Information Sciences 209 (2012) 80–92

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem

Ashok Kumar Das [a,*], Nayan Ranjan Paul [b], Laxminath Tripathy [c]

[a] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India
[b] Department of Computer Science, KMBB College of Engineering and Technology, Khurda 752 056, India
[c] Department of Information Technology, Eastern Academy of Science and Technology, Bhubaneswar 754 001, India

ARTICLE INFO

Article history:
Received 20 November 2010
Received in revised form 28 December 2011
Accepted 25 April 2012
Available online 9 May 2012

Keywords:
Key management
Elliptic curve
Hierarchical access control
Polynomial interpolation
Security
Exterior root finding attacks

ABSTRACT

In a key management scheme for hierarchy based access control, each security class having higher clearance can derive the cryptographic secret keys of its other security classes having lower clearances. In 2008, Chung et al. proposed an efficient scheme on access control in user hierarchy based on elliptic curve cryptosystem [Information Sciences 178 (1) (2008) 230–243]. Their scheme provides solution of key management efficiently for dynamic access problems. However, in this paper, we propose an attack on Chung et al.'s scheme to show that Chung et al.'s scheme is insecure against the exterior root finding attack. We show that under this attack, an attacker (adversary) who is not a user in any security class in a user hierarchy attempts to derive the secret key of a security class by using the root finding algorithm. In order to remedy this attack, we further propose a simple improvement on Chung et al.'s scheme. Overall, the main theme of this paper is very simple: a security flaw is presented on Chung et al.'s scheme and then a fix is provided in order to remedy the security flaw found in Chung et al.'s scheme.

1. Introduction

Hierarchical access control is a fundamental problem in computer and network systems. In a hierarchical access control, a user of higher security level class has the ability to access information items (such as message, data, and files) of other users of lower security classes. A user hierarchy consists of a number n of disjoint security classes, say, $SC_1, SC_2, \ldots, SC_n$. Let this set be $SC = \{SC_1, SC_2, \ldots, SC_n\}$. A binary partially ordered relation $\geqslant$ is defined in SC as $SC_i \geqslant SC_j$, which means that the security class SC_i has a security clearance higher than or equal to the security class SC_j. In addition the relation $\geqslant$ satisfies the following properties:

(a) [Reflexive property] $SC_i \geqslant SC_i, \forall SC_i \in SC$.
(b) [Anti-symmetric property] If $SC_i, SC_j \in SC$ such that $SC_i \geqslant SC_j$ and $SC_j \geqslant SC_i$, then $SC_i = SC_j$.
(c) [Transitive property] If $SC_i, SC_j, SC_k \in SC$ such that $SC_i \geqslant SC_j$ and $SC_j \geqslant SC_k$, then $SC_i \geqslant SC_k$.

If $SC_i \geqslant SC_j$, we call SC_i as the predecessor of SC_j and SC_j as the successor of SC_i. If $SC_i \geqslant SC_k \geqslant SC_j$, then SC_k is an intermediate security class. In this case SC_k is the predecessor of SC_j and SC_i is the predecessor of SC_k. In a user hierarchy, the encrypted message by a successor security class is only decrypted by that successor class as well as its all predecessor security classes in that hierarchy.

* Corresponding author. Tel.: +91 40 6653 1506; fax: +91 40 6653 1413.
E-mail addresses: iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in (A.K. Das), nayan.p@kmbb.in (N.R. Paul), laxmintripathy@gmail.com (L. Tripathy).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.04.036

Akl and Taylor [1] first developed the cryptographic key assignment scheme in an arbitrary partial order set (poset) hierarchy. MacKinnon et al. [2] presented an optimal algorithm, called the canonical assignment, to reduce the value of public parameters. Harn and Lin [3] then proposed a bottom up key generating scheme, instead of using a top-down approach as in the Akl and Taylor scheme and MacKinnon et al.'s scheme.

In order to solve dynamic access control problems, many schemes have been proposed in the literature [4–13]. Chang et al. [8] proposed a key assignment scheme based on Lagrange's interpolation method and one-way hash function. In their scheme, a user with higher security clearance must iteratively perform the key derivation process for deriving the secret key of a user who is not an immediate successor. Other proposed schemes [7,10] enhance Akl and Taylor's scheme [1], and explore other possible approaches that can enable a user in a hierarchy to modify the secret key as and when necessary. Thus, a predecessor can directly and efficiently derive the secret keys of its successor(s). Kuo et al. later developed a method [6] that employs the public key to encrypt the secret key. Their scheme has a straightforward key assignment algorithm, small storage space requirement, and uses a one-way hash function.

Shen and Chen proposed a novel key management scheme based on discrete logarithms and polynomial interpolations in a user hierarchy [9]. However, Hsu and Wu [12] presented an attack, called the exterior root finding attack, on Shen-Chen's scheme [9] so that an attacker can derive the encryption key of a user in the hierarchy. In [12], authors showed that some malicious insider, for example a user in a security class, can have access to the information items held by those who are not his/her subordinates. They further proposed an improvement to amend the flaws in Shen-Chen's scheme.

Chen and Huang proposed an efficient novel key management scheme for dynamic access control in a user hierarchy [13]. Their scheme is based on the efficiencies of one-way hash function and symmetric-key encryptions and decryptions. Further, their scheme supports dynamic access control including adding new security classes in the hierarchy, deleting existing security classes from the hierarchy, adding new relationships in the hierarchy, deleting existing relationships from the hierachchy as well as changing secret keys of security classes in the hierarchy. The performance of their scheme is also efficient compared to Akl-Taylor's scheme [1], Kuo et al.'s scheme [6], and Lin's scheme [4].

In 2008, Chung et al. [11] proposed an efficient key management and derivation scheme based on the elliptic curve cryptosystem. In their scheme, the secret key of each security class can be determined by a trusted centralized authority (CA). An attractive advantage of their scheme is that it solves dynamic key management efficiently and flexibly. However, we show that their scheme is vulnerable to exterior root finding attacks.

In this paper, we propose an exterior root finding attack on Chung et al.'s scheme [11] to show that their scheme is vulnerable under this proposed attack. Our attack on Chung et al.'s scheme is similar to Hsu-Wu's exterior root finding attack on Shen-Chen's novel key management scheme. In our exterior root finding attack on Chung et al.'s scheme, an attacker (adversary) who is not a user in any security class in a user hierarchy can derive the secret key of a security class by using the root finding algorithm. In order to eliminate this security flaw in their scheme, we further propose a simple improvement on their scheme. Hence, the theme of this paper is very simple: a security flaw is presented on Chung et al.'s scheme and then a fix is provided to remedy the security flaw found in Chung et al.'s scheme.

The rest of this paper is sketched as follows. In Section 2, we review some mathematical background which are useful to review Chung et al.'s scheme. We then give briefly an overview of Chung et al. scheme [11] in Section 3. In Section 4, we describe our proposed exterior root finding attack on Chung et al.'s scheme [11]. In Section 5, we propose a simple improvement on Chung et al.'s scheme to remedy the attack proposed in Section 4 and discuss security of our improved scheme. We provide performance comparison of our improved scheme with Chung et al.'s scheme and Chen-Haung's scheme in Section 6. Finally, we conclude the paper in Section 7.

2. Mathematical background

In this section, we discuss the elliptic curve and its properties. We then discuss the rules for adding points on elliptic curve and the elliptic curve discrete logarithm problem. We, finally, discuss the properties of a one-way hash function and M. Ben-or's method [14] for root finding and factorization of polynomials in finite field.

2.1. Elliptic curve over finite field

Let a and $b \in Z_p$, where $Z_p = \{0, 1, \ldots, p-1\}$ and $p > 3$ be a prime, such that $4a^3 + 27b^2 \neq 0 \pmod p$. A non-singular elliptic curve $y^2 = x^3 + ax + b$ over the finite field $GF(p)$ is the set $E_p(a, b)$ of solutions $(x, y) \in Z_p \times Z_p$ to the congruence

$$y^2 = x^3 + ax + b \pmod p,$$

where a and $b \in Z_p$ are constants such that $4a^3 + 27b^2 \neq 0 \pmod p$, together with a special point $\mathcal{O}$ called the point at infinity or zero point.

