Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

A practical and dynamic key management scheme for a user hierarchy

JENG Fuh-gwo[1,2], WANG Chung-ming[1]

([1]*Institute of Computer Science, National Chung Hsing University, Taichung, Taiwan 402, China*)

([2]*Department of Applied Mathematics, National Chiayi University, Chiayi, Taiwan 60004, China*)

E-mail: fgjeng@mail.ncyu.edu.tw; cmwang@cs.nchu.edu.tw

Received Oct. 20, 2005; revision accepted Nov. 21, 2005

Abstract: In this paper, we propose a practical and dynamic key management scheme based on the Rabin public key system and a set of matrices with canonical matrix multiplication to solve the access control problem in an arbitrary partially ordered user hierarchy. The advantage is in ensuring that the security class in the higher level can derive any of its successor's secret keys directly and efficiently and show it is dynamic while a new security class is added into or a class is removed from the hierarchy. Even the ex-member problem can be solved efficiently. Moreover, any user can freely change its own key for some security reasons.

Key words: User hierarchy, Key management, Rabin public key, Matrix multiplication
doi:10.1631/jzus.2006.A0296 **Document code:** A **CLC number:** TP391

INTRODUCTION

The access control problem in an arbitrary partially ordered user hierarchy is defined below. In an organization, the users and their authorized data are organized into a group of disjoint sets of security classes, and each user is assigned to a certain security class called his security clearance. Let $C_1, C_2, \ldots, C_n$, $n\in\mathbb{N}$, be n disjoint security classes and '$\leq$' be a binary partial-order relation over the set $C=\{C_1, C_2, \ldots, C_n\}$. For the set $(C, \leq)$, $C_j\leq C_i$ $(i, j\in\mathbb{N})$ means that the users in the security class C_i have a security clearance higher than or equal to that in the security class C_j. In other words, the users in security class C_i can read or store the data held by the users in security class C_j, but the opposite is not allowed. Fig.1 shows an example of a partial-order hierarchy.

Note that the arrowhead in Fig.1 means the higher level security classes have a security clearance higher than that of the lower level classes. For the relation $C_j\leq C_i$, C_i is called a predecessor of C_j, and C_j a successor of C_i. Further, if $C_j\leq C_i$ and if there is no other security class C_k $(k\in\mathbb{N})$ such that $C_j\leq C_k\leq C_i$, then C_i is said to be an immediate predecessor of C_j, and C_j an immediate successor of C_i. Generally speaking, each user in security class C_i is assigned a secret key K_i. When he wants to store a data x into the database or broadcast it to the network, he first encrypts x by his secret key K_i to obtain $x'=E_{k_i}(x)$, then stores or broadcasts x'. Only users in possession of K_i are able to retrieve x by calculating $x=D_{k_i}(x')$. E and D are called the encryption and decryption algorithms, respectively. However, the key K_i is only used for encrypting or decrypting the database entitled to security class C_i. That means when a user in security class C_i, with a higher clearance than C_j, would like to retrieve data encrypted by K_j, he should get the right key K_j first.

In the real world, it is not difficult to conceive that examples of hierarchical access control are required. One is the personnel of a chain of department stores, where employees are grouped by their ranks into a partial-order hierarchy. Another is a hospital where only doctors with certain degree of seniority may have access to some personal information in a patient's medical record. Similar situations abound in other areas, particularly in the government and the

military, are easily envisaged. Moreover, consider a secure distributed system where hosts operate at different security levels and the encrypted data are broadcast to the network without concern for misrouting since the unintended recipients would be unable to decrypt the data.

