition $4a^3 + 27b^2 \bmod p \neq 0$ is necessary to ensure that $x^3 + ax + b \pmod{p}$ has no repeated factors, which means that a finite abelian group can be defined based on the set $E_p(a, b)$. The definition of an elliptic curve also includes a point at infinity denoted as O, which is the third point of intersection of any straight line with the curve; such a line has points of intersection of the form (x, y), $(x, -y)$ and O. Not any elliptic curve over Z_p can be applied in cryptographic applications. Fig. 2 [11] shows an example of the elliptic curve group, where the elliptic curve is defined by the equation, $y^2 = x^3 + x + 1 \pmod{23}$ (taken from [23]).

The example depicted in Fig. 2 has $a = 1$ and $b = 1$ so that $4a^3 + 27b^2 \bmod 23 \equiv 8 \bmod 23 \neq 0$. Thus, the elliptic group $E_{23}(1, 1)$ consists of the points shown in Table 1, extracted from [23].

Addition operation has been used over $E_p(a, b)$. For all points $P, Q \in E_p(a, b)$, the rules for addition over $E_p(a, b)$ are defined as follows:

1. $P + O = P$, where O serves as the additive identity.
2. If $P = (x_p, y_p)$, then $P + (x_p, -y_p) = O$. The point $(x_p, -y_p)$ is the negative of P, denoted as $-P$. For example, in $E_{23}(1, 1)$, for $P = (6, 4)$, we have $-P = (6, -4)$. Since $-4 \bmod 23 \equiv 19$, $-P = (6, 19)$, which is also over $E_{23}(1, 1)$.
3. If $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ with $P \neq -Q$, then $R = P + Q = (x_r, y_r)$ is over $E_{23}(1, 1)$ and is determined by the following the rules:

$$x_r = (\lambda^2 - x_p - x_q) \bmod p$$

$$y_r = (\lambda(x_p - x_r) - y_p) \bmod p$$

where λ is given as shown below

$$\lambda = \begin{cases} \left(\frac{y_q - y_p}{x_q - x_p}\right) \bmod p, & \text{if } P \neq Q \\ \left(\frac{3x_p^2 + a}{2y_p}\right) \bmod p, & \text{if } P = Q \end{cases}$$

4. Multiplication by an integer is defined by repeated addition; for example, $2P = P + P$.

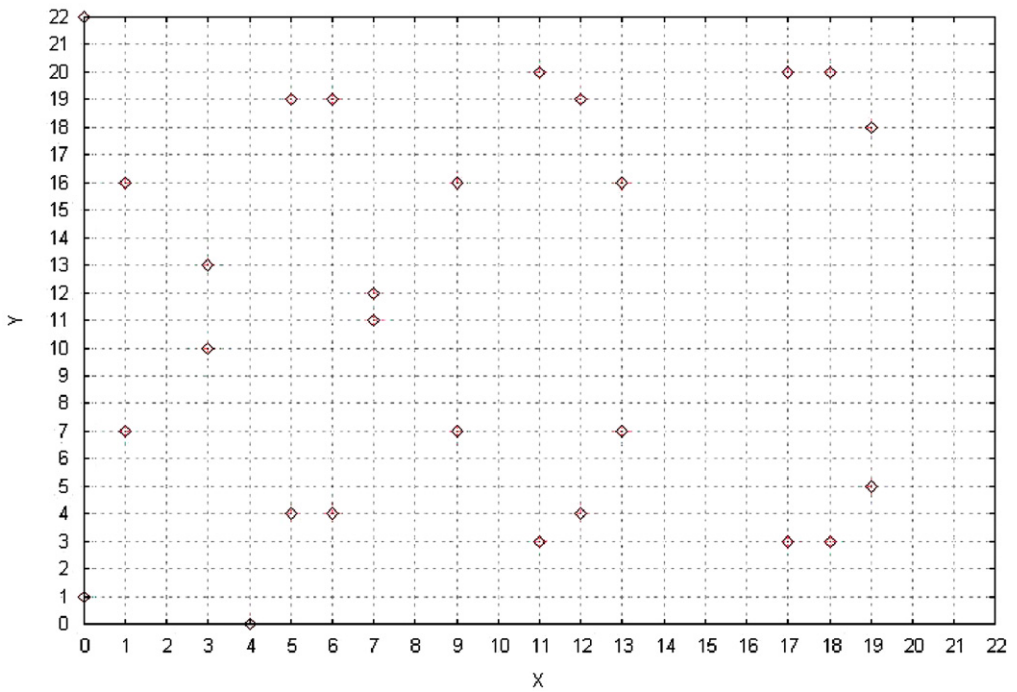

Fig. 2. Example of elliptic curve in case of $y^2 = x^3 + x + 1 \pmod{23}$.