The condition $4a^3 + 27b^2 \neq 0 \pmod p$ is the necessary and sufficient to ensure that the equation $x^3 + ax + b = 0$ has a non-singular solution [15]. If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be points in $E_p(a, b)$, then $P + Q = \mathcal{O}$ implies that $x_Q = x_P$ and $y_Q = -y_P$. Also, $P + \mathcal{O} = \mathcal{O} + P = P$, for all $P \in E_p(a, b)$. Moreover, an elliptic curve $E_p(a, b)$ over Z_p has roughly p points on it. More precisely, a well-known theorem due to Hasse asserts that the number of points on $E_p(a, b)$, which is denoted by $\#E$, satisfies the following inequality [16]:

$$p + 1 - 2\sqrt{p} \leqslant \#E \leqslant p + 1 + 2\sqrt{p}.$$

In addition, $E_p(a, b)$ forms an abelian group or commutative group under addition modulo p operation.

2.1.1. Addition of points on elliptic curve over finite field

We require the following parameters over the elliptic curve domain. We take an elliptic curve over a finite filed $GF(p)$ as $E_p(a, b)$: $y^2 = x^3 + ax + b \pmod p$, where a and $b \in Z_p^*$. The field size p is considered as a large prime. We take G as the base point on $E_p(a, b)$ whose order is n, that is, $nG = G + G + \ldots + G(n\ times) = \mathcal{O}$.

The elliptic curve addition differs from the general addition [17]. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on elliptic curve $y^2 = x^3 + ax + b \pmod p$, with $P \neq -Q$, then $R = (x_3, y_3) = P + Q$ is computed as follows:

$$x_3 = (\lambda^2 - x_1 - x_2)(\text{mod } p),$$
$$y_3 = (\lambda(x_1 - x_3) - y_1)(\text{mod } p),$$

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

2.1.2. Point multiplication on elliptic curve over finite field

In elliptic curve cryptography, multiplication is defined as repeated additions. For example, if $P \in E_p(a, b)$, then $6P$ is computed as $6P = P + P + P + P + P + P \pmod p$.

2.2. Discrete logarithm problem

The discrete logarithm problem (DLP) is as follows: given an element g in a finite group G whose order is n, that is, $n = \# G_g$ (G_g is the subgroup of G generated by g) and another element h in G_g, find the smallest non-negative integer x such that $g^x = h$. It is relatively easy to calculate discrete exponentiation $g^x \pmod n$ given g,x and n, but it is computationally infeasible to determine x given h,g and n, when n is large.

2.3. Elliptic curve discrete logarithm problem

Let $E_p(a, b)$ be an elliptic curve modulo a prime p. Given two points $P \in E_p(a, b)$ and $Q = kP \in E_p(a, b)$, for some positive integer k. $Q = kP$ represents the point P on elliptic curve $E_p(a, b)$ is added to itself k times. The elliptic curve discrete logarithm problem (ECDLP) is to determine k given P and Q. It is relatively easy to calculate Q given k and P, but it is computationally infeasible to determine k given Q and P, when the prime p is large.

2.4. One-way hash function

A one-way hash function h: $\{0, 1\}^* \rightarrow \{0, 1\}^l$ takes an arbitrary-length input $X \in \{0, 1\}^*$, and produces a fixed-length (say, l-bits) output $h(X) \in \{0, 1\}^l$, called the message digest. The hash function is the fingerprint of a file, a message, or other data blocks, and has the following attributes [16].

(i) h can be applied to a data block of all sizes.
(ii) For any given variable X, $h(X)$ is easy to operate, enabling easy implementation in software and hardware.
(iii) The output length of $h(X)$ is fixed.
(iv) Deriving X from the given value $Y = h(X)$ and the given hash function $h(\cdot)$ is computationally infeasible.
(v) For any given variable X, finding any $Y \neq X$ so that $h(Y) = h(X)$ is computationally infeasible.
(vi) Finding a pair of inputs (X, Y) with $X \neq Y$, so that $h(X) = h(Y)$ is computationally infeasible.

2.5. Root finding and factorization of polynomials in finite field

In 1981, M. Ben-or [14] proposed an efficient probabilistic algorithm for finding all the roots of a given polynomial $f(x) \in GF(q)[x]$, where n is the degree of $f(x)$, in $GF(q)$. He also proposed another efficient probabilistic algorithm for factoring a given polynomial $f(x) \in GF(q)[x]$, with degree n of $f(x)$, in $GF(q)$. In this section, we discuss these methods for the convenience of our cryptanalysis and improvement of Chung et al.'s scheme.

2.5.1. Root finding of polynomial in finite field GF(q)

Suppose we are given a polynomial $f(x) \in GF(q)[x]$ of degree $n(deg \cdot f = n)$, in $GF(q)$ and we want to find all the roots $a \in GF(q)$ of $f(x) = 0$. Let q be odd.

M. Ben-or's method [14] for finding all the roots of $f(x) = 0$ is as follows. Compute the greatest common divisor (gcd) as $f_1(x) = \text{gcd}\ (f(x), x^q - x)$. If $f_1(x) = 1$, then $f(x)$ has no roots in $GF(q)$. Otherwise, in general, we have,

$f_1(x) = (x - a_1)(x - a_2) \ldots (x - a_k)$, $k \leqslant n$, where a_i's are all the pairwise distinct roots of $f(x) = 0$ in $GF(q)$. The probabilistic algorithm for finding the roots of $f_1(x) \in GF(q)[x]$ is given in Algorithm 1 [14]:

Algorithm 1. Finding roots of $f_1(x) \in GF(q)[x]$:$ROOTS(f_1(x))$

```
if ($deg \cdot f_1(x) = 1$, i.e., $f_1(x) = (x - a)$) then
  return $a$;
end if
repeat
  Choose $\delta \in GF(q)$ randomly;
  Compute $f_2(x) = \gcd[f_1(x), (x + \delta)^{(q-1)/2} - 1]$;
until ($0 < deg \cdot f_2 < deg \cdot f_1$)
return $ROOTS(f_2(x)) \cup ROOTS(f_1(x)/f_2(x))$; {$\cup$ is the set union operator.}
```

M. Ben-or showed that the expected number of operations used by the above algorithm (Algorithm 1) for finding all roots of $f(x) = 0$ is $O(n \cdot \log n \cdot L(n) \cdot \log q)$, where $L(n) = \log n \cdot \log \log n$. In other words, the expected running time of the Algorithm 1 is polynomial in n.

2.5.2. Factorization of polynomial in finite field GF(q)

Let we want to factor a polynomial $f(x) \in GF(q)[x]$ of degree n into its irreducible factors. Assume that $f(x)$ has no repeated factors and we want to factor $f(x)$ as $f(x) = g_1(x) \cdot g_2(x) \ldots g_n(x)$, where $g_d(x)$ is the product of irreducible factors of $f(x)$ of degree d.

M. Ben-or proposed the following algorithm [14] for computing factors of the polynomial $f(x)$ (Algorithm 2). Further, M. Ben-or discussed how to factor $g_i(x)$ into irreducible factors of equal degree. He proved that the expected number of operations used by the algorithm (Algorithm 2) to factor an n-degree polynomial is $O(n^2 \cdot L(n) \cdot \log q)$. That is, the expected running time of the Algorithm 2 is also polynomial in n.