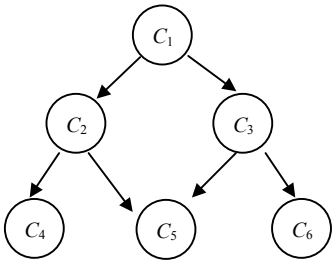

Fig.1 A partial-order hierarchy

Thus, how to find a key-generating mechanism such that each user owns a secret key and afterwards uses this key to retrieve all the information items he is entitled to is a job for all the researchers in this field. The first attempt to solve the problem by using cryptographic keys was proposed by Akl and Taylor (1983). They first assigned a public integer t_i to each security class C_i with the property that $t_i|t_j$ if and only if $C_j \leq C_i$. The cryptographic key K_i for C_i is computed by $K_i=K_0^{t_i} \pmod m$, where K_0 is the secret key of the central authority, responsible for generating and distributing the cryptographic keys and m is the public product of a pair of large secret prime numbers. Since t_j/t_i is an integer if $C_j \leq C_i$, C_i can derive K_j by the following deduction:

$$K_j = K_0^{t_j} = K_0^{t_i \cdot t_j / t_i} = (K_0^{t_i})^{t_j/t_i} = (K_i^{t_j/t_i}) \pmod m$$

Obviously, t_j/t_i is not an integer if C_j is not a successor of C_i and the key derivation will fail. However, the size of the public information t_i will grow dramatically when the increasing number of security classes becomes larger.

Hence, Mackinnon *et al*.(1985) proposed a canonical assignment method to reduce the t_i values. Later, Harn and Lin (1990) presented a bottom-up key generation scheme instead of the top-down design approach in (Akl and Taylor, 1983; Mackinnon *et al*., 1985) to improve it. In fact, the t_i values are really smaller than those in the Akl-Taylor scheme. Yet, these key assignment schemes (Akl and Taylor, 1983; Harn and Lin, 1990; Mackinnon *et al*., 1985) violate the security requirement whenever a new security class is added into the system as an immediate successor of an existing security class. In other words, all the issued keys should be re-generated when a new security class is added into the user hierarchy.

Several key assignment solutions are proposed (Chang *et al*., 1982; Chang and Buehrer, 1993; Tsai and Chang, 1995; Zhong, 2002) subsequently to achieve the target of dynamically adding or removing a security class, together with keeping the size of public information as small as possible. However, a big disadvantage of these schemes is that when the user with a higher security clearance wants to derive a secret key of his successor, but not an immediate one, he should iteratively implement the key derivation algorithm. Such an approach is quite inefficient.

The ex-member problem mentioned in (Zheng, 1993) is defined as that of a user who should have no privilege to access all the information entitled to him originally after his dismissal from the hierarchy. A trivial solution is re-generating the secret keys and secret information for all the security classes after a member dismissed from the hierarchy; however, it becomes undesirable if dismissal occurs quite frequently in the hierarchy.

Thus in this paper, a practical and dynamic key management scheme is proposed to achieve the goals of dealing with the ex-member problem flexibly, by deriving keys much more efficiently and practically, by adding or removing a security class more dynamically, and by protecting the secret keys from collaborating with one's successors. And more than these, each security class is able to choose its own key for its convenience or to change freely its secret key for some security reason without altering existing secret keys and less information is required to be updated.

PRACTICAL AND DYNAMIC KEY MANAGEMENT SCHEME

Before describing our key management scheme, we introduce Rabin public-key system (Rabin, 1979) first. The key point of Rabin system is to select a secret pair of large prime numbers (p, q) and to publish the number m where $m=pq$. The encryption procedure E is given by

$$C=E(M)=M(M+b) \pmod m. \quad (1)$$

M is a plaintext, C is a ciphertext, and b is a public integer. The decryption procedure D is to get solutions M from the following congruence:

$$M^2+Mb-C=0 \pmod m. \quad (2)$$

Since the number m is the product of two large prime numbers p and q, solving Eq.(2) is equivalent to solve both of the following congruences:

$$M^2+Mb-C=0 \pmod p, \quad (3)$$
$$M^2+Mb-C=0 \pmod q. \quad (4)$$

Thus, four possible solutions of M can be derived from Eqs.(3) and (4); that is,

$$M=(-b/2)\pm\sqrt{(b/2)^2+C} \pmod p,$$

and

$$M=(-b/2)\pm\sqrt{(b/2)^2+C} \pmod q.$$

To determine the true plaintext, we often patch some identifier onto the original plaintext M, for example, sender ID, receiver ID, data, or time, etc. In our scheme, we patch the secret key K with an identifier ID to substitute for the message M in the encryption function E such that Eq.(1) is modified as $C=(K||ID)(K||ID+b) \bmod m$, where '||' represents the concatenation operation. And later on, there is no doubt that the secret key K can be recovered correctly.

