ANALYSIS OF SECURE KEY MANAGEMENT SCHEME FOR DYNAMIC HIERARCHICAL ACCESS CONTROL BASED ON ECC AND DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL

Mr. Manoj Ahke*

Mr. Ram Ratan Ahirwar*

Abstract

An access control mechanism in a user hierarchy is used to provide the management of sensitive information for authorized users. The users and their own information can be organized into a number of disjoint sets of security classes according to their responsibilities. Each security class in a user hierarchy is assigned an encryption key and can derive the encryption keys of all lower security classes according to predefined partially ordered relation. In 2006, Jeng and Wang proposed an efficient key management scheme based on elliptic curve cryptosystems. This paper, however, pointed out that Jeng–Wang scheme is vulnerable to the so-called compromising attack that the secret keys of some security classes can be compromised by any adversary if some public information modified. We further proposed a secure key management scheme based on elliptic curve cryptosystems to eliminate the pointed out the security leak and provide better security requirements. As compared with Jeng and Wang's scheme (Jeng and Wang, 2006), the proposed scheme has the following properties. (i) It is simple to execute the key generation and key derivation phases. (ii) It is easily to address dynamic access control when a security class is added into or deleted from the hierarchy. (iii) It is secure against some potential attacks. (iv) The required storage of the public/secret parameters is constant.

Keywords: Key management, Key assignment ,Elliptic curve,Hierarchical access control

* S.A.T.I VIDISHA.

1. Introduction

The access control problem in a user hierarchy is used to many applications such as schools, governments, military, corporations, computer network systems, and database management systems. All users in such a system form a user hierarchy and can be assigned into a number of disjoint sets of security classes, say SC ={SC1, SC2, . . ., SCn}, which are partially ordered by a binary relation "≤". In (SC, ≤), SCj ≤SCi means that the security level of class SCi is higher than or equal to the security class SCj. In other words, users in SCj can access the encrypted information held by users in SCj, but the opposite is disallowed. The secret key Ki is used by each security class SCi to encrypt/decrypt its sensitive information. When a user in SCi would like to retrieve data encrypted by SCj, he should get the right key Kj.

Akl and Taylor (1983) first proposed a solution to solve the hierarchical access control problem. In their scheme, each security class is assigned a secret key and a public parameter. The security class SCi can successfully use its secret key and some public parameters

to derive the secret key of the security class SCj such that SCj ≤SCi.Main drawback of Akl and Taylor's scheme is that the size of the public parameter grows linearly with the number of security classes. Latter, Mackinnon et al. (1985) presented an optimal algorithm, called the canonical assignment, to reduce the value of public parameters. However, it is difficult to find an optimal canonical algorithm. Above two schemes adopted the top-down approach to generate all secret keys. All secret keys must be re-generated when a security class is added into or deleted from the user hierarchy. The dynamic access control problems in access control cannot

be efficiently solved. Harn and Lin (1990) proposed a bottom-up key generating scheme to improve the computational and storage complexities. Since then, several schemes (Chang et al., 1992, 2004; Chung et al., 2008; Das et al., 2005; Hsu et al., 2008; Hwang and Yang, 2003; Jeng and Wang, 2006; Kuo et al., 1999; Shen and Chen, 2002; Wu and Wei, 2006; Wu et al., 1995; Wu and Chang, 2001; Yang and Li, 2004) have been proposed to efficiently deal with the dynamic access control problems.

Chang et al. (1992) proposed a key assignment scheme based on Newton's interpolations method and one-way function. In their scheme, a user with higher security class must iteratively perform the key derivation process for deriving the secret keys of its lowest security class(es). It is inefficient in the key derivation process. Wu and Chang (2001) and Shen and Chen (2002) proposed cryptographic key assignment schemes to solve the access policy using polynomial interpolations. In their schemes, the system does not need to maintain the security classes' and the users' secret keys. That is, any user can freely change his/her secret key for some security reasons. However, Hsu and Wu pointed out a security leak inherent in both schemes (Hsu and Wu, 2003). An attacker can violate the predefined access control policy to access to the unauthorized information. Latter, Yang and Li (2004) proposed a cryptographic key assignment scheme based on one-way hash function. The cryptographic key of Yang and Li's scheme is determined by one-way hash functions. Hsu et al. (2008) also pointed out some security flaws of Yang and Li's scheme to show that the claimed security requirement is violated. That is, the users can overstep his authority to access unauthorized information. Hsu et al. further proposed two improvements to eliminate the pointed out flaws.