Algorithm 2. Factorization

```
$r_0(x) = x$;
$f_0(x) = f(x)$;
for $d = 1 \rightarrow n$ do
  $r_d(x) = (r_{d-1}(x))^q \pmod{f_{d-1}(x)}$, $deg \cdot r_d < deg \cdot f_{d-1}$;
  $g_d(x) = \gcd\ (f_{d-1}(x), r_d(x) - x)$;
  $f_d(x) = f_{d-1}(x)/g_d(x)$;
end for
return $g_1(x), g_2(x), \ldots, g_n(x)$;
```

3. Overview of Chung et al.'s scheme

Chung et al.'s scheme [11] consists of three phases, namely, the relationship building phase, the key generation phase, and the key derivation phase. In the relationship building phase, a central authority (CA) builds the hierarchical structure for controlling access according to the relationship between the nodes. In the key generation phase, CA chooses a pair of points and constructs the public polynomial using a one-way hash function. In the key derivation phase, the predecessor in the hierarchy can use its own secret key and the public information related to the successor(s) to derive the decryption key(s) for the successor(s) for accessing the authorized file(s). Further, Chung et al. provided the solution of key management of dynamic access problems by means of providing inserting a new security class, removing an existing security class, creating a new relationship, revoking an existing relationship, and changing secret key of a security class in the hierarchy. We discuss the followings for Chung et al.'s scheme in the following subsections.

3.1. Relationship building phase

In this phase, CA builds the hierarchical structure for controlling access according to the relationships among the nodes in the hierarchy. Let $U = \{SC_1, SC_2, \ldots, SC_n\}$ be a set of n security classes in the hierarchy. Assume that SC_i is a security class with higher clearance and SC_j a security class with lower clearance, that is, $SC_i \geqslant SC_j$. A legitimate relationship $(SC_i, SC_j) \in R_{i,j}$ between two security classes SC_i and SC_j exists in the hierarchy if SC_i can access SC_j.

3.2. Key generation phase

In this phase, CA performs the following steps:

Step 1: Selects randomly a large prime p, an elliptic curve $E_p(a,b)$ defined over Z_p such that the order of $E_p(a,b)$ lies in the interval $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$, and a one-way function $h(\cdot)$ to transform a point into a number and a base point G_j from $E_p(a,b)$, $1 \leqslant j \leqslant n$. Then, for each security class $SC_j (1 \leqslant j \leqslant n)$, selects a secret key sk_j and a sub-secret key s_j.
Step 2: For all $\{SC_i | (SC_i, SC_j) \in R_{i,j}\}$, computes $s_i G_j = (x_{j,i}, y_{j,i})$, and $h(x_{j,i} \| y_{j,i})$, where $\|$ is a bit concatenation operator.
Step 3: Finally, computes the public polynomial $f_j(x)$ using the values of $h(x_{j,i} \| y_{j,i})$ as

$$f_j(x) = \prod_{SC_i > SC_j} (x - h(x_{j,i} \| y_{j,i})) + sk_j \pmod p$$

Step 4: Sends sk_j and s_j to the security class SC_j via a secret channel, and announces $p, h(\cdot), G_j, f_j(x)$ as public.

3.3. Key derivation phase

In order to compute the secret keys sk_j of all successors, SC_j, the predecessor SC_i, for which the relationships $(SC_i, SC_j) \in R_{i,j}$ between SC_i and SC_j hold, proceeds as follows:

Step 1: For $\{SC_i | (SC_i, SC_j) \in R_{i,j}\}$, computes $s_i G_j = (x_{j,i}, y_{j,i})$, and $h(x_{j,i} \| y_{j,i})$.
Step 2: Computes the secret key sk_j using $h(x_{j,i} \| y_{j,i})$ as follows:

$$f_j(x) = \prod_{SC_i > SC_j} (x - h(x_{j,i} \| y_{j,i})) + sk_j \pmod p,$$
$$f_j(h(x_{j,i} \| y_{j,i})) = sk_j \pmod p.$$

3.4. Inserting new security classes phase

If a new security class SC_k is inserted into the hierarchy such that $SC_i \geqslant SC_k \geqslant SC_j$, then the relationships $(SC_i, SC_k) \in R_{i,k}$ for $SC_i \geqslant SC_k$ and $(SC_k, SC_j) \in R_{k,j}$ for $SC_k \geqslant SC_j$ need to be updated into the hierarchy. CA needs the following steps to manage the accessing priority of SC_k in the hierarchy.

Step 1: Updates the partial relationships R that follow when the security class SC_k joins the hierarchy, and randomly selects the secret key sk_k, the sub-secret key s_k and the base point G_k for the class SC_k.
Step 2: For all $\{SC_i | (SC_i, SC_k) \in R_{i,k}\}$ that satisfies $SC_i \geqslant SC_k$ when the new class SC_k is inserted in the hierarchy, computes $s_i G_k = (x_{k,i}, y_{k,i})$, and $h(x_{k,i} \| y_{k,i})$.
Step 3: Computes the public polynomial $f_k(x)$ as follows:

$$f_k(x) = \prod_{SC_i > SC_k} (x - h(x_{k,i} \| y_{k,i})) + sk_k \pmod p$$

Step 4: For all $\{SC_i | (SC_i, SC_k) \in R_{i,k}\}$ and $\{SC_j | (SC_k, SC_j) \in R_{k,j}\}$ that satisfy $SC_i \geqslant SC_k \geqslant SC_j$ when the new class SC_k is inserted in the hierarchy, computes $s_k G_j = (x_{j,k}, y_{j,k})$, $s_i G_j = (x_{j,i}, y_{j,i})$, $h(x_{j,k} \| y_{j,k})$ and $h(x_{j,i} \| y_{j,i})$.
Step 5: Computes the public polynomial $f'_j(x)$ as follows:

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

Step 6: Replaces $f_j(x)$ with $f'_j(x)$, and sends sk_k and s_k to SC_k via a secure channel, and announces publicly $G_k, f_k(x)$ and $f'_j(x)$.

3.5. Removing existing security classes phase

If an existing member SC_k, such that the relationship $SC_i \geqslant SC_k \geqslant SC_j$ breaks up, wants to leave from a user hierarchy, then 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 this phase, CA executes the following steps.

Step 1: Updates the partial relationship R that follows when SC_k is removed.
Step 2: For all $\{SC_k | (SC_k, SC_j) \in R_{k,j}\}$ does the followings:
  Renews the secret key sk_j as sk'_j and the base point G_j as G'_j of SC_j.
  For all $\{SC_i | (SC_i, SC_j) \in R_{i,j}\}$ does the followings:
    Renews $\{SC_i | (SC_i, SC_j) \in R_{i,j}\}$ after removing SC_k.
    Computes $s_i G'_j = (x_{j,i}, y_{j,i})$, and $h(x_{j,i}, y_{j,i})$.

Computes the public polynomial $f_j'(x)$ as

$$f_j'(x) = \prod_{SC_i > SC_j} (x - h(x_{j,i} \| y_{j,i})) + sk_j' \pmod p$$

Replaces $f_j(x)$ with $f_j'(x)$.

Step 3: Sends sk_j' to SC_j via a secret channel and announces G_j' and $f_j'(x)$ as public.

4. Cryptanalysis of Chung et al.'s scheme

In this section, we present an exterior root finding attack on Chung et al.'s scheme to show that their scheme is insecure against this attack. Note that Hsu-Wu's exterior root finding attack on Shen-Chen's scheme requires some malicious insider, for example a user in a security class, who can have access to the information items held by those who are not his/her subordinates. However, in our exterior root finding attack, any attacker (he/she needs not to be any malicious insider in the user hierarchy as in Hsu-Wu's attack), who is not a user in any security class in the user hierarchy, can easily derive the secret key of a security class.

4.1. The exterior root finding attack on Chung et al.'s scheme

In exterior root finding attack, an attacker (adversary) who is not a user in any security class in a user hierarchy attempts to derive the secret key of a security class by using the root finding algorithm. Note that in key generation algorithm of Chung et al.'s scheme (see Section3.2), for each security class SC_i, CA generates the base point G_i, the sub-secret key s_i and the secret key sk_i, determines the public polynomial $f_i(x)$ and then securely sends sk_i to SC_i, and announces publicly p,$h(\cdot)$,G_i and $f_i(x)$. In the construction of the public polynomial $f_j(x)$ of a security class SC_j, the sub-secret parameters of its all predecessors are embedded in its public polynomial $f_j(x)$.