Obviously, the reason why Rabin system is regarded as one of the most promising public key cryptosystems is that its encryption function is a second-order polynomial over module m, the product of two large primes (p, q), and its decryption function is a square-root function over module p and q respectively. Thus, applying Rabin system makes our scheme elegant and practical in implementation. However, the security of Rabin system is based upon the computational difficulty of factoring a product of two large primes. If we review the decryption function D, where the secret primes p and q are revealed when a user wants to derive the secret key ($K||ID$). That means the system security is vulnerable if there is only one pair of primes. Thus, we will generate a distinct pair of primes for each security class so as to preserve the security that the system should originally have.

In order to make the key-derivation process be more efficient, we construct a set of matrices to maintain the relationship among all the security classes in the hierarchy. For security class C_i, we construct a matrix $\boldsymbol{V}_i$.

$$\boldsymbol{V}_i=(v_{xy})_{n\times 3}\times\begin{bmatrix} p_i & 0 & 0 \\ 0 & q_i & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where the jth row vector of $(v_{xy})_{n\times 3}$, $1\le x\le n$, $1\le y\le 3$, is defined as $[p_j\ q_j\ ID_j]$. Thus, by the definition of the matrix multiplication operation, the jth row vector of $\boldsymbol{V}_i$ is equal to $[p_jp_i\ q_jq_i\ ID_j]$. Obviously, the jth row vector of $\boldsymbol{V}_i$ is set to $[p_jp_i\ q_jq_i\ ID_j]$ if C_j is a successor of C_i; otherwise, it is [0 0 0] if C_j is not a successor of C_i. For example, the class C_1 in Fig.1 has the access rights to C_2, C_3, C_4, C_5, and C_6, so CA will construct a matrix $\boldsymbol{V}_1$ for C_1 as follows.

$$\boldsymbol{V}_1=\begin{bmatrix} 0 & 0 & 0 \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \\ v_{41} & v_{42} & v_{43} \\ v_{51} & v_{52} & v_{53} \\ v_{61} & v_{62} & v_{63} \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 \\ p_2 & q_2 & ID_2 \\ p_3 & q_3 & ID_3 \\ p_4 & q_4 & ID_4 \\ p_5 & q_5 & ID_5 \\ p_6 & q_6 & ID_6 \end{bmatrix}\times\begin{bmatrix} p_1 & 0 & 0 \\ 0 & q_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that the 2nd row vector of $\boldsymbol{V}_1$, $[v_{21}\ v_{22}\ v_{23}]$, is equal to $[p_2p_1\ q_2q_1\ ID_2]$ after the matrix multiplication operation. The purpose of constructing a matrix is to have each security class able to deduce directly his successor's secret pair of primes with the identifier and then to derive his successor's secret key subsequently. An efficient and practical strategy is implemented by the key generation and derivation algorithms shown follows.

Key generation algorithm

Step 1: Suppose there are n, $n\in\mathbb{N}$, security classes in a user hierarchy over the partially ordered relation. A central authority (CA) publishes the value b and dominates a pair of distinct prime numbers p_i and q_i for security class C_i where $1\le i\le n$.

Step 2: For each security class C_i, CA constructs

a secret matrix $\boldsymbol{V}_i=(v_{st})_{n\times 3}$, where the jth row vector of $\boldsymbol{V}_i$ is equal to $[p_j p_i\ q_j q_i\ ID_j]$ for all $C_j \le C_i$ where ID_j is the identifier of C_j; otherwise, it is [0 0 0] if C_j is not a successor of C_i.

Step 3: CA distributes (p_i, q_i, ID_i, $\boldsymbol{V}_i$) to security class C_i secretly.

Step 4: For each security class C_i, it freely chooses its own secret key K_i. However, the secret key concatenated with its own identifier ID_i should be in the range of [1, m_i, −1]; that is, $1 \le (K_i||ID_i) \le m_i - 1$, where $m_i=p_i q_i$. And then it calculates $X_i=(K_i||ID_i)\times(K_i||ID_i+b) \bmod m_i$, and sets X_i into the public information database.