Recently, Jeng and Wang (2006) proposed an efficient key management and derivation scheme based on the elliptic curve cryptosystems (it is denoted as the Jeng–Wang scheme for short). In Jeng–Wang scheme, the secret key of each security class can be determined by itself instead of a trusted central authority. Major advantage of Jeng–Wang scheme is to solve dynamic key management efficiently and flexibly. It is unnecessary to re-generate keys for all the security classes in the hierarchy when the security class is added into or deleted from the user hierarchy. This paper, however, pointed a compromising attack on Jeng–Wang scheme, which implies their scheme cannot achieve the claimed requirements.

Finally, we proposed a secure key management scheme based on elliptic curve cryptosystem against the compromise attack. As compared with Jeng and Wang's scheme (Jeng and Wang, 2006), the proposed scheme has the following properties. (i) It is simple to execute the key generation and key derivation phases. (ii) It is easily to address dynamic access control when a security class is added into or deleted from the user hierarchy. (iii) It is secure against both interior and exterior attacks. (iv) The required storage of the public/secret parameters is constant.

The rest of this paper is sketched as follows. In Section 2, we reviewed Jeng–Wang's key management scheme and demonstrated the compromise attack on Jeng–Wang scheme. In Section 3, we proposed a secure key management scheme based on elliptic curve cryptosystem. In Section 4,we discussed the dynamic key management. We analyzed the security and performance of the proposed scheme in Sections 5 and 6, respectively. Finally, we give some conclusions.

2. The Jeng–Wang key management scheme and its security leak

In this section, we briefly reviewed Jeng and Wang's key management scheme (Jeng and Wang, 2006). We also demonstrated a compromising attack on their scheme to show that the claimed necessary security requirement is violated.

2.1. The Jeng–Wang scheme

In 2006, Jeng and Wang proposed an efficient key management and derivation scheme based on the elliptic curve cryptosystem to solve the hierarchical access control problems (Jeng and Wang, 2006). Their scheme consists of the initialization, the key generation, and the key derivation phases. In the initialization phase, a central authority (CA) determines all system parameters. In the key generation phase, each security class determines a secret point on an elliptic group over a finite field as its secret key. All secret points are sent to CA via a secure channel for constructing a key relationship derivation hierarchy. In the key derivation phase, the predecessor can use its own secret key and the public information related to the successor(s) to derive the encryption/decryption key(s) for accessing the authorized file(s). Detailed descriptions of these phases are given below.

Initialization phase – CA randomly chooses a large prime p and an elliptic curve $E_p(a, b) : y^2 = x^3 + ax + b \pmod p$ with a point O at infinity, where $a, b \in Z_p^*$ are two random integers satisfying that $4a^3 + 27b^2 \bmod p \neq 0$. Let $G \in E_p(a, b)$ be a base point of order q, where q is a large prime. CA also selects a transformation function $\tilde{A} : (x, y) \rightarrow v$ for transforming a point on $E_p(a, b)$ into a real number $v \in Z_p^*$. Finally, CA publishes $(p, q, \tilde{A}, E_p(a, b), G)$. Key generation phase – Initially, CA determines its secret key

$k_{ca} \in Z_q^*$ and publishes the corresponding public key $Y_{ca} = k_{ca}G$. Without loss of generality, let $SC = \{SC_1, SC_2, \ldots, SC_n\}$ be a user hierarchy with n disjoint sets of security classes which are partially ordered by a binary relation "≤". Each security classownstwosecret keys, a secret key and an encryption key. The secret key is used to derive the successor's encryption key. The encryption key is used to encrypt messages for confidentiality. Each security class SC_i (for $i=1,2, \ldots, n$) randomly chooses a secret key $k_{i,1} \in Z_q^*$ and an encryption key $k_{i,2} \in [1, p-1]$, and computes the corresponding public key $Y_i = k_{i,1}G$. Each security class SC_i further chooses a random integer $r_i \in Z_q^*$, computes