Table 1
Points over the elliptic curve $E_{23}(1,1)$

(0,1)	(6,4)	(12,19)	(0,22)	(6,19)	(13,7)	(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)	(3,10)	(9,7)	(17,20)	(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)	(5,4)	(11,20)	(19,5)	(5,19)	(12,4)	(19,18)

Example. Let $P = (6,4)$ and $Q = (7,11)$ in $E_{23}(1,1)$. When $P \neq Q$, we must derive λ before calculating $P + Q$, as follows:

$$\lambda = \left(\frac{11-4}{7-6}\right) \bmod 23 \equiv 7$$

So, when $\lambda = 7$, x_r and y_r can be derived as shown below. Thus, $P + Q = (13,16)$.

$$x_r = (7^2 - 6 - 7) \bmod 23 \equiv -6 \bmod 23 \equiv 13$$
$$y_r = (7(6-13) - 4) \bmod 23 \equiv -53 \bmod 23 \equiv 16$$

To calculate $2P$, we must first derive λ as follows:

$$\lambda = \left(\frac{3(6^2)+1}{2 \times 4}\right) \bmod 23 \equiv \frac{109}{8} \bmod 23 \equiv 5$$

So, when $\lambda = 7$, x_r and y_r can be derived as shown below. Thus, $2P = (13,7)$.

$$x_r = (5^2 - 6 - 6) \bmod 23 \equiv 13 \bmod 23 \equiv 13$$
$$y_r = (5(6-13) - 4) \bmod 23 \equiv -39 \bmod 23 \equiv 7$$

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiplication in ECC is the counterpart of modular exponentiation in RSA. A difficult problem is essential to creating a cryptographic system using elliptic curves over Z_p. Consider the equation $Q = kP$, where $Q, P \in E_p(a,b)$ and $k < p$. Given k and P, it is relatively easy to calculate Q, but given Q and P, it is relatively hard to determine k. This is called the elliptic curve discrete logarithm problem (ECDLP) [1,16,23].

Given an example taken from [23], suppose $E_{23}(9,17)$ is an elliptic curve defined by $y^2 = x^3 + 9x + 17 \pmod{23}$. Find the discrete logarithm k of $Q = (4,5)$ to the base $P = (16,5)$. One solution is the brute-force method, in which multiples of P is computed until Q is found. Thus, $P = (16,5)$, $2P = (20, 20)$, $3P = (14, 14)$, $4P = (19,20)$, $5P = (13,10)$, $6P = (7,3)$, $7P = (8, 7)$, $8P = (12,17)$, $9P = (4,5)$.

Because $9P = (4,5) = Q$, the discrete logarithm Q to the base P is $k = 9$. But in the real implementation, the brute-force method is quite infeasible as p and k would be so large that the method would not be viable.

Apparently, the efficiency of ECC depends on the fast calculation of $Q = kP$ for some number k and a point P on the curve. The addition of elliptic curve points requires a few modular calculations. As shown in [23], ECC can have a prime p that is much smaller than the numbers in the other types of systems. This allows for significant improvement in efficiency in the operation of ECC over both integer factorization and discrete logarithm systems.

3.2. Relationship building phase

First, establish the Central Authority to specialize in system and member maintenance. In this phase, *CA* builds the hierarchical structure for controlling access according to the relationships between the nodes. Suppose there are n members which together form a set denoted as $U = \{SC_1, SC_2, \ldots, SC_n\}$. Let SC_i be a security class with higher clearance and SC_j a security class with lower clearance. If there is a legitimate relationship between SC_i and SC_j such that SC_i can access SC_j, then this relationship can be represented as $(SC_i, SC_j) \in R_{i,j}$.

3.3. Key generation phase

To complete the key generation phase, CA executes the algorithm below.

Step 1: randomly select a large prime p
Step 2: select an elliptic curve E defined over Z_p, where the order of E is located in the interval between $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$
Step 3: select a one-way function $h(x)$ to transform a point into a number and a base point G_j from $E(Z_p)$, where $j=1,\ldots,n$
Step 4: select a secret key sk_j and a sub-secret key s_j for SC_j, where $j=1,\ldots,n$
Step 5: for all $\{SC_i|(SC_i,SC_j)\} \in R_{i,j}$
determine $s_iG_j=(x_{j,i}, y_{j,i})$
determine $h(x_{j,i}\|y_{j,i})$ using the one-way hash function, where $\|$ is a bit concatenation operator
end for
Step 6: determine the public polynomial $f_j(x)$ using $h(x_{j,i}\|y_{j,i})$ as follows

$$f_j(x)=\prod_{SC_i \geqslant SC_j}[x-h(x_{j,i}\|y_{j,i})]+sk_j \bmod p$$

Step 7: send sk_j and s_j to the security class SC_j via a secret channel, and announce p, $h(x)$, G_j, and $f_j(x)$

Example. As shown in Fig. 1, the user set has six security classes, denoted as $U=\{SC_1,SC_2,SC_3,SC_4,SC_5,SC_6\}$. CA determines the public elliptic curve polynomial $f_j(x)$ for each security class. Each security class can then derive the secret keys of his successors, as follows:

$$f_j(x)=\prod_{SC_i \geqslant SC_j}[x-h(x_{j,i}\|y_{j,i})]+sk_j \bmod p$$

$$SC_1: f_1(x)=[x-h(x_{1,0}\|y_{1,0})]+sk_1 \bmod p, \quad \text{where } s_0 \text{ is given by } CA$$

$$SC_2: f_2(x)=[x-h(x_{2,1}\|y_{2,1})]+sk_2 \bmod p$$

$$SC_3: f_3(x)=[x-h(x_{3,1}\|y_{3,1})]+sk_3 \bmod p$$

$$SC_4: f_4(x)=[x-h(x_{4,1}\|y_{4,1})][x-h(x_{4,2}\|y_{4,2})]+sk_4 \bmod p$$

$$SC_5: f_5(x)=[x-h(x_{5,1}\|y_{5,1})][x-h(x_{5,2}\|y_{5,2})][x-h(x_{5,3}\|y_{5,3})]+sk_5 \bmod p$$

$$SC_6: f_6(x)=[x-h(x_{6,1}\|y_{6,1})][x-h(x_{6,3}\|y_{6,3})]+sk_6 \bmod p$$

3.4. Key derivation phase

For the relationship $(SC_i,SC_j) \in R_{i,j}$ between SC_i and SC_j, the predecessor SC_i calculates the secret keys sk_j of all successors, SC_j, as follows:

Step 1: for $\{SC_i|(SC_i,SC_j)\} \in R_{i,j}$
determine $s_iG_j=(x_{j\,i}, y_{j,i})$
determine $h(x_{j,i}\|y_{j,i})$ using the one-way hash function, where $\|$ is a bit concatenation operator
end for
Step 2: determine sk_j using $h(x_{j,i}\|y_{j,i})$ as follows

$$f_j(x)=\prod_{SC_i \geqslant SC_j}[x-h(x_{j,i}\|y_{j,i})]+sk_j \bmod p$$

$$f_j(h(x_{j,i}\|y_{j,i}))=sk_j \bmod p$$

4. Solution to key management of dynamic access problems

After establishing the protocol to generate and derive keys in a hierarchy, the solution to dynamic key management problems such as inserting a new security class, removing an existing security class, creating a new relationship, revoking an existing relationship, and changing secret keys is given as shown below.

4.1. Inserting new security classes

Assume that a new security class SC_k is inserted into the hierarchy such that $SC_i \geqslant SC_k \geqslant SC_j$; the relationship $SC_i \geqslant SC_k$ is given as $(SC_i, SC_k) \in R_{i,k}$, and the relationship $SC_k \geqslant SC_j$ is denoted as $(SC_k, SC_j) \in R_{k,j}$. CA follows the procedure below to manage the accessing priority of SC_k in the hierarchy.

Step 1: update the partial relationship R that follows when SC_k joins the hierarchy
Step 2: randomly select sk_k, s_k, and G_k
Step 3: for all $\{SC_i|(SC_i, SC_k)\} \in R_{i,k}$ that satisfies $SC_i \geqslant SC_k$ while inserting the new security class SC_k
  determine $s_iG_k = (x_{k,i}, y_{k,i})$
  determine $h(x_{k,i}\|y_{k,i})$ using the one-way hash function, where $\|$ is a bit concatenation operator
  end for
Step 4: determine the public polynomial $f_k(x)$ using $h(x_{k,i}\|y_{k,i})$ as follows

$$f_k(x) = \prod_{SC_i \geqslant SC_k} [x - h(x_{k,i}\|y_{k,i})] + sk_k \bmod p$$

Step 5: for all $\{SC_i|(SC_i, SC_k)\} \in R_{i,k}$ and $\{SC_k|(SC_k, SC_j)\} \in R_{k,j}$ that satisfy $SC_i \geqslant SC_k \geqslant SC_j$
  determine $s_kG_j = (x_{j,k}, y_{j,k})$
  determine $s_iG_j = (x_{j,i}, y_{j,i})$
  determine $h(x_{j,k}\|y_{j,k})$ and $h(x_{j,i}\|y_{j,i})$ using the one-way hash function, where $\|$ is a bit concatenation operator
  end for
Step 6: determine the public polynomial $f'_j(x)$ using $h(x_{j,k}\|y_{j,k})$ and $h(x_{j,i}\|y_{j,i})$ as follows

$$f'_j(x) = \prod_{SC_i \geqslant SC_k \geqslant SC_j} [x - h(x_{j,i}\|y_{j,i})][x - h(x_{j,k}\|y_{j,k})] + sk_j \bmod p$$

Step 7: replace $f_j(x)$ with $f'_j(x)$
Step 8: send sk_k and s_k to SC_k via a secret channel, and announce G_k, $f_k(x)$ and $f'_j(x)$

Example. In Fig. 3, a new security class SC_7 is inserted into the user hierarchy such that $SC_1 \geqslant SC_7 \geqslant SC_6$. For SC_7, CA randomly selects sk_7, s_7, and G_7. Since SC_7 is assigned as a successor to SC_1 and as a predecessor to SC_6, CA constructs the public polynomial $f_7(x)$, and replaces the public polynomial $f_6(x)$ with $f'_6(x)$. CA first calculates $h(x_{7,1}\|y_{7,1})$ with the help of the sub-secret key s_1 to construct $f_7(x)$; then determines $h(x_{6,7}\|y_{6,7})$ using the sub-secret key s_7 to derive $f'_6(x)$. Finally, CA transmits sk_7 and s_7 to SC_7 via a secret channel and announces G_7, $f_7(x)$, and $f'_6(x)$.

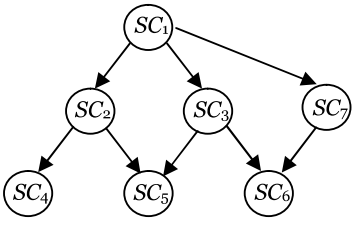

Fig. 3. The consequent *poset* after inserting SC_7.