End (of the key generation algorithm).

Key derivation algorithm

Assume a user u_i in the security class C_i wants to access the encrypted data held by the user u_j in one of his successor classes C_j, u_i can derive the secret key K_j of u_j by the following steps:

Step 1: Get b, X_j from the public database and the jth row vector of his own secret vector $\boldsymbol{V}_i$.

Step 2: Compute $p_j=v_{j1}/p_i$.

Step 3: Compute $q_j=v_{j2}/q_i$.

Step 4: Solve

$$(K_j||ID_j) = (-b/2) \pm \sqrt{(b/2)^2 + X_j} \pmod{p_j},$$

$$(K_j||ID_j) = (-b/2) \pm \sqrt{(b/2)^2 + X_j} \pmod{q_j}.$$

Step 5: Determine the actual key K_j from four solutions by the identifier ID_j shown in v_{j3}.

End (of the key derivation algorithm).

DYNAMIC PROPERTY OF OUR SCHEME

First, we consider the problem of addition of new security classes. Suppose a new security class C_k is added into an existing hierarchy. Only those security classes having the privilege to access the new class C_k have to update the corresponding kth row vector of their secret matrices. All the other secret keys or public information in the system do not have to be changed. For example, let C_7 be a new security class to be added as an immediate successor of C_2, as shown in Fig.2. Since C_1 and C_2 are the classes endowed with the privilege to access the new class, CA substitutes $[p_7p_1\ q_7q_1\ ID_7]$ for the 7th row vector of matrix $\boldsymbol{V}_1$ and $[p_7p_2\ q_7q_2\ ID_7]$ for the 7th row vector of matrix $\boldsymbol{V}_2$ and then re-distributes secretly the new $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ for C_1, and C_2 respectively after doing the key-assignment phase for C_7.

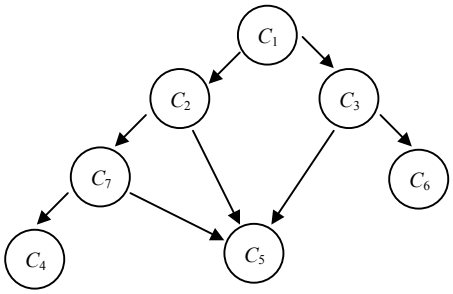

Fig.2 A class C_7 added to the hierarchy of Fig.1

Second, delete a security class from the hierarchy. It is so trivial since there is no relationship between some classes with the deleting one, the corresponding row vector of those security classes with the right to access the deleting class is set to be [0 0 0]. For example, C_2 is assumed to be a deleting class from the hierarchy in Fig.2, as shown in Fig.3. Since C_1 has the right to access C_2, CA will set the 2nd row vector of matrix $\boldsymbol{V}_1$ to be [0 0 0] to show there is no more privilege to access C_2.

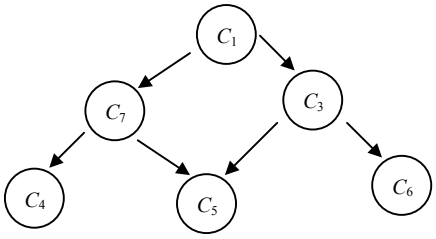

Fig.3 A class C_2 deleted from the hierarchy of Fig.2

Third, the addition of a relation between two classes is a process the same as the addition of a security class. For example, a new relation between C_3 and C_7 is added into the hierarchy, shown in Fig.4. Since C_7 is endowed with the right to access C_3, all the classes entitled to C_3 is no doubt entitled to C_7 consequently. Thus, CA only updates the matrix $\boldsymbol{V}_7$ as having the 3rd row vector to be $[p_3p_7\ q_3q_7\ ID_3]$ and the 6th row vector to be $[p_6p_7\ q_6q_7\ ID_6]$, and then re-distributes secretly the new $\boldsymbol{V}_7$ to C_7.