$C_{i,1} = r_iG$ (1)

$C_{i,2} = (k_{i,2}, k_{i,1}) + r_iY_{ca}$ (2)

and transmits $(C_{i,1}, C_{i,2})$ to the central authority CA. The CA can derive $(k_{i,2}, k_{i,1})$ from $(C_{i,1}, C_{i,2})$ by the following equation:

$(k_{i,2}, k_{i,1}) = C_{i,2} - k_{ca}C_{i,1}$ (3)

For security class SC_i (for $i = n, n-1, \ldots, 1$), CA employs the bottom-up approach to compute

$v_{i,j} = \tilde{A}(k_{j,1}Y_i)$ (4)

and construct a polynomial $f_i(x)$ by interpolating the points $(v_{i,j}, k_{j,2})$'s for all $SC_i < SC_j$. CA finally publishes $f_i(x)$'s for $i=1, 2, \ldots, n$.

Key derivation phase – When the security class SC_i wants to access the encrypted and held by SC_j where $SC_j < SC_i$, it can use its secret key $k_{i,1}$, the public key Y_j of SC_j, and the public information $f_i(x)$ to derive $k_{j,2} = f_j(\tilde{A}(k_{i,1}Y_j))$. With the knowledge of the encryption key $k_{i,2}$, the security class SC_i can decrypt and access the data encrypted by SC_j.

2.2. Compromising attack on Jeng–Wang scheme

Firstweproposed a compromising attack on Jeng–Wang scheme to show that any outsider is able to derive an unauthorized encryption key if the relationship between any two security classes is updated.

Recall Jeng–Wang scheme, each security class SCi generates its key pair (ki,1, Yi) and an encryption key ki,2, which are contributed to construct the polynomial fj(x) for its successor SCj where SCj < SCi. Considering the scenario of the dynamic access control management that CA can add or delete some predecessors into or from SCj, CA will update the public polynomial as f _ j (x). Let ¯Gj be the set of the security classes SCl's (SCj < SCl), which still remain as the predecessors of SCj. We can see that the secret keys belonged to the security class SCl ∈ ¯Gj are also contributed to the new polynomial f _ j (x). It means that the point (*v*l,j, kj,2) associated with the security class SCl ∈ ¯Gj will satisfy the polynomial (*v*l,j) = 0, where (x) = fj(x) − f _ j (x). With the knowledge of fj(x) and f _ j (x), the adversary can try to derive all *v*l,j's such that (*v*l,j) = 0 by finding the roots

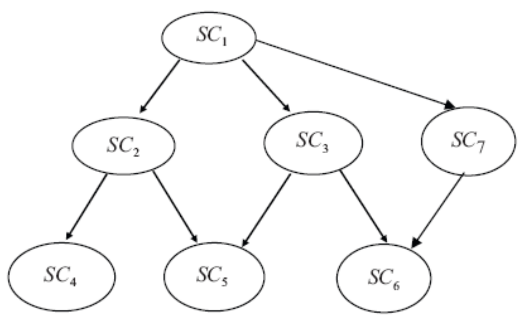

Fig. 1. A partially ordered user hierarchy.

of the polynomial (x) = 0 in the polynomial time (Ben-Or, 1981; Cohen, 1991). The adversary without knowing any secret information can further derive the encryption key kj,2 of the security class SCj by kj,2 = fj(*v*l,j). Hence, Jeng–Wang scheme is vulnerable to our proposed compromising attack. For simplicity, we gave a simple example to demonstrate that our proposed compromising attack is effective in attacking on Jeng–Wangscheme. Suppose that the user hierarchy contains seven security classes as shown in Fig. 1. The predecessors of the security class SC6 are SC1, SC3, and SC7, and the public polynomial f6(x) for SC6 is constructed by the points (*v*1,6, k6,2), (*v*3,6, k6,2), and (*v*7,6, k6,2). If the security class SC7 is removed from the user hierarchy, the public polynomial f6(x) for SC6 will be replaced with f_6 (x) which is constructed by the points (*v*1,6, k6,2) and (*v*3,6, k6,2). With the knowledge of f6(x) and f_6 (x), any adversary can derive *v*1,6 or *v*3,6 by finding the roots of the polynomial (x) = f6(x) − f_6(x) in the polynomial time (Ben-Or, 1981; Cohen, 1991). The adversary can further use *v*1,6 and *v*3,6 to derive the encryption key of SC6 by k6,2 = f6(*v*1,6) and k6,2 = f6(*v*3,6) by himself.