Consider the case when a security class SC_k is inserted in the user hierarchy with the relationship $SC_i \geqslant SC_k \geqslant SC_j$. Thus, when a new security class SC_k is added as a predecessor of SC_j, CA updates the public polynomial of SC_j by replacing $f_j(x)$ by $f_j'(x)$ (see Section 3.4). However, for those predecessors, which remain as predecessors of SC_j in $f_j'(x)$, their secrets are still at the same positions of $f_j'(x)$. Now, knowing the public polynomial $f_j(x)$ of SC_j before adding the security class SC_k and the public polynomial $f_j'(x)$ of SC_j after adding the security class SC_k until the secret key sk_j of SC_j has been changed by CA, an attacker can generate a polynomial by taking the difference of $f_j(x)$ and $f_j'(x)$. Let this difference polynomial be denoted by $\phi(x) = f_j(x) - f_j'(x)$. It is noted that

$$\begin{aligned}\phi(x) = f_j(x) - f_j'(x) &= \left(\prod_{SC_i > SC_j} [x - h(x_{j,i} \| y_{j,i})] + sk_j \pmod p\right) - \left(\prod_{SC_i > SC_k > SC_j} [x - h(x_{j,i} \| y_{j,i})][x - h(x_{j,k} \| y_{j,k})] + sk_j \pmod p\right) \\ &= \prod_{SC_i > SC_j} [x - h(x_{j,i} \| y_{j,i})] - \prod_{SC_i > SC_k > SC_j} [x - h(x_{j,i} \| y_{j,i})][x - h(x_{j,k} \| y_{j,k})] \pmod p\end{aligned}$$

Further, we observe that the constructed polynomial $\phi(x)$ has common factors $(x - h(x_{j,i} \| y_{j,i}))$. Then the attacker finds the roots of the equation $\phi(x) = f_j(x) - f_j'(x) = 0$ in a polynomial time using [14,18]. With the knowledge of the roots, the attacker can easily derive the secret key sk_j of the security class SC_j. The attacker, who is not a user in hierarchy of security classes, first obtains the roots $h(x_{j,i} \| y_{j,i})$ and then computes the secret key sk_j of SC_j as $sk_j = f_j(h(x_{j,i} \| y_{j,i})) = f_j'(h(x_{j,i} \| y_{j,i})) \pmod p$. This clearly shows that Chung et al.'s scheme is vulnerable to our proposed exterior root finding attack.

4.2. An example

For simplicity, we give the following simple example in order to demonstrate that our proposed exterior root finding attack is effective in attacking on Chung et al.'s scheme. As shown in Fig. 1, the user hierarchy consists of six security classes, denoted by $U = \{SC_1, SC_2, SC_3, SC_4, SC_5, SC_6\}$. CA computes the public elliptic curve polynomial $f_j(x)$ for each security class SC_j. Each security class SC_i then derives the secret keys of its successors SC_j, using the key generation algorithm as follows:

$$\begin{aligned}&f_j(x) = \prod_{SC_i > SC_j} [x - h(x_{j,i} \| y_{j,i})] + sk_j \pmod p, \\ &SC_1 : f_1(x) = [x - h(x_{1,0} \| y_{1,0})] + sk_1 \pmod p, \quad \text{where } s_0 \text{ is given by CA} \\ &SC_2 : f_2(x) = [x - h(x_{2,1} \| y_{2,1})] + sk_2 \pmod p, \\ &SC_3 : f_3(x) = [x - h(x_{3,1} \| y_{3,1})] + sk_3 \pmod p, \\ &SC_4 : f_4(x) = [x - h(x_{4,1} \| y_{4,1})][x - h(x_{4,2} \| y_{4,2})] + sk_4 \pmod p, \\ &SC_5 : f_5(x) = [x - h(x_{5,1} \| y_{5,1})][x - h(x_{5,2} \| y_{5,2})][x - h(x_{5,3} \| y_{5,3})] + sk_5 \pmod p, \\ &SC_6 : f_6(x) = [x - h(x_{6,1} \| y_{6,1})][x - h(x_{6,3} \| y_{6,3})] + sk_6 \pmod p\end{aligned}$$

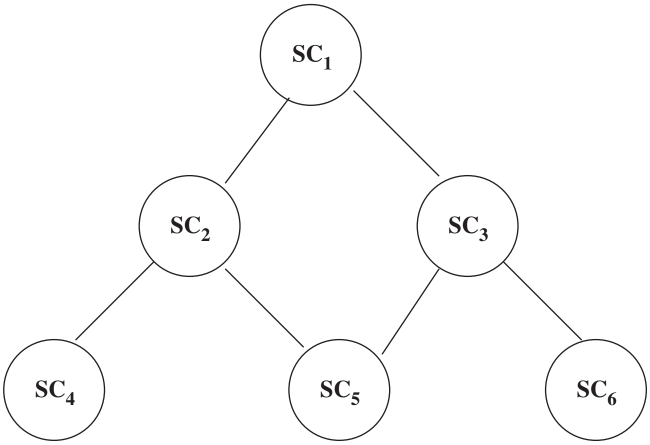

Fig. 1. A small sample of poset in a user hierarchy.

We now consider the case that a new security class SC_7 is inserted into the existing user hierarchy shown in Fig. 1, with the relationship $SC_1 \geqslant SC_7 \geqslant SC_6$. The resulting user hierarchy is shown in Fig. 2. Due to insertion of SC_7, CA needs to select randomly sk_7, s_7 and G_7. Since SC_7 is a successor of SC_1 and a predecessor of SC_6, CA constructs the public polynomial $f_7(x)$ and replaces the public polynomial $f_6(x)$ with $f_6'(x)$, using the phase for inserting new security classes described in Section 3.4. We note that before joining the security class SC_7 into the hierarchy, the public elliptic curve polynomial for security class SC_6 was

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

After joining the security class SC_7, the public polynomial $f_6'(x)$ for SC_6 and $f_7(x)$ for SC_7 are formed as follows:

$$f_6'(x) = [x - h(x_{6,1}\|y_{6,1})][x - h(x_{6,3}\|y_{6,3})][x - h(x_{6,7}\|y_{6,7})] + sk_6 \pmod p \tag{2}$$

$$f_7(x) = [x - h(x_{7,1}\|y_{7,1})] + sk_7 \pmod p \tag{3}$$

Now, knowing the public polynomials $f_6(x)$ and $f_6'(x)$ in Eqs. (1) and (2), an attacker finds the roots of the equation:

$$\phi(x) = f_6(x) - f_6'(x) = 0 \Rightarrow [x - h(x_{6,1}\|y_{6,1})][x - h(x_{6,3}\|y_{6,3})][1 - (x - h(x_{6,7}\|y_{6,7}))] = 0 \pmod p \tag{4}$$

Solving the Eq. (4) in polynomial time, the attacker obtains the roots as $x = h(x_{6,1}\|y_{6,1}), h(x_{6,3}\|y_{6,3})$ and $1 + h(x_{6,7}\|y_{6,7})$. Out of these roots, $h(x_{6,1}\|y_{6,1})$ and $h(x_{6,3}\|y_{6,3})$ satisfy both Eqs. (1) and (2). Thus, knowing these values, the attacker easily computes the secret key sk_6 of security class SC_6 as

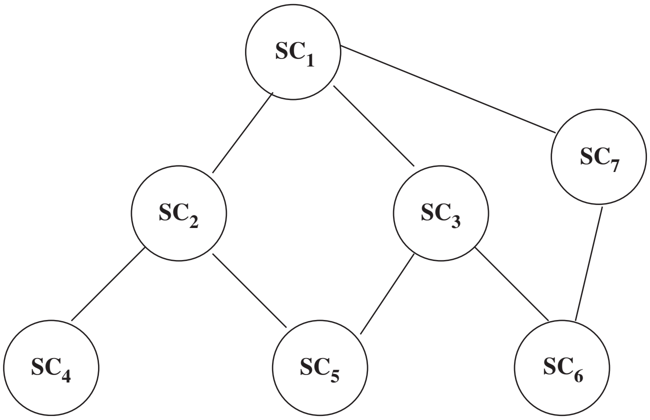

Fig. 2. A small sample of poset in a user hierarchy: when a new security class SC_7 is added into the hierarchy.

$$\begin{aligned}sk_6 &= f_6(h(x_{6,1}\|y_{6,1}))\pmod p\\ &= f'_6(h(x_{6,1}\|y_{6,1}))\pmod p\\ &= f_6(h(x_{6,3}\|y_{6,3}))\pmod p\\ &= f'_6(h(x_{6,3}\|y_{6,3}))\pmod p.\end{aligned}$$

5. Improvement on Chung et al.'s scheme

Our proposed exterior root finding attack shows that Chung et al.'s scheme is insecure against this type of attack. In order to remedy this weakness of Chung et al.'s scheme, we propose a simple improvement on Chung et al.'s scheme.