Fourth, the deletion of a relation between two classes is a process the same as the deletion of a security class. For example, a relation between C_3 and

C_5 is deleted from the hierarchy, shown in Fig.5. Since C_3 no longer has the right to access C_5, CA will set the 5th row vector of the matrix $\boldsymbol{V}_3$ to be [0 0 0].

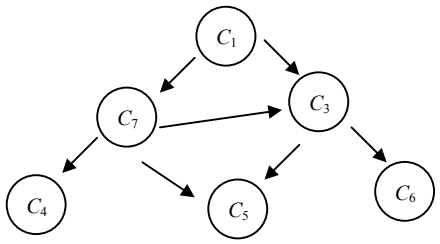

Fig.4 A new relation between C_3 and C_7 is added to Fig.3

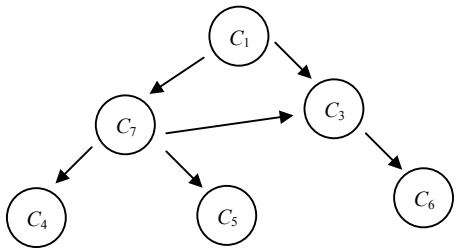

Fig.5 A relation between C_3 and C_5 is deleted from Fig.4

As to the ex-member problem, it is a problem similar to the problem of deleting a class. Consider a user u_i in the security class C_i is dismissed; so that he no longer has the right to access his own classes and all his successors as well. Thus, to ensure the security of the hierarchy, all the classes C_j, $C_j \leq C_i$, should change their secret keys. For example, assume a user u_3 in class C_3 is dismissed. In Fig.5 of our scheme in this case, C_3 and C_6 have to alter their keys and to re-publish X_3 and X_6 to the public database after CA passes them the new distinct pair of primes (p, q) and new identifiers. Finally, CA updates the corresponding matrices $\boldsymbol{V}_1$ and $\boldsymbol{V}_7$ so as to match the newly-generated primes and identifiers.

Changing a secret key for some security reason is sometimes necessary for practical implementation. Our strategy is let each security class have the freedom of key-choosing, which means that the security class can change its secret key freely as desired and only has to update its public information X in the database. For example, if the security class C_1 wants to change its key from K_1 to K_1', it republishes its public information X_1' onto the database as $X_1'=(K_1'||ID_1)((K_1'||ID_1)+b) \bmod m_1$; having nothing to do with the other keys or public information in the hierarchy.

SECURITY ANALYSIS AND DISCUSSIONS

Let us review the key derivation process. It is practical and flexible for any security class via its secret matrix $\boldsymbol{V}$ to derive any of its successor's keys directly, instead of iteratively. Yet the opposite is not allowed since each security class has no way to get the secret pair of primes of his predecessor.

Now, let us consider the problem of two or more users at a lower level of the system collaborating to derive a higher level key they are not entitled to. From Step 4 of the key derivation algorithm, we know the pair of primes (p, q) is essential for revealing the secret key K of a security class. However, each security class in the hierarchy is authorized only to access his own primes and be able to derive his successors' primes by way of his secret matrix, which means that he has no idea about the secret pair of primes of his predecessor. Thus, it is safe because the predecessor's secret key K cannot be revealed by his successors' collaboration.

As to the time complexity of our scheme, the kernel operations in our scheme for our key generation or derivation are based upon the Rabin public key cryptosystem and the canonical mathematic operations (+, −, ∗, /, module). The Rabin system is considered to be one of the most promising public key cryptosystems because its encryption function is a second-order polynomial over the module the product of two primes, and its decryption function is a square-root function over the module of each of the two primes. Thus, the time complexity for key-generation and key-derivation phases is focused on the modulus operations. Each of p and q to be a 10-digit number is secure for our scheme since the product m in the key generation algorithm will never show up again after the generation of the public information X. Note that the product m is an integer with 20 digits, which means the entire exhaustive searching space will be equal to 10^{20}. Assuming 10^{11} steps can be performed per day (i.e. about 1 step per μs), the entire computation would take about 10^9 days or several million years. Therefore, our scheme is computationally secure when both p and q are

10-digit numbers. Hence, the time complexity $O(LK)$ for key-generation phase can be reduced as $O(L)$, where L is the number of relations in the hierarchy and K is the length of the public key modules. And moreover, the time complexity for key-derivation phase is $O(1)$ since the successor's secret key can be derived directly whether he is an immediate successor or not. Therefore, the computation overhead of our scheme is quite low.