Before SC_7 joins the hierarchy, the public polynomial $f_6(x)$ is formed as follows:

$$f_6(x) = [x - h(x_{6,1}\|y_{6,1})][x - h(x_{6,3}\|y_{6,3})] + sk_6 \bmod p$$

After SC_7 joins the hierarchy, the public polynomials $f'_6(x)$ and $f_7(x)$ are formed as follows:

$$f'_6(x) = [x - h(x_{6,1}\|y_{6,1})][x - h(x_{6,3}\|y_{6,3})][x - h(x_{6,7}\|y_{6,7})] + sk_6 \bmod p$$

$$f_7(x) = [x - h(x_{7,1}\|y_{7,1})] + sk_7 \bmod p$$

4.2. Removing existing security classes

Assume that an existing member SC_k is to be removed from a user hierarchy, such that the relationship $SC_i \geqslant SC_k \geqslant SC_j$ breaks up. *CA* not only directly revokes information related to SC_k, but also alters the accessing relationship between the involved ex-predecessor SC_i and ex-successor SC_j, of SC_k. In particular, to control the forward security of SC_j, *CA* needs to renew the secret key sk_j as sk'_j, the base point G_j as G'_j, and the public polynomial $f_j(x)$ as $f'_j(x)$, as follows:

Step 1: update the partial relationship R that follows when SC_k is removed
Step 2: for all $\{SC_k|(SC_k, SC_j)\} \in R_{k,j}$
  renew the secret key sk_j as sk'_j and the base point G_j as G'_j, of SC_j
  for all $\{SC_i|(SC_i, SC_j)\} \in R_{i,j}$
    renew $\{SC_i|(SC_i, SC_j)\} \in R_{i,j}$ after removing SC_k
    determine $s_i G'_j = (x_{j,i}, y_{j,i})$
    determine $h(x_{j,i}\|y_{j,i})$ using the one-way hash function, where $\|$ is a bit concatenation operator
  end for
  determine the public polynomial $f'_j(x)$ as follows

$$f'_j(x) = \prod_{SC_i \geqslant SC_j} [x - h(x_{j,i}\|y_{j,i})] + sk'_j \pmod p$$

  replace $f_j(x)$ with $f'_j(x)$
 end for
Step 3: send sk'_j to SC_j via a secret channel, and announce G'_j and $f'_j(x)$

Example. For instance, considering Fig. 4, let SC_3 be removed from the *poset*, so that the relationships $SC_1 \geqslant SC_3 \geqslant SC_5$ and $SC_1 \geqslant SC_3 \geqslant SC_6$ break up. To revoke the accessibility of SC_3, *CA* removes all parameters related to SC_3, and updates the relationships among the relative predecessors and successors on the connected path, such as SC_1 and SC_6. To ensure forward security of the successor SC_6, *CA* renews the secret key sk_6 as sk'_6 and the base point G_6 as G'_6. Then, *CA* identifies all predecessors of SC_6, namely SC_1 in Fig. 4; determines $h(x_{6,1}||y_{6,1})$ using the sub-secret key s_1 and G'_6, and builds the newly available polynomial $f'_6(x)$. The same procedure is then executed on other successors involved on the connected path, namely SC_5. After completing all renewals, *CA* transmits sk'_5 to SC_5 and sk'_6 to SC_6 through a secret channel and announces G'_5, G'_6, $f'_5(x)$, and $f'_6(x)$.

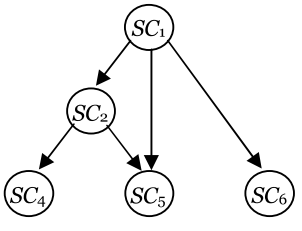

Fig. 4. The consequent *poset* after deleting SC_3.

Before deleting SC_3, $f_5(x)$ and $f_6(x)$ are formed as shown below.

$$f_5(x) = [x - h(x_{5,1}\|y_{5,1})][x - h(x_{5,2}\|y_{5,2})][x - h(x_{5,3}\|y_{5,3})] + sk_5 \bmod p$$

$$f_6(x) = [x - h(x_{6,1}\|y_{6,1})][x - h_(x_{6,3}\|y_{6,3})] + sk_6 \bmod p$$

After deleting SC_3, $f'_5(x)$ and $f'_6(x)$ are formed as shown below.

$$f'_5(x) = [x - h(x_{5,1}\|y_{5,1})][x - h(x_{5,2}\|y_{5,2})] + sk_{5'} \bmod p$$

$$f'_6(x) = [x - h(x_{6,1}\|y_{6,1})] + sk'_6 \bmod p$$

4.3. Creating new relationships

The relationships among members in an organization might be changeable. For instance, a new relationship $SC_k \geqslant SC_l$ might be added such that $SC_i \geqslant SC_k \geqslant SC_l \geqslant SC_j$. Notably, the relationship between SC_k and SC_l is immediate. *CA* performs the following procedure to link the relationships between SC_l and his predecessors (SC_k, SC_i), and the relationships between SC_j and his predecessors (SC_l, SC_k, SC_i).

Step 1: save the partial relationship $SC_i \geqslant SC_k \geqslant SC_l \geqslant SC_j$ formed due to the creation of $SC_k \geqslant SC_l$
Step 2: for all $SC_i \geqslant SC_l$
  if $\{SC_i|(SC_i, SC_l)\} \in R_{i,l}$ does not hold until $SC_k \geqslant SC_l$ is created such that $SC_i \geqslant SC_k \geqslant SC_l \geqslant SC_j$
    determine $s_iG_l = (x_{l,i}, y_{l,i})$
    determine $s_kG_l = (x_{l,k}, y_{l,k})$
    determine $h(x_{l,i}\|y_{l,i})$ and $h(x_{l,k}\|y_{l,k})$ using the one-way hash function, where $\|$ is a bit concatenation operator
  end if
end for
Step 3: determine the public polynomial $f_l(x)$ as follows

$$f_l(x) = \prod_{SC_i \geqslant SC_l} [x - h(x_{l,i}\|y_{l,i})][x - h(x_{l,k}\|y_{l,k})] + sk_l \bmod p$$