From above analysis, we can see that some secret key(s) might be compromised if the corresponding public polynomial is modified but its some point(s) are unchanged. Considering another scenario that the secret and public keys of some (not all) predecessors of the security class SCj are updated, CA will update the public polynomial as f _ j (x). Let ˆGj be the set of the security classes SCl's (SCj < SCl) whose secret and public keys are unchanged. We can see that the secret keys belonged to the security class SCl ∈ ˆGj are also contributed to the new polynomial f _ j (x). Hence, any outsider is able to derive all *v*l,j's for SCl ∈ ˆGj such that (*v*l,j) = 0, where (x) = fj(x) − f _ j (x), in the polynomial time (Ben-Or, 1981; Cohen, 1991) and then obtains the encryption key for SCj as kj,2 = fj(*v*l,j).

That is, Jeng–Wang scheme is also vulnerable to our proposed compromising attack even if the relationship between any two security classes is not updated.Recall the same simple example as shown in Fig. 1. The predecessors of the security class SC6 are SC1, SC3, and SC7, and the public polynomial f6(x) for SC6 is constructed by the points (*v*1,6, k6,2), (*v*3,6, k6,2), and (*v*7,6, k6,2). If the security and public keys for the security class SC7 are changed as (k_7,1, Y_7 = k_7,1G), the public polynomial f6(x) for SC6 will be replaced with f_6(x) which is constructed by the points (*v*1,6, k6,2), (*v*3,6, k6,2), and (*v*_7,6, k6,2) where *v*_7,6 = ˜A(k6,1Y_7). With the knowledge of f6(x) and f_6 (x), any outsider can derive *v*1,6 or *v*3,6 by finding the roots of the polynomial (x) = f6(x) − f_6(x) in the polynomial time (Ben-Or, 1981; Cohen, 1991). The encryption key of SC6 will be compromised by k6,2 = f6(*v*1,6) or k6,2 = f6(*v*3,6).

Computational complexity for such kind of attack includes that of finding the roots of a polynomial and that of computing a polynomial for a given x-coordinate value. Complexity of these two problems depends on the number of the relationship related to the target security class. From the references (Ben-Or, 1981; Cohen, 1991), all of them can be solved in the polynomial time. Hence, the adversary can plot such an attack in the polynomial time.

3. The key mangement scheme

This scheme can also be divided into three phases as those in Jeng–Wang's scheme. In the initialization phase, a central authority (CA) determines all system parameters. In the key generation phase, each security class chooses its secret key and computes the corresponding public key. Then each security class uses its secret key and a public parameter to compute its encryption key. Afterward each security class sends its encryption key to CA. CA extracts the encryption key of the security class and constructs
the public polynomials of each security class by using a top-down approach. In the key derivation phase, the predecessor can use its own encryption key and the public information related to the successor(s) to derive the decryption key(s) for accessing the authorized file(s). Detailed descriptions of these phases are given below.

Initialization phase – CA randomly chooses a large prime p, a large prime q, and a, b ∈ Z∗p be two parameters satisfying that 4a3 +27b2 modp /= 0. Let Ep(a, b) be an elliptic curve over GF(p) containing a set of points (x,y)'s with x, y ∈ Z∗p and a point O at infinity, where y2 = x3 + ax + b(mod p). Let G1 be an additive cyclic group with prime order q and G be a generator of G1. CA selects a symmetric cryptosystem in which Ek(·) and Dk(·) are the encryption and decryption algorithms with the key k. CA also selects two secure one-way hash functions $H1 : \{0, 1\}^* \times G1 \rightarrow Z^*_q$ and $H2 : G1 \rightarrow Z^*_q$.