5.1. Description of the improved scheme

We only give the improved versions of the phases, namely inserting new security classes for key management in dynamic access problems and key derivation for deriving the secret key of successor security classes of a predecessor security class. The remaining phases, such as the relationship building phase, key generation phase, removing an existing security class, creating a new relationship, revoking an existing relationship, and changing secret keys of a security class in the hierarchy, remain same as in Chung et al.'s scheme.

- **Inserting new security classes:** If a new security class SC_k is inserted into the hierarchy such that $SC_i \geqslant SC_k \geqslant SC_j$, then the relationships $(SC_i, SC_k) \in R_{i,k}$ for $SC_i \geqslant SC_k$ and $(SC_k, SC_j) \in R_{k,j}$ for $SC_k \geqslant SC_j$ need to be updated into the hierarchy. In this phase, CA renews the secret keys sk_j of successors SC_j of the newly added security class SC_k. Moreover, CA must change the public base points G_j by G'_j and the public elliptic curve polynomials $f_j(x)$ with $f'_j(x)$ of SC_j. CA needs the following steps to manage the accessing priority of SC_k in the hierarchy.

Step 1: Updates the partial relationships R that follow when the security class SC_k joins the hierarchy.
Step 2: Randomly selects the secret key sk_k, the sub-secret key s_k and the base point G_k for the class SC_k.
Step 3: For all $\{SC_i|(SC_i, SC_k) \in R_{i,k}\}$ that satisfies $SC_i \geqslant SC_k$ when the new class SC_k is inserted in the hierarchy, computes $s_iG_k = (x_{k,i}, y_{k,i})$, and $h(x_{k,i}\|y_{k,i})$.
Step 4: Computes the public polynomial $f_k(x)$ as follows:

$$f_k(x) = \prod_{SC_i > SC_k}(x - h(x_{k,i}\|y_{k,i})) + sk_k \pmod p$$

Step 5: For all $\{SC_i|(SC_i, SC_k) \in R_{i,k}\}$ and $\{SC_j|(SC_k, SC_j) \in R_{k,j}\}$ that satisfy $SC_i \geqslant SC_k \geqslant SC_j$ when the new class SC_k is inserted in the hierarchy:
Replaces the secret key sk_j with sk'_j and the base point G_j with G'_j of the successor security class SC_j of SC_k.
Computes $s_kG'_j = \left(x'_{j,k}, y'_{j,k}\right)$.
Computes $s_iG'_j = \left(x'_{j,i}, y'_{j,i}\right)$.
Computes $h\left(x'_{j,k}\|y'_{j,k}\right)$ and $h\left(x'_{j,i}\|y'_{j,i}\right)$ using the one-way function $h(\cdot)$.
Step 6: Computes the public polynomial $f'_j(x)$ as follows:

$$f'_j(x) = \prod_{SC_i > SC_k > SC_j}\left(x - h\left(x'_{j,i}\|y'_{j,i}\right)\right)\left(x - h\left(x'_{j,k}\|y'_{j,k}\right)\right) + sk'_j \pmod p$$

Step 7: Replaces $f_j(x)$ with $f'_j(x)$, and sends sk'_j to SC_j via a secure channel, and announces publicly G'_j and $f'_j(x)$.
Step 8: Sends sk_k and s_k to SC_k via a secure channel, and announces publicly G_k and $f_k(x)$.

- **Key derivation:** For the relationship $(SC_i,SC_j) \in R_{i,j}$ between two security classes SC_i and SC_j, if the predecessor SC_i wants to compute the updated secret key sk'_j of its successor SC_j, then SC_i needs to proceed the following steps:

Step 1: For $\{SC_i|(SC_i, SC_j) \in R_{i,j}\}$, computes $s_iG'_j = \left(x'_{j,i}, y'_{j,i}\right)$ and $h\left(x'_{j,i}\|y'_{j,i}\right)$.
Step 2: Computes the secret key sk'_j using the computed hash value $h\left(x'_{j,i}\|y'_{j,i}\right)$ as follows:
SC_i has the updated public elliptic curve polynomial for SC_j as

$$f'_j(x) = \prod_{SC_i > SC_k > SC_j}\left(x - h\left(x'_{j,i}\|y'_{j,i}\right)\right)\left(x - h\left(x'_{j,k}\|y'_{j,k}\right)\right) + sk'_j \pmod p.$$

Computes

$$sk'_j = f'_j\left(h\left(x'_{j,i}\|y'_{j,i}\right)\right) \pmod p.$$

5.2. Security analysis of the improved scheme

In this section, we discuss the security tolerances of our improved scheme with respect to the following attacks.

5.2.1. Exterior root finding attack

Suppose an attacker who is not a user in any security class in a user hierarchy attempts to derive the secret key of a security class SC_j by using the root finding algorithm. The attacker knows the public polynomial $f_j(x)$ for SC_j. When CA adds a new security class SC_k in the existing user hierarchy with the relationship $SC_i \geqslant SC_k \geqslant SC_j$, then in our improved scheme CA updates the secret key sk_j with sk_j' and the base point G_j with G_j' for the security class SC_j and also announces the public updated elliptic curve polynomial $f_j'(x)$. It is noted that the points and secret key sk_j used in construction of $f_j(x)$ are different from the points and secret key sk_j' constructed in $f_j'(x)$. Now, knowing the public polynomial $f_j'(x)$, the attacker can try to find the roots of the equation $\phi(x) = f_j(x) - f_j'(x) = 0$ in the polynomial time. However, the roots of $\phi(x) = 0$ will not help in finding the secret key sk_j' from either $f_j(x)$ or $f_j'(x)$, and as a result, the attacker cannot derive the secret key sk_j' for the security class SC_j. Hence, our improved scheme can resist against exterior root finding attacks.

Remark. An efficient way to improve the *inserting new security classes phase* of the proposed scheme can be as follows. In Step 5 of the inserting new security classes phase (Section 5.1), for all $\{SC_i | (SC_i, SC_k) \in R_{i,k}\}$ and $\{SC_j | (SC_k, SC_j) \in R_{k,j}\}$ that satisfy $SC_i \geqslant SC_k \geqslant SC_j$ when the new class SC_k is inserted in the hierarchy, CA may simply replace the secret key sk_j with sk_j' without renewing the base point G_j of the successor security class SC_j of SC_k. In this case, we have, $s_kG_j = (x_{j,k}, y_{j,k})$ and $s_iG_j = (x_{j,i}, y_{j,i})$. CA may compute the updated new public polynomial $f_j'(x)$ for the security class SC_j as follows:

$$f_j'(x) = \prod_{SC_i > SC_k > SC_j} (x - h(x_{j,i} \| y_{j,i}))(x - h(x_{j,k} \| y_{j,k})) + sk_j' \pmod p$$

It is easy to see that an attacker cannot find the updated secret key sk_j' knowing the public polynomials $f_j(x)$ of SC_j before adding SC_k and $f_j'(x)$ of SC_j after adding SC_k into the user hierarchy with $SC_i \geqslant SC_k \geqslant SC_j$, using the root finding attack.