Step 4: for all $SC_i \geqslant SC_j$
  if $\{SC_i|(SC_i, SC_j)\} \in R_{i,j}$ do not hold until $SC_k \geqslant SC_l$ is created such that $SC_i \geqslant SC_k \geqslant SC_l \geqslant SC_j$
    for all $\{SC_i|(SC_i, SC_j)\} \in R_{i,j}$
      determine $s_iG_j = (x_{j,i}, y_{j,i})$
      determine $s_kG_j = (x_{j,k}, y_{j,k})$
      determine $s_lG_j = (x_{j,l}, y_{j,l})$
      determine $h(x_{j,i}\|y_{j,i})$, $h(x_{j,k}\|y_{j,k})$, and $h(x_{j,l}\|y_{j,l})$, where $\|$ is a bit concatenation operator
    end for
  end if
end for
Step 5: determine the public polynomial $f'_j(x)$ as follows

$$f'_j(x) = \prod_{SC_i \geqslant SC_j} [x - h(x_{j,i}\|y_{j,i})][x - h(x_{j,k}\|y_{j,k})][x - h(x_{j,l}\|y_{j,l})] + sk_j \bmod p$$

Step 6: replace $f_j(x)$ with $f'_j(x)$
Step 7: announce $f_l(x)$ and $f'_j(x)$

Example. Fig. 5 displays the creation of a relationship between SC_5 and SC_6 such that $SC_2 \geqslant SC_5 \geqslant SC_6$, making SC_5 a new predecessor of SC_6. To authorize access to SC_5 from SC_6, CA calculates $h(x_{6,5}\|y_{6,5})$ with s_5 and $h(x_{6,2}\|y_{6,2})$ with s_2 to build the public polynomial $f_6(x)$ using previously obtained parameters $h(x_{6,1}\|y_{6,1})$ and $h(x_{6,3}\|y_{6,3})$.

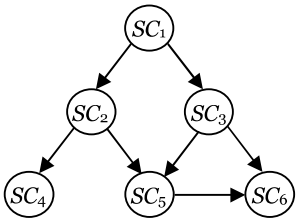

Fig. 5. The consequent *poset* after creating $SC_5 \geqslant SC_6$.

Before creating the relationship $SC_2 \geqslant SC_5 \geqslant SC_6$, $f_6(x)$ is formed as follows:

$$f_6(x) = [x - h(x_{6,1}\|y_{6,1})][x - h(x_{6,3}\|y_{6,3})] + sk_6 \bmod p$$

After creating the relationship $SC_2 \geqslant SC_5 \geqslant SC_6$, $f'_6(x)$ is formed as follows:

$$f'_6(x) = [x - h(x_{6,1}\|y_{6,1})][x - h(x_{6,3}\|y_{6,3})][x - h(x_{6,2}\|y_{6,2})][x - h(x_{6,5}\|y_{6,5})] + sk'_6 \bmod p$$

4.4. Revoking existing relationships

Consider the case of revoking an existing relationship $(SC_k, SC_l) \in R_{k,l}$. In addition to directly deleting the relationship, *CA* updates the accessibility of SC_k over SC_l for controlling the forward security of the ex-successor SC_l. Restated, *CA* renews the secret key sk_l as sk'_l, the base point G_l as G'_l, and $f_l(x)$ as $f'_l(x)$, related to SC_l. *CA* follows the following procedure to revoke an existing relationship.

Step 1: revoke the partial relationship R due to the deletion of $(SC_k, SC_l) \in R_{k,l}$
Step 2: renew the secret key sk_l as sk'_l and the base point G_l as G'_l, related to SC_l
Step 3: for all $\{SC_i|(SC_i, SC_l)\} \in R_{i,l}$ that holds after revoking $(SC_k, SC_l) \in R_{k,l}$
  determine $s_i G'_l = (x_{l,i}, y_{l,i})$
  determine $h(x_{l,i}\|y_{l,i})$ using the one-way hash function, where || is a bit concatenation operator
 end for
Step 4: determine the public polynomial $f'_l(x)$ using $h(x_{l,i}\|y_{l,i})$ as follows

$$f'_l(x) = \prod_{SC_i \geqslant SC_l} [x - h(x_{l,i}\|y_{l,i})] + sk'_l \bmod p$$

Step 5: for all $\{SC_k|(SC_k, SC_j)\} \in R_{k,j}$
  if $\{SC_k|(SC_k, SC_j)\} \in R_{k,j}$ breaks up after revoking $(SC_k, SC_l) \in R_{k,l}$
   renew the secret key sk_j as sk'_j and the base point G_l as G'_l, related to SC_j
    for all $\{SC_i|(SC_i, SC_j)\} \in R_{i,j}$
     determine $s_i G'_j = (x_{j,i}, y_{j,i})$
     determine $s_l G'_j = (x_{j,l}, y_{j,l})$
     determine $h(x_{j,i}\|y_{j,i})$ and $h(x_{j,l}\|y_{j,l})$, where $\|$ is a bit concatenation operator
    end for
   renew the public polynomial $f'_j(x)$ as follows

$$f'_j(x) = \prod_{SC_i \geqslant SC_j} [x - h(x_{j,i}\|y_{j,i})][x - h(x_{j,l}\|y_{j,l})] + sk'_j \bmod p$$

  end if
 end for
Step 6: send the sk'_l to SC_l and sk'_j to SC_j via a secret channel, and announce $f'_l(x)$, $f'_j(x)$, G_l, and G'_l

Example. Consider the revoking of relationship $\{SC_2|(SC_2, SC_5)\} \in R_{2,5}$ in Fig. 6, such that $\{SC_2|(SC_2, SC_5)\} \notin R_{2,5}$.