Finally, we would like to discuss the storage required in our scheme. For each security class, the secret information (p, q, $\boldsymbol{V}$, K, ID) and the public information X are required. The Rabin security is based on the computational difficulty of factoring a product of two 100-digit primes (Rabin, 1979); however, our scheme can avoid the attack of factoring m since the product m in the key generation algorithm will never show up again after the generation of the public information X. Thus, we said each of p and q to be a 10-digit number is secure for our scheme. Therefore, the storage required for the secret key K and the public information X, each with about 20 digits, is much smaller that that required in (Akl and Taylor, 1983; Chang *et al*., 1992; Chang and Buehrer, 1993; Harn and Lin, 1990; MacKinnon *et al*., 1985; Zhong, 2002). As to the secret matrix $\boldsymbol{V}$, we know most of the row vectors are [0 0 0] when the hierarchy is not strongly related. The storage required is clearly proportional to the number of the relations among the security classes; although each element in the matrix only requires 20 digits. Therefore, the total storage required is bound to $O(NL)$, where N is the number of the security classes in the hierarchy. Obviously, the requirement for storage is insignificant when compared to the other schemes based on the computational difficulty of factoring a product of two 100-digit primes.

CONCLUSION

A practical and dynamic cryptographic key management scheme for access control in a hierarchy is proposed. It is ensured that the conspiracy of one's successors cannot reveal their predecessor's secret key, and similarly cannot generate their sibling's secret key. The scheme is proven to be secure and have the following advantages.

(1) It is practical in both of the key generation and key derivation phases since the Rabin public key cryptosystem has the benefit of reducing the overhead on computation.

(2) It is efficient in the key-derivation process especially, since any user can derive any of its successor's key directly, instead of iteratively.

(3) It is dynamic as only a small set of related matrices has to be updated when a new class or a new relation is added into the hierarchy; an old class or an old relation is deleted from it; or a member is dismissed from the hierarchy.

(4) It is flexible on the key selection, since any security class can select its own secret key K for its own convenience, and can change its secret key from K to K' for some security reason; having nothing to do with the others in the hierarchy.

(5) It has insignificant storage requirement, since p and q with 10 digits each have made our scheme as secure as those schemes with 100 digits.

References

Akl, S.G., Taylor, P.D., 1983. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans*. *on Computer System*, **1**(3):239-247.

Chang, C.C., Buehrer, D.J., 1993. Access control in a hierarchy using a one-way trapdoor function. *Computers and Mathematics with Applications*, **26**(5):71-76. [doi:10.1016/0898-1221(93)90075-7]

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. [doi:10.1016/0306-4379(92)90015-F]

Harn, L., Lin, H.Y., 1990. A cryptographic key generation scheme for multilevel data security. *Computers & Security*, **9**(6):539-546. [doi:10.1016/0167-4048(90)90132-D]

MacKinnon, S.T., Taylor, P.D., Meijer, H., Akl, S.G., 1985. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Trans*. *Computers*, **C-34**(9):797-802.

Rabin, M.O., 1979. Digitalized Signatures and Public-key Functions as Intractable as Factorization. Technical Report, MIT/LCS/TR-212.

Tsai, H.M., Chang, C.C., 1995. A cryptographic implementation for dynamic access control in a user hierarchy. *Computers & Security*, **14**:159-166. [doi:10.1016/0167-4048(95)97049-G]

Zheng, Y., Hardjono, T., Seberry, J., 1993. New Solutions to the Problem of Access Control in a Hierarchy. Technical Report Preprint No.93-2, Department of Computer Science, University of Wollongong, Australia.

Zhong, S., 2002. A practical key management scheme for access control in a user hierarchy. *Computers & Security*, **21**(8):750-759. [doi:10.1016/S0167-4048(02)00815-5]