However, in the following we show in this case that the attacker can still obtain a relationship between the old secret key sk_j and the new secret key sk_j' with the help of public polynomials $f_j(x)$ and $f_j'(x)$. For simplicity, we take $f_j(x) = (x - a_1) + sk_j \pmod p$, where $a_1 = h(x_{j,i} \| y_{j,i})$, and $f_j'(x) = (x - a_1)(x - a_2) + sk_j' \pmod p$, where $a_2 = h(x_{j,k} \| y_{j,k})$. Rewriting $f_j(x)$ and $f_j'(x)$, we have,

$$f_j(x) = x + u \pmod p, \tag{5}$$

where

$$u = sk_j - a_1, \tag{6}$$

and

$$f_j'(x) = x^2 + vx + w \pmod p, \tag{7}$$

where

$$v = -(a_1 + a_2), \tag{8}$$

$$w = a_1a_2 + sk_j'. \tag{9}$$

Since an attacker knows the public polynomials $f_j(x)$ and $f_j'(x)$, so the coefficients u,v and w are known to the attacker. From Eq. (8), we have, $a_2 = -v - a_1 = -(v + sk_j - u)$. As a result, from Eq. (9), we obtain $sk_j' = w - a_1a_2 = sk_j^2 - (u - v)sk_j + (uv + w)$, that is, $sk_j' = g(sk_j)$, where $g(sk_j) = sk_j^2 - (u - v)sk_j + (uv + w)$. In this case, it is observed that if the secret key sk_j is simply replaced by the renewed secret key sk_j' in the updated public polynomial $f_j'(x)$, then an attacker may establish a relationship between the old secret key sk_j and updated secret key sk_j'. If the attacker does not have any ability to know the old secret key sk_j, it is computationally infeasible for an attacker to obtain the new secret key sk_j'. But, it is desirable to change the previous factors of the old polynomial $f_j(x)$ along with the secret key sk_j embedded in it, into the updated public polynomial $f_j'(x)$. Hence, in order to achieve maximum security in our improved scheme described in Section 5.1, we have also changed the base point G_j of SC_j so that there will be no repeated factors of $f_j(x)$ in $f_j'(x)$, and the attacker does not have any scope to form a relationship between the old secret key and new secret key.

5.2.2. Contrary attack

Let SC_j be a successor security class of the predecessor class SC_i in the user hierarchy. In this attack, the question is whether SC_j can compute SC_i's secret key sk_i from the public elliptic curve polynomial $f_i(x)$ and the hash value $h(x_{j,i} \| y_{j,i})$. However, it is computationally hard for the attacker to obtain sk_i due to the ECDLP and one-way hash function properties, and thus the proposed scheme is secure against this attack.

5.2.3. Exterior collecting attack

This type of attack is from an outsider, where an intruder can generate the secret key from a lower security class by accessible public parameters. As in Chung et al.'s scheme, the improved scheme also resists intrusion from outsiders.

5.2.4. Collaborative attack

This attack analysis of the improved scheme is also similar to that of Chung et al.'s scheme. In this type of attack, several users can collaborate to launch the attack. For example, if SC_j and SC_k be two immediate successors of SC_i with $(SC_i, SC_j) \in R_{i,j}$ and $(SC_i, SC_k) \in R_{i,k}$, both SC_j and SC_k can collaborate to hack the secret key sk_i of SC_i and then try to derive the sub-secret key s_i of SC_i from $f_j(x)$ and $f_k(x)$. However, this problem is computationally infeasible due to the ECDLP and one-way hash function properties, and as a result, the improved scheme also resists this attack.

5.2.5. Equation attack

In this attack, a member in the user hierarchy may use the common successor to hack the secret key of another member in the hierarchy such that it does not have an accessibility relationship with that member. The analysis of this attack is similar to that in Chung et al.'s scheme. Hence, the improved scheme has also ability to resist this type of attack.

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

When the relationship $SC_i \geqslant SC_k \geqslant SC_j$ is modified to $SC_i \geqslant SC_j$ due to removable of the security class SC_k in the hierarchy, then CA not only deletes the accessibility relationship $SC_k \geqslant SC_j$, but also updates the accessibility-link relationship between SC_i and SC_j. In this case, CA replaces the secret key sk_j of SC_j with renewed secret key sk'_j and also the base point G_j with G'_j. As a result, the computed renewed public elliptic curve polynomial $f'_j(x)$ does not include the previous factor $h(x_{k,i}\|y_{k,i})$, and the authority of SC_k over SC_j is terminated so that SC_k cannot have any ability to determine later the secret key sk'_j of SC_j. The forward security of the existing security class SC_j is then retained.

5.3. Formal security proof of the proposed scheme

In this section, we provide the formal security proof of our improved scheme.

Definition 1 (*Formal definition of ECDLP*). We define the elliptic curve discrete logarithm problem (ECDLP) formally as in [19]. Given $E_p(a, b)$ be an elliptic curve modulo a prime p. Let $P \in E_p(a, b)$ and $Q = kP \in E_p(a, b)$ be two points, where $k \in {}_R Z_p$. (We use the notation $a \in {}_R S$ to denote that a is chosen randomly from the set S.)

Instance: (P, Q, r) for some $k, r \in {}_R Z_p$.
Output: **yes**, if $Q = rP$, i.e., $k = r$, and output **no**, otherwise.

We now consider the following two distributions

$$\Delta_{real} = \{k \in_R Z_p, A = P, B = Q(= kP), C = k : (A, B, C)\},$$
$$\Delta_{rand} = \{k, r \in_R Z_p, A = P, B = Q(= kP), C = r : (A, B, C)\}.$$

Then the advantage of any probabilistic, polynomial-time, 0/1-valued (false/true-valued) distinguisher $\mathcal{D}$ in solving ECDLP on $E_p(a, b)$ is defined as

$$Adv^{ECDLP}_{\mathcal{D},E_p(a,b)} = |Pr[(A, B, C) \leftarrow \Delta_{real} : \mathcal{D}(A, B, C) = 1] - Pr[(A, B, C) \leftarrow \Delta_{rand} : \mathcal{D}(A, B, C) = 1]|,$$

where the probability $Pr[\cdot]$ is taken over the random choices of k and r. We call $\mathcal{D}$ to be a (t, ϵ)-ECDLP distinguisher for $E_p(a, b)$ if $\mathcal{D}$ runs at most in time t such that $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}(t) \geqslant \epsilon$.

ECDLP assumption: There exists no (t, ϵ)-ECDLP distinguisher for $E_p(a,b)$. In other words, for every probabilistic, polynomial-time 0/1-valued distinguisher $\mathcal{D}$, we have $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)}(t) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$.

Theorem 1. *Under the elliptic curve discrete logarithm problem assumption, our improved scheme is provably secure against an adversary for computing the secret key of a security class in the user hierarchy.*

Proof. Let $f_j(x) = \prod_{SC_i > SC_j}(x - h(x_{j,i}\|y_{j,i})) + sk_j \pmod p$ be the public elliptic curve polynomial of SC_j. Note that $f_j(x)$ can be also the polynomial after inserting new security classes into the hierarchy. Let $a_{j,i} = h(x_{j,i}\|y_{j,i})$. In this proof, we need to construct an adversary $\mathcal{A}$ who can find or guess the value $a_{j,i}$ correctly from public elliptic curve parameters and $f_j(x)$ such that the secret key sk_j can be computed as $sk_j = f_j(a_{j,i}) \pmod p = f_j(h(x_{j,i}\|y_{j,i})) \pmod p$.

We define the following two random oracles for the adversary $\mathcal{A}$:

- *Reveal:* This unconditionally outputs one of values $h(x_{j,i}\|y_{j,i})$, say $a'_{j,i}$ from the public elliptic curve polynomial $f_j(x)$ and also a secret key sk for the security class SC_j.
- *Test:* This query is allowed at any time during the attacker's execution. This oracle computes the attacker's ability to distinguish between the correct secret key and a fake secret key for SC_j by the following test:

Accept sk as the correct secret key sk_j of SC_j, if $f_j(a'_{j,i}) = sk \pmod p$;
Reject sk as the correct secret key sk_j of SC_j, otherwise.