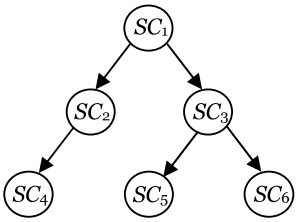

Fig. 6. The consequent *poset* after revoking $SC_2 \geqslant SC_5$.

Because $\{SC_2|(SC_2, SC_5)\} \in R_{2,5}$ does not hold, CA renews the secret key sk_5 as sk'_5, the base point G_5 as G'_5, and the public polynomial $f_5(x)$ as $f'_5(x)$, related to SC_5.

Before revoking $\{SC_2|(SC_2, SC_5)\} \in R_{2,5}$, $f_5(x)$ is formed as follows:

$$f_5(x) = [x - h(x_{5,1}\|y_{5,1})][x - h(x_{5,2}\|y_{5,2})][x - h(x_{5,3}\|y_{5,3})] + sk_5 \bmod p$$

After revoking $\{SC_2|(SC_2, SC_5)\} \in R_{2,5}$, $f_5(x)$ is replaced with $f'_5(x)$ as follows:

$$f'_5(x) = [x - h(x_{5,1}\|y_{5,1})][x - h(x_{5,3}\|y_{5,3})] + sk'_5 \bmod p$$

4.5. Changing secret keys

A secret key must be changeable to maximize security. To change a secret key sk_j to sk'_j, CA must replace the base point G_j with G'_j and the public polynomial $f_j(x)$ with $f'_j(x)$, as follows:

Step 1: replace the secret key sk_j with sk'_j and the base point G_j with G'_j
Step 2: for all $\{SC_i|(SC_i, SC_j)\} \in R_{i,j}$
  determine $s_i G'_j = (x_{j,i}, y_{j,i})$
  determine $h(x_{j,i}\|y_{j,i})$, where $\|$ is a bit concatenation operator
  end for
Step 3: determine the public polynomial $f'_j(x)$ as follows

$$f'_j(x) = \prod_{SC_i \geqslant SC_j} [x - h(x_{j,i}\|y_{j,i})] + sk'_j \bmod p$$

Step 4: replace $f_j(x)$ with $f'_j(x)$
Step 5: send sk'_j to SC_j via a secret channel, and announce G'_j and $f'_j(x)$

5. Discussion of security

This section addresses the possible types of attacks. Security tolerance of the proposed model in response to the various attacks is discussed in the following subsections.

5.1. Contrary attack

The first potential attack is from a successor, who might wish to obtain the secret key of the immediate or any prior predecessor through the public parameters and his own secret key. That is, can a successor SC_j compute the predecessor's secret key from the public polynomial $f_i(x)$ and the one-way hash function $h(x_{j,i}\|y_{j,i})$? The unauthorized user can generally solve this problem by the given plaintext. However, both the elliptic curve cryptosystem and the one-way hash function can resist forced attack in the proposed approach because their time complexity is placed at reasonable computational security. An unauthorized successor cannot obtain the secret key even after years of attempting. Hence, the proposed scheme is highly secure against such an attack.

5.2. Exterior collecting attack

The second potential attack is from an outsider. Can an intruder generate the secret key from a lower security class by accessible public parameters? In addition to deriving both the elliptic curve cryptosystem and the one-way hash function, the invader must successfully launch a ciphertext attack against the asymmetric cryptosystem. A ciphertext attack against an asymmetric cryptosystem is much harder than a plaintext attack against an asymmetric cryptosystem. Therefore, the proposed model resists intrusion from outsiders.

5.3. Collaborative attack

The collaborative attack is a type of attack where several users collaborate to launch the attack. Suppose SC_j and SC_k are the immediate successors of SC_i; their relationship can be denoted as $(SC_i, SC_j) \in R_{i,j}$ and $(SC_i, SC_k) \in R_{i,k}$, as shown in Fig. 7.

When SC_j and SC_k collaborate to try to hack the secret key sk_i of SC_i, first, SC_j and SC_k must exchange secret keys with each other, and then derive the sub-secret key s_i of SC_i through $f_j(x)$ and $f_k(x)$.

$$f_j(x) = \prod_{SC_i \geqslant SC_j} [x - h(x_{j,i} \| y_{j,i})] + sk_j \bmod p$$

$$f_k(x) = \prod_{SC_i \geqslant SC_k} [x - h(x_{k,i} \| y_{k,i})] + sk_k \bmod p$$

However, s_i is protected by the one-way hash function and the ECDLP among which one-way hash function is irreversible while the ECDLP is computationally extremely complex. Therefore, attackers cannot invert the procedure to derive s_i.

5.4. Equation attack

This is a type of attack where a member uses the common successor to try to hack the secret key of another member it does not have an accessibility relationship with, like those shown in Fig. 8. For the relationships $SC_i \geqslant SC_j$ and $SC_k \geqslant SC_j$, SC_i may try to obtain the sub-secret key s_k of SC_k through $f_j(x)$.