CA determines its secret key kca and makes Yca public, where Yca = kcaG. Finally, CA publishes (p, q, H1, H2, Ep(a, b), G, Yca). Key generation phase – Without loss of generality, let SC ={SC1, SC2, ..., SCn} be a user hierarchy with n disjoint sets of security classes which are partially ordered by a binary relation "≤". For each security class SCi, it chooses its own secret key $k_{i,1} \in Z^*_q$ and computes the corresponding public key $Y_i = k_{i,1}G$ and the encryption key $k_{i,2}$ for i=1, 2, . . ., n. This phase can be achieved by the following steps:

Step 1. Each security class SCi ∈ SC performs the following tasks to generate its secret data Vi:

(a) Randomly choose a number $r_i \in Z^*_q$ and compute the public information $R_i = r_iG$.

(b) Compute the encryption key $k_{i,2} = H1(k_{i,1}, R_i)$.

(c) Compute $v_i = k_{i,2}(H2(r_iY_{ca}))\bmod q$.

(d) Send the secret data v_i to CA by a secure method.

Step 2. Upon receiving v_i from SCi for i=1, 2, . . ., n, CA computes $k_{i,2} = v_i(H2(k_{ca}R_i))^{-1} \bmod q$ to extract $k_{i,2}$ by using his secret key kca.

Step 3. CA computes the integer $DK_{i\rightarrow j} = H1(k_{i,2}, Y_j)$ for all SCi's (for i=1, 2, . . ., n) such that SCj < SCi. The symbol $DK_{i\rightarrow j}$ denotes the derivation key of the security class SCi for deriving the encryption key of its successor SCj.

Step 4. CA uses the polynomial interpolation to determine a public function fj(x) for each security class SCj and j = n, n−1, . . ., 1. The polynomial fj(x) is constructed by the points $(DK_{i\rightarrow j}, E_{H1(k_{i,2},R_j)}(k_{j,2}))$ for all SCi's such that SCj < SCi.

Key derivation phase – When the security class SCi wants to access the encrypted data held by SCj where SCj < SCi, SCi can perform the following steps to obtain SCj's encryption key $k_{j,2}$:

Step 1. Compute the derivation key $DK_{i\rightarrow j} = H1(k_{i,2}, Y_j)$.

Step 2. Compute $E_{H1(k_{i,2},R_j)}(k_{j,2}) = f_j(DK_{i\rightarrow j})$.

Step 3. Obtain the encryption key $k_{j,2}$ of SCj by decrypting $E_{H1(k_{i,2},R_j)}(k_{j,2})$ as $D_{H1(k_{i,2},R_j)}(E_{H1(k_{i,2},R_j)}(k_{j,2}))$.

4. Dynamic key management

This section presented the solutions to dynamic key management problems, including adding/deleting a security class and changing a secret key.

4.1. Adding a security class

Consider the scenario that a new security class SCl is added into an existing user hierarchy such that SCi > SCl > SCj. The security class SCl computes its own secret key kl,2, generates the secret data *vl*, and sends *vl* to CA. Then, CA updates the public function fi(x) of those security classes SCj (SCi > SCl). We describe these steps below in detail.

Step 1. The security class SCl performs the following tasks to generate its secret data *vl*:

(a) Randomly choose a number rl ∈ Z∗p and compute the public information Rl = rlG.

(b) Compute the secret key kl,2 =H1(kl,1, Rl).

(c) Compute *vl* = kl,2(H2(rlYca))mod q.

(d) Send the secret data *vl* to CA via a secure channel.

Step 2. Upon receiving *vl* from SCl, CA computes kl,2 = *vl*(H2(kcaRl))−1modq to extract kl,2 by using his secret key kca.

Step 3. For all SCj's (where SCj < SCl), CA computes DKl_→j =H1(kl,2, Yj) and constructs new polynomials fj(x)'s including the point (DKl_→j, EH1(kl,2,Rj)(kj,2)).

Step 4. For all SCi's (where SCi > SCl), CA computes DKi_→l =H1(ki,2, Yl), DKi_→l =H1(ki,2, Yl) and constructs new polynomials fl(x)'s including the points (DKi_→l, EH1(ki,2,Rl)(kl,2))'s.