The adversary $\mathcal{A}$ runs the following experimental algorithm, $Experiment_{XP,\mathcal{A}}^{ECDLP}$ for our improved scheme, say *XP*.

Algorithm 3. $Experiment_{XP,\mathcal{A}}^{ECDLP}$

```
Call Reveal oracle. Let $\left(a'_{j,i}, sk\right) \leftarrow Reveal(f_j(x))$.
Call Test oracle: $Test\left(a'_{j,i}, sk\right)$.
if (Test oracle detects $sk$ as the correct key) then
  return 1
else
  return 0
end if
```

As in [20], we define $Succ_{XP,\mathcal{A}}^{ECDLP} = 2Pr\left[Experiment_{XP,\mathcal{A}}^{ECDLP} = 1\right] - 1$. Then an advantage function for the proposed scheme *XP* becomes

$$Adv_{XP}^{ECDLP}(t, q_R, q_T) = \max_{\mathcal{A}}\left\{Succ_{XP,\mathcal{A}}^{ECDLP}\right\},$$

where the maximum is taken over all $\mathcal{A}$ with execution time t, q_R is the number of queries to the Reveal oracle and q_T the number of queries to the Test oracle.

We say a probability function $f:N \to R_{[0,1]}$ is negligible in M if, for all $c > 0$, there exists $M_0 \in N$ suh that $f(M) \leqslant \frac{1}{M^c}$ whenever $M \geqslant M_0$, where N is the set of natural numbers, R the set of real numbers, and $R_{[0,1]} = \{x \in R | 0 \leqslant x \leqslant 1\}$ [20]. Our improved scheme *XP* is said to be secure against an adversary for computing a secret key of a security class in the user hierarchy, if there is no polynomial time adversary with non-negligible advantage. In other words, we say that our proposed scheme *XP* is secure if $Adv_{XP}^{ECDLP}(t, q_R, q_T) \leqslant \epsilon$, for any sufficiently small $\epsilon > 0$.

Consider the above experiment for an adversary $\mathcal{A}$. The adversary needs to compute or guess correctly the value $h(x_{j,i}||y_{j,i})$, where $s_iG_j = (x_{j,i}, y_{j,i})$ so that the secret key sk_j of SC_j can be computed correctly by that adversary. In order to get correct secret key sk_j of SC_j, the attacker needs to know s_i correctly from $s_iG_j = (x_{j,i}, y_{j,i})$. Recall that s_i is the sub-secret key of the predecessor class SC_i of SC_j and it is only known to SC_i. The attacker cannot to know s_i, because it is computationally infeasible problem to determine s_i due to ECDLP assumption. Thus, the problem of finding the secret key sk_j of a security class SC_j in the user hierarchy by the adversary $\mathcal{A}$ essentially reduces to the problem of finding the discrete logarithm s_i from s_iG_j knowing the public polynomial $f_j(x)$, the base point G_j and public elliptic curve parameters. Hence, $Adv_{XP}^{ECDLP}(t, q_R, q_T)$ depends on $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP}(t)$. Since $Adv_{\mathcal{D},E_p(a,b)}^{ECDLP}(t)$ is negligible, we finally have $Adv_{XP}^{ECDLP}(t, q_R, q_T)$ is also negligible. As a result, no adversary can compute the secret key sk_j of SC_j and our improved scheme becomes provably secure against an adversary for computing the secret key of a security class in the user hierarchy. □

6. Performance comparison of our improved scheme with Chung et al.'s scheme and Chen-Huang's scheme

In this section, we compare the performance analysis in terms of complexity and security for access control problems among Chung et al.'s scheme, Chen-Huang's scheme and our improved scheme.

In Table 1, we have compared the performance in terms of complexity for access control problems between our improved scheme, Chen-Huang's scheme and Chung et al.'s scheme. For analysis, the following notations are used:

- n: number of security classes in a user hierarchy.
- $|p|$: bit length of an integer p.
- k_i: degree of the public elliptic curve polynomial $f_i(x)$ for Chung et al.'s scheme and our improved scheme, that is the number of successors of a security class SC_i. In case of Chen-Huang's scheme, k_i is the number of successors of a security class SC_i.
- T_M: time for performing a modular multiplication.
- T_{EC_M}: time for performing a scalar multiplication.
- T_{XOR}: time for performing an XOR (Exclusive-OR) operation.
- T_H: time for performing a one-way hash function $h(\cdot)$.
- T_{ENC}: time for performing a symmetric-key encryption.
- T_{DEC}: time for performing a symmetric-key decryption.

In Chung et al.'s scheme and our improved scheme, the complexity requirement is as follows. Each security class needs to store the public parameters of all other security classes including its own public parameters. Note that CA publishes the public parameters: base point G_i and polynomial $f_i(x)$ for each security class SC_i. Since there are n security classes, each security class requires storage complexity $n|p|$ for G_i's. Each public elliptic curve polynomial $f_i(x)$, which is of degree k_i, requires

Table 1
Performance analysis in terms of complexity for access control problems among Chen-Huang's scheme, Chung et al.'s scheme and our improved scheme.

Required complexity	Chen-Huang's scheme	Chung et al.'s scheme	Our improved scheme
I_1	$n\lvert P_i\rvert + \sum_{i=1}^{n} k_i\lvert R_{ij}\rvert$	$\left(2n + \sum_{i=1}^{n} k_i + 3\right)\lvert p\rvert$	$\left(2n + \sum_{i=1}^{n} k_i + 3\right)\lvert p\rvert$
I_2	$\lvert sk_i\rvert$	$2\lvert p\rvert$	$2\lvert p\rvert$
I_3	$\sum_{i=1}^{n} k_i(T_{XOR} + T_H + T_{ENC})$	$\sum_{i=1}^{n} k_i(2T_{EC_M} + T_H)$	$\sum_{i=1}^{n} k_i(2T_{EC_M} + T_H)$
I_4	$T_{XOR} + T_H + T_{DEC}$	$T_{EC_M} + k_iT_M + T_H$	$T_{EC_M} + k_iT_M + T_H$
I_5	Updating the related public relation-parameters R_{ij}'s	Updating the related public parameters of $f_j(x)$'s	Renewing the secret keys sk_j's, the base points G_j's, and updating the public parameters of $f_j(x)$'s

I_1: Storage for public parameters.
I_2: Storage of private keys in each security class SC_i.
I_3: Construction of public polynomials $f_i(x)$'s in Chung et al.'s scheme and our improved scheme, or construction of public relation-parameters R_{ij}'s in Chen-Huang's scheme for key derivation.
I_4: Key derivation for a security class.
I_5: Adding a new security class.

Table 2
Performance analysis in terms of security for access control problems among Chen-Huang's scheme, Chung et al.'s scheme and our improved scheme.

	Chen-Huang's scheme	Chung et al.'s scheme	Our improved scheme
Resists contrary attack	Yes	Yes	Yes
Resists exterior root collecting attack	Yes	Yes	Yes
Resists collaborative attack	Yes	Yes	Yes
Resists equation attack	N/A	Yes	Yes
Resists forward security of the successors while changing $SC_i \geqslant SC_k \geqslant SC_j$ to $SC_i \geqslant SC_j$	Yes	Yes	Yes
Resists exterior root finding attack	N/A	No	Yes

storage space $(k_i + 1)|p|$, and each security class requires storage complexity due to storing n public polynomials is then $\sum_{i=1}^{n}(k_i + 1)|p|$. Apart from these public parameters, the common public parameters are p, a and b of the elliptic curve $E_p(a, b)$ and so due to these, each node requires additional storage complexity $3|p|$. Summing up all these, the total storage complexity for each security class becomes $n|p| + \sum_{i=1}^{n}(k_i + 1)|p| + 3|p| = \left(2n + \sum_{i=1}^{n} k_i + 3\right)|p|$. On the other hand, the storage complexity due to storing two private keys (secret key and sub-secret key) for each security class is $2|p|$. Now, CA needs the complexity of $\sum_{i=1}^{n} k_i(T_{EC_M} + T_H)$ to compute $s_iG_j = (x_{j,i}\|y_{j,i})$'s and $h(x_{j,i}\|y_{j,i})$'s, and $\sum_{i=1}^{n} k_iT_{EC_M}$ to construct n polynomials for all security classes. Each security class in the user hierarchy also requires the complexity $T_{EC_M} + k_iT_M + T_H$ in order to derive a successor's secret key.