Taking Fig. 1 as example, aimed at the relationships $SC_2 \geqslant SC_5$ and $SC_3 \geqslant SC_5$, SC_2 may attempt to obtain s_3 through their common successor SC_5. Using $s_1 G_5 = (x_{5,1}, y_{5,1})$, $s_2 G_5 = (x_{5,2}, y_{5,2})$, and $s_3 G_5 = (x_{5,3}, y_{5,3})$, $f_5(x)$ can be formed as follows:

$$f_5(x) = [x - h(x_{5,1} \| y_{5,1})][x - h(x_{5,2} \| y_{5,2})][x - h(x_{5,3} \| y_{5,3})] + sk_5 \bmod p$$

$$f_5(x) - sk_5 = [x - h(x_{5,1} \| y_{5,1})][x - h(x_{5,2} \| y_{5,2})][x - h(x_{5,3} \| y_{5,3})] \bmod p$$

$$x - h(x_{5,3} \| y_{5,3}) = [f_5(x) - sk_5] / [x - h(x_{5,1} \| y_{5,1})][x - h(x_{5,2} \| y_{5,2})] \bmod p$$

$$\text{Let } x = 0, \text{ then } h(x_{5,3} \| y_{5,3}) = [sk_5 - f_5(0)] / [h(x_{5,1} \| y_{5,1})][h(x_{5,2} \| y_{5,2})] \bmod p$$

The derivation of hacking s_3 from $f_5(x)$ is based on the difficulty of solving one-way hash function and ECDLP, which is of reasonable computational security.

5.5. Forward security of the successors while changing $SC_i \geqslant SC_k \geqslant SC_j$ to $SC_i \geqslant SC_j$

Modifying the relationship $SC_i \geqslant SC_k \geqslant SC_j$ to $SC_i \geqslant SC_j$ annuls the accessibility authority of SC_k over SC_j. The forward security of the existing security class SC_j should be considered seriously. *CA* not only deletes

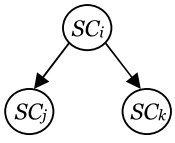

Fig. 7. Relationships potentially risking a collaborative attack.

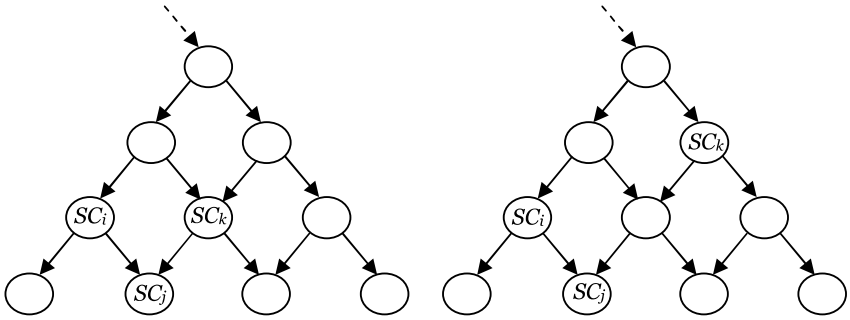

Fig. 8. Relationships potentially risking an equation attack.

the accessibility-link relationship, but also updates the accessibility-link relationship between SC_i and SC_j. CA replaces the secret key sk_j with sk'_j and the base point G_j with G'_j, and thus computes the renewed public polynomial $f'_j(x)$ which no longer includes the factor $h(x_{k,i}\|y_{k,i})$. The authority of SC_k over SC_j is thus terminated, so SC_k cannot later determine the secret key sk_j of SC_j.

6. Analysis of performance

Table 2 analyzes the proposed approach in comparison to other methods in terms of the required complexity for processing dynamic access control problems, in which Chang denotes the model in [6], Wu represents that in [14], and Hwang is that in [17]. The analysis in Table 2 clearly reveals that the proposed method is more straightforward than the other ones, and also requires less storage space.

Table 2 shows functional comparisons between the presented scheme and other previously proposed ones. In terms of storage size, both Hwang's and Wu's schemes require large storage space. In these schemes, the number of public parameters and the length of the public parameters grow as the number of successor increases, so the required storage space too gets increasingly larger. As for Chang's scheme, it is similar to the scheme proposed in this study. Each class has only one fixed public parameter that needs to be stored. In terms of dynamic access control problem, all four schemes need only make partial update to information when inserting and deleting security classes, creating and revoking relationship, and changing security key.

On computational complexity, key generation and key derivation requires executing elliptic curve addition operations, hash operations and constructing interpolating polynomials. In terms of computational overheads, Vanstone [18] had summarized that the key sizes and bandwidth required by ECC provides higher efficiency with order of magnitude roughly 10 times that of integer factorization systems and discrete logarithm systems. Besides, Stallings [23] estimated that the 4096-bit key size of the RSA gives the same level of security as the 313-bit one in ECC. That is, the length of the prime p in $E_p(a,b)$ is secure enough with 300 bits.

The storage required for the polynomials $f_i(x)$ is proportional to the number of successors a security class is assigned. The length of the prime p is 300 bits such that the coefficients of the polynomial are defined over p. Let m be the degree of $f_i(x)$; then the storage occupies about $m\lceil \log p + 1 \rceil$ bits. In integer factorization systems or discrete logarithm systems, the chosen prime should be of at least 100 decimal digits to provide sufficient security.