Note that the proposed schememust perform secret key updating for all successors belonged to SCl before executing above tasks if the proposed scheme should achieve backward confidentiality.

Backward confidentiality is that user(s)/security class(es) join into the user hierarchy cannot access to any old key or sensitive information. Tasks for secret key updating are mentioned in Section 4.3.

4.2. Deleting a security class

A security class SCl is deleted from an existing user hierarchy such as SCi > SCl > SCj, CA not only revokes information related to SCl, but also alters the relationship between the involved expredecessor SCi and ex-successor SCj of SCl. CA performs the following steps:

Step 1. For all SCj's (where SCj < SCl), reconstruct new polynomial fj(x)'s excluding the point (DKl_→j, EH1(kl,2,Rj)(kj,2)).

Step 2. Delete the security class SCl from the user hierarchy and discard the secret key and public parameters of SCl.

Note that the proposed schememust perform secret key updating for all successors belonged to SCl before executing above tasks if the proposed scheme should achieve forward confidentiality. Forward confidentiality is that users/security class(es) left from the user hierarchy cannot access to any future key or sensitive information.

Tasks for secret key updating are mentioned in Section 4.3.

4.3. Changing a secret key

It might be necessary to change the secret key for some security consideration. For example, if the security class suspected that its secret key(s) are revealed or disclosed to adversary, the security class might change its secret keys. Another example is that the secret keys of the security class will be updated when one of its predecessors is deleted from user hierarchy. If the secret keys are unchanged under the circumstance, the deleted predecessor is still able to access the information hold by the security class. As mentioned in Sections 4.1 and 4.2, it is necessary to update secret key(s) for achieving backward and forward confidentiality. When

a security class SCi wants to change its secret key(s), CA only needs to update the public key and the public information of SCi in our proposed scheme. The key pair (ki,1, Yi = ki,1G) of the security Sci is used to establish a secure communications between SCi and CA, and the secret key ki,2 is used to encrypt its sensitive information and derive the secret key(s) of its successors. If the security class SCi want to change ki,1 as k_i,1, SCi only informs CA to update and announce its new public key Y_i = k_i,1G. If the security class SCi only wants to change ki,2 as k_i,2, SCi and CA cooperatively perform the following steps:

Step 1. Security class SCi performs the following tasks to generate its secret data *v*_i:

(a) Randomly choose a number r_i ∈ Z∗q and compute the public information R_i = r_iG.

(b) Compute the encryption key k_i,2 = H1(ki,1,R_i).

(c) Compute *v*_i = k_i,2(H2(r_iYca))mod q.

(d) Send the secret data *v*_i to CA by a secure method.

Step 2. Upon receiving *v*_i from SCi, CA computes k_i,2 = *v*_i(H2(kcaR_i))−1modq to extract k_i,2 by using his secret key kca.

Step 3. CA computes the integer DKi_→j = H1(k_i,2, Yj) for all SCj's such that SCj < SCi and reconstructs the new polynomial f _ j (x) by replacing the point (DKi_→j, EH1(ki,2,Rj)(kj,2)) with (DKi_→j, EH1(k_i,2,Rj)(kj,2)) in fj(x).

Step 4. For all SCl's such that SCl > SCi, CA reconstructs the new polynomial f _ i (x) by replacing the points (DKl_→i, EH1(kl,2,Ri)(ki,2))'s with (DKl_→i, EH1(kl,2,R_i)(k_i,2))'s in fi(x).

5. Security Analysis

In this section, we define the well-known security assumption (Diffie and Hellman, 1976; Menezes et al., 1997) and then discuss the security analysis of the proposed scheme based the assumption. The security assumption is defined as follows. One-way hash function (OWHF) (Diffie and Hellman, 1976; Menezes et al., 1997): a secure one-way hash function h has the following properties that (i) Given a output h(x) of a one-way hash function h, it is computationally infeasible to derive x from h(x); (ii) it is computationally infeasible to find two distinct values x and x_ such that that h(x) = h(x_).

5.1. Considerations to the prevention of compromising attacks

Consider the scenario that a successor SCj (SCj < SCi) who knows the public parameters (Rj, Yj, fi(x)) attempts to derive SCi's encryption key ki,2.No information is about SCi's encryption key ki,2 except the public polynomial fi(x). The public data fi(x) is constructed by the points (DKi_→j, EH1(ki,2,Rj)(kj,2))'s and the adversary will face the problem of reversing the one-way hash function to derive ki,2.