In Chen-Huang's scheme, each security class needs to store all the public parameters of all other security classes including its own public parameters. In this scheme, CA publishes the large positive integer P_i for each security class SC_i, which requires $|P_i|$ bits. CA then publishes the public relation-parameter R_{ij} for deriving the secret key sk_j of the security class SC_j in which $R_{ij} = E_{h(P_j \oplus sk_i)}(sk_j)$, where P_j is the public large positive integer for SC_j and sk_i the secret key of SC_i. If there are k_i number of public relation-parameters for a security class SC_i, each security class requires storage complexity of $n|P_i| + \sum_{i=1}^{k} k_i|R_{ij}|$ bits, where $|P_i|$ and $|R_{ij}|$ are the number of bits present in P_i and R_{ij} respectively. For example, if we use the Advanced Encryption Standard (AES) [21] as symmetric-key cryptosystem, then AES takes an input of 128 bits in encryption algorithm and produces a ciphertext of 128 bits and thus, $|R_{ij}|$ = 128 bits when the key sk_j is 128 bits only. The storage space required for each security class SC_i to store the large secret key sk_i is $|sk_i|$ bits. CA needs the complexity of $\sum_{i=1}^{n} k_i(T_{XOR} + T_H + T_{ENC})$ for generating the public relation-parameters of all security classes in the hierarchy. Finally, each security class SC_i requires the complexity of $T_{XOR} + T_H + T_{DEC}$ to derive the secret key of a successor class SC_j.

It is observed that the complexity for our improved scheme is higher than that for Chung et al.'s scheme when a new security class is inserted into the existing hierarchy. This is because CA needs to renew the secret keys sk_j's and also the base points G_j's for successor classes SC_j's for adding a new security class. On the other hand, Chen-Huang's scheme uses the hash function and symmetric-key encryption/decryptions. Since symmetric-key cryptosystem is efficient than elliptic curve public-key cryptosystem, obviously Chen-Huang's scheme requires less computational complexity compared with Chung et al.'s scheme and our improved scheme. However, for ECC cryptosystem to provide sufficient security the prime number p to be chosen 160 bits only. As a result, in Chung et al.'s scheme and our improved scheme the secret key sk_i and sub-secret key s_i of a security class SC_i are of 160 bits only, whereas in Chen-Huang's scheme P_i and sk_i for the security class SC_i may be larger than 160 bits.

Table 2 shows the performance analysis in terms of security for access control problems among Chung et al.'s scheme, Chen-Huang's scheme and our improved scheme. As stated earlier that our main theme of this paper is to identify a security flaw in Chung et al.'s scheme and then provide a fix to remedy the flaw in their scheme, our improved scheme provides better security as compared to Chung et al.'s scheme, because Chung et al.'s is vulnerable to exterior root finding attack in which an attacker is able to compute the secret key of a security class when a new security class is inserted into the existing hierarchy. Even if our improved scheme requires more complexity due to inserting new security classes into the existing hierarchy as compared with that for Chung et al.'s scheme, by considering the security aspects our improved scheme is better than Chung et al.'s scheme.

7. Conclusion

In this paper, we have pointed out a security flaw in Chung et al.'s scheme. Our proposed exterior root finding attack shows that the secret key sk_j of a security class SC_j is revealed to an attacker, who is not any user in the user hierarchy, by knowing the public polynomial $f_j(x)$ of SC_j and the updated public polynomial $f_j'(x)$ of SC_j when CA adds a new security class SC_k such that $SC_i \geqslant SC_k \geqslant SC_j$. To eliminate this security flaw, we have further proposed a simple improvement on Chung et al.'s scheme. We have shown that a predecessor class SC_i can easily compute the secret key sk_j' using the updated public polynomial $f_j'(x)$ of the successor class SC_j and its own sub-secret key s_i. We have analyzed and compared the performance in terms of complexity and security for access control problems of our improved scheme with those for Chung et al.'s scheme. It is observed that though our improved scheme requires more complexity for adding a new security class to achieve maximum security as compared with Chung et al.'s scheme, our scheme outperforms in terms of security because Chung et al.'s scheme is vulnerable to exterior root finding attack.

Acknowledgments

The authors thank the anonymous reviewers, the Editor and the Editor-in-Chief, Prof. Witold Pedrycz, for providing constructive and generous feedback.

References

[1] S.G. Akl, P.D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, ACM Transactions on Computer Systems (TOCS) 1 (3) (1983) 239–248.

[2] S. Mackinnon, P. Taylor, H. Meijer, S. Akl, An optimal algorithm for assigning cryptographic keys to control access in a hierarchy, IEEE Transactions on Computers 34 (9) (1985) 797–802.

[3] L. Harn, H.Y. Lin, A cryptographic key generation scheme for multilevel data security, Computers & Security 9 (6) (1990) 539–546.

[4] C.H. Lin, Dynamic key management schemes for access control in a hierarchy, Computer Communications 20 (15) (1997) 1381–1385.

[5] 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.

[6] F.H. Kuo, V.R.L. Shen, T.S. Chen, F. Lai, Cryptographic key assignment scheme for dynamic access control in a user hierarchy, Computers and Digital Techniques, IEE Proceedings 146 (5) (1999) 235–240.

[7] H.M. Tsai, C.C. Chang, A cryptographic implementation for dynamic access control in a user hierarchy, Computers & Security 14 (2) (1995) 159–166.

[8] 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 18th International Conference on Advanced Information Networking and Applications (AINA'04), vol. 2, 2004, pp. 376–379.

[9] V.L.R. Shen, T.S. Chen, A novel key management scheme based on discrete logarithms and polynomial interpolations, Computers & Security 21 (2) (2002) 164–171.

[10] V.L.R. 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.

[11] Y.F. Chung, H.H. Lee, F. Lai, T.S. Chen, Access control in user hierarchy based on elliptic curve cryptosystem, Information Sciences 178 (1) (2008) 230–243.

[12] C.-L. Hsu, T.-S. Wu, Cryptanalysis and improvements of two cryptographic key assignment schemes for dynamic access control in a user hierarchy, Computers & Security 22 (5) (2003) 453–456.

[13] T.-S. Chen, J.-Y. Huang, A novel key management scheme for dynamic access control in a user hierarchy, Applied Mathematics and Computation 162 (1) (2005) 339–351.

[14] M. Ben-Or, Probabilistic algorithms in finite fields, in: Proceedings of 22nd Annual Symposium on Foundations of Computer Science (IEEE FOCS'81), 1981, pp. 394–398.

[15] R.W.D. Nickalls, A new approach to solving the cubic: Cardan's solution revealed, The Mathematical Gazette 77 (480) (1993) 354–359.

[16] W. Stallings, Cryptography and Network Security: Principles and Practices, third ed., Prentice Hall, 2003.

[17] N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation 48 (1987) 203–209.

[18] H. Cohen, A Course in Computational Algebraic Number Theory, fourth ed., Springer-Verlag, 1991.

[19] R. Dutta, R. Barua, Provably secure constant round contributory group key agreement, IEEE Transactions on Information Theory 54 (5) (2008) 2007–2025.

[20] J. Baek, R. Steinfeld, Y. Zheng, Formal proofs for the security of signcryption, in: Proceedings of PKC 2002, LNCS, vol. 2274, 2002, pp. 80–98.

[21] Advanced Encryption Standard, FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>. (accessed November 2010).