Table 2
Performance analysis in terms of complexity for access control problems

Required complexity	Chang	Wu	Hwang	The proposal
Key generation	Exponential	Exponential	Factorization	ECC + hash + encryption
Key derivation	Exponential	Exponential	Factorization	ECC + hash + decryption
Inserting/removing security classes	Partial update	Partial update	Partial update	Partial update
Creating/revoking relationships	Partial update	Partial update	Partial update	Partial update
Changing secret keys	Partial update	Partial update	Partial update	Partial update
Storage for public parameters	Fixed and small	Large	Large	Fixed and small

As to constructing an interpolating polynomial, Knuth [8] completed it with a computation time of $O(m(\log m)^2)$. The overall computational complexity of establishing and updating the polynomials is $O(n \cdot m(\log m)^2)$, where n is the number of security classes in the hierarchy.

7. Conclusions

The proposed key management method for controlling dynamic access problems is a simple and efficient solution for ensuring hierarchical organization. It allows the access of members to data to be classified according to their ranks. Members in higher-ranked security class can directly access the secret keys of members in lower-ranked classes, but not vice versa. The members can change the secret keys at will in consideration of security, showing that the key generation and public polynomial are flexible.

References

[1] A. Cilardo, L. Coppolino, N. Mazzocca, L. Romano, Elliptic curve cryptography engineering, Proceedings of the IEEE 94 (2) (2006) 395–406.
[2] A.D. Fernandes, Elliptic-curve cryptography, Dr. Dobb's Journal (1999).
[3] A.D. Santis, A.L. Ferrara, B. Masucci, A new key assignment scheme for access control in a complete tree hierarchy, in: Proceeding of the International Workshop on Coding and Cryptography—WCC 2005, LNCS 3969, 2006, pp. 202–217.
[4] A.D. Santis, A.L. Ferrara, B. Masucci, Cryptographic key assignment schemes for any access control policy, Information Processing Letters 92 (4) (2004) 199–205, Nov.
[5] A.D. Santis, A.L. Ferrara, B. Masucci, Enforcing the security of a time-bound hierarchical key assignment scheme, Information Sciences 176 (12) (2006) 1684–1694, June.
[6] C.C. Chang, I.C. Lin, H.M. Tsai, H.H. Wang, A key assignment scheme for controlling access in partially ordered user hierarchies, in: Proceedings of the 18th IEEE International Conference on Advanced Information Networking and Applications (AINA2004), Fukuoka, Japan, vol. 2, March 2004, pp. 376–379.
[7] C.H. Lin, Dynamic key management schemes for access control in a hierarchy, Computer Communications 20 (15) (1997) 1381–1385.
[8] D.E. Knuth, 3rd ed., The Art of Computer Programming, vol.2: Seminumerical Algorithms, Addison-Wesley, Reading, MA, 1998.
[9] F.G. Jeng, C.M. Wang, An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem, Journal of Systems and Software 79 (8) (2006) 1161–1167, Oct.
[10] F.H. Kuo, V.R.L. Shen, T.S. Chen, F. Lai, Cryptographic key assignment scheme for dynamic access control in a user hierarchy, IEE Proceeding—Computers and Digital Techniques 146 (5) (1999) 235–240.
[11] Francisco Rodríguez-Henríquez, Doctoral Dissertation: New Algorithms and Architectures for Arithmetic in GF(2^m) Suitable for Elliptic Curve Cryptography, Oregon EUA, June 2000. Supervisor: Dr. Cetin K. Koc. Available from: <http://delta.cs.cinvestav.mx/~francisco/tesis.html>.
[12] H.M. Tsai, C.C. Chang, A cryptographic implementation for dynamic access control in a user hierarchy, Computers and Security 14 (2) (1995) 159–166.
[13] J.H. Yeh, R. Chow, R. Newman, Key assignment for enforcing access control policy exceptions in distributed systems, Information Sciences 152 (2003) 63–88.
[14] J. Wu, R. Wei, An access control scheme for partially ordered set hierarchy with provable security, in: Proceedings of SAC 2005, LNCS 3897, 2006, pp. 221–232.
[15] L. Harn, H.Y. Lin, A cryptographic key generation scheme for multilevel data security, Computers and Security 9 (6) (1990) 539–546.
[16] M.A. Strangio, Efficient Diffie-Hellmann two-party key agreement protocols based on elliptic curves, in: Proceedings of the 2005 ACM Symposium on Applied Computing, 2005, pp. 324–331.
[17] M.S. Hwang, W.P. Yang, Controlling access in large partially-ordered hierarchies using cryptographic keys, Journal of Systems and Software 67 (2) (2003) 99–107.
[18] S.A. Vanstone, Elliptic curve cryptosystem—The answer to strong, fast public-key cryptography for securing constrained environments, Information Security Technical Report 2 (2) (1997) 78–87.
[19] S.G. Akl, P.D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, ACM Transactions on Computer Systems 1 (3) (1983) 239–248.
[20] S.J. MacKinnon, P.D. Taylor, H. Meijer, S.G. Akl, An optimal algorithm for assigning cryptographic keys to control access in a hierarchy, IEEE Transactions on Computers 34 (9) (1985) 797–802.
[21] V.R.L. Shen, T.S. Chen, A novel key management scheme based on discrete logarithms and polynomial interpolations, Computers and Security 21 (2) (2002) 164–171.
[22] V.R.L. Shen, T.S. Chen, F. Lai, Novel cryptographic key assignment scheme for dynamic access control in a hierarchy, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E80-A (10) (1997) 2035–2037.
[23] W. Stallings, Cryptography and Network Security: Principles and Practice, fourth ed., Prentice Hall, 2005.