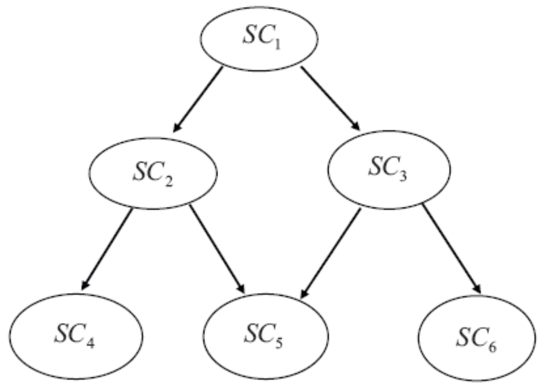

Fig. 2. A small sample of a POSET hierarchy.

Therefore, the lower security class cannot derive the secret key of the higher security class.

5.2. Considerations to the prevention of collusive attacks

Consider the problem of two or lower security classes collude with each other to derive the secret key of higher security class. For simplicity, let SCA, SCB and SCC be the successors of SCi and then attempt to cooperatively derive SCi's encryption key ki,2. They can have the following equations:

kA,2 = DH1(ki,2,RA)(fi(DKi_→A)) = DH1(ki,2,RA)(fi(H1(ki,2, YA))),

kB,2 = DH1(ki,2,RB)(fi(DKi_→B)) = DH1(ki,2,RB)(fi(H1(ki,2, YB))), and

kC,2 = DH1(ki,2,RC)(fi(DKi_→C)) = DH1(ki,2,RC)(fi(H1(ki,2, YC))).

It can be seen that ki,2 is protected under the secure one-way hash function H1. This implies that our scheme is secure against the collusive attacks.

5.3. Considerations to the prevention of equation attacks

If two security classes have the common successor(s), one of them might attempt to use the public polynomial(s) of another class(es) for deriving unauthorized secret keys. However, this attack is not possible in our scheme since the successors' encryption keys are encrypted by using the secret key of its predecessor.

As shown in Fig. 2, SC2 and SC3 have the common successor SC5. The security class SC2 might attempt to derive SC6's encryption key by using SC3's public polynomial f3(x). For simplicity,weuse the example depicted in Fig. 2 to demonstrate that the relationships SC2 > SC5 and SC3 > SC5. SC2 might attempt to obtain SC6's encryption key k6,2 through their common successor SC5.With the knowledge of f3(x) and the encryption key k5,2, the security class SC2 still cannot derive SC6's encryption key k6,2. We can see that f3(x) is constructed by the points (DK3_→5, EH1(k3,2,R5)(k5,2)) and (DK3_→6, EH1(k3,2,R6)(k6,2)), where DK3_→5 =H1(k3,2, Y5) and DK3_→6 =H1(k3,2, Y6). It is computationally hard for SC2 to derive SC6's encryption key k6,2 from the public parameters and SC5's encryption key k5,2 without knowing SC3's encryption key k3,2. It can be seen that the derivation of SC6's encryption key k6,2 is based on the difficulty of reversing one-way hash function.

5.4. Considerations to the prevention of exterior root finding attacks

Consider the scenario that an adversary who is not a user in any security class in a user hierarchy attempts to derive the encryption key of a security class by using the root finding algorithm. All successors' encryption keys of a security class SCi are embedded in its public polynomial fi(x). When CA adds/deletes some successors into/from SCi, CA updates the public polynomial as f _ i (x). However, for those successors, which remain as successors of SCi in f _ i (x), their secrets are still at the same positions of f _ i (x). An adversary can generate a polynomial by taking the difference of fi(x) and f _ i (x). Then an adversary can refer (Ben-Or, 1981; Cohen, 1991) to the roots of the equation fi(x) − f _ i (x) = 0 in a polynomial time. With the knowledge of the roots, the adversary can easily derive the secret keys of the successors of SCi. In our scheme, an adversary can compute the x-coordinates from the equation fi(x) − f _ i (x) = 0. That is, the adversary can get DKi_→j for SCj < SCi. From this value, it is computationally infeasible to compute ki,2 of SCi. Therefore, it is computational hard to derive the encryption key kj,2 of SCi. Since kj,2 is encrypted by the

key H1(ki,2, Rj), which is composed by the encryption key ki,2 of SCi, our scheme is secure against such type of attack.

5.5. Considerations to group confidentiality

We consider the group confidentiality that users who are not in the user hierarchy or its predecessor(s) should not have access to any key that can decrypt any multicast data sent to the security class. In our proposed scheme, all information for a security class is encrypted by its encryption key. From above security analysis, the security of the secret keys is based on a secure one-way hash function.

Only the users in this security class or its predecessor(s) can follows our proposed scheme to derive the encryption and access the information. Others will face the intractability of reversing the one-way hash function. Hence, the proposed scheme can achieve group confidentiality.

5.6. Considerations to forward/backward confidentiality

Forward confidentiality is that users/security class(es) left from the user hierarchy cannot access to any future key or sensitive information.

Backward confidentiality is that user(s)/security class(es) join into the hierarchy cannot access to any old key or sensitive information. Our proposed scheme considers the dynamic key management and hence it can achieve forward/backward confidentiality if changing secret key is performed under above two circumstances.

5.7. Considerations to collusion freedom

Collusion freedom is that any set of deleted members or security classes should not be able to deduce the current used key. In Section 4.2, we consider the issues and the tasks about deleting a security class from the user hierarchy. If any security class is deleted from the user hierarchy, the public information of its successor(s) must be updated. As mentioned in Section 4.2, the proposed scheme must perform secret key updating for all successors belonged to SCl after deleting a security class if the proposed scheme should achieve forward confidentiality. Moreover, all secret keys hold by deleted security classes is independent. Hence, deleted security classes cannot act in collude to obtain any sensitive information of the user hierarchy. This implies that the proposed scheme can achieve collusion freedom.

6. Conclusions

We pointed out that Jeng–Wang scheme (Jeng and Wang, 2006) is insecure against our proposed compromising attack, which implies their claimed necessary security requirement is violated. Elaborating on the merits of elliptic curve cryptosystem, we further presented a secure and efficient cryptographic key management scheme. The proposed scheme is secure against the proposed compromising attack and some potential attacks. Not only necessary security requirements for key management are achieved, but also the efficient and practical dynamic key management solutions are proposed. The security class in the user hierarchy can freely select its own secret keys and change its encryption key ki,2 to k_i,2 for some security considerations. Therefore, the proposed scheme can achieve higher security with low computational costs.

References

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

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

[3] Chang, C.C., Hwang, R.J., Wu, T.C., 1992. Cryptographic key assignment scheme for access control in a hierarchy. Information Systems 17 (3), 243–247.

[4] Chang, C.C., Lin, I.C., Tsai, H.M., Wang, H.H., 2004. A key assignment scheme for controlling access in partially ordered user hierarchies. In: Proceedings of the 18th FIEEE. International Conference on Advanced Information Networking Applications AINA) 2 , pp. 376–379.

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

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

[7] Das, M.L., Saxena, A., Gulati, V.P., Phatak, D.B., 2005. Hierarchical key management scheme using polynomial interpolation. Operating Systems Review 39 (1), 40–47.

[8] Diffie, W., Hellman, M., 1976. New directions in cryptography. IEEE Transactions on Information Theory IT 22 (6), 644–654.

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

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

[11] Hsu, C.L., Tsai, P.L., Chou, Y.C., 2008. Robust dynamic access control scheme in a user hierarchy based on one-way hash function.International Computer Symposium.

[12] Hwang, M.S., Yang, W.P., 2003. Controlling access in large partially-ordered hierarchies using cryptographic keys. Journal of Systems and Software 67 (2), 99–107.

[13] Jeng, F.G., Wang, C.M., 2006. An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem. The Journal of Systems and Software 79 (8), 1161–1167.

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

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

[16] Menezes, A.J., Oorschot, P.C., Vanstone, S.A., 1997. Handbook of Applied Cryptography. CRC Press Inc.