Cryptographic key assignment scheme for dynamic access control in a user hierarchy

F.H.Kuo, V.R.L.Shen, T.S.Chen and F.Lai

Abstract: The authors present a cryptographic key assignment scheme based on both the Rabin public key system and the Chinese remainder theorem, in order to solve dynamic access control problems in user hierarchies. Compared to earlier assignment schemes, the new scheme ensures that a security class can efficiently derive secret keys of its successors. Dynamic access control problems, such as adding/deleting classes, adding/deleting relationships, and changing secret keys, are considered. Moreover, without altering other keys present in the system, each user in the hierarchy can freely change his own key.

1 Introduction

In computer communication systems, both computer cryptography and information security have become important requirements. The unauthorised user is not allowed to access secret data, and such data must be well protected. Due to the growth of computer networks, and the fast progress of computer technologies on multi-user systems, sharing resources has become common, and the administration of these resources is important in a multi-user computer environment. In other words, access control through authorisation becomes important.

The access control problem in a computer environment can be described by a user hierarchy. In such a hierarchy, the users and their own information items are organised into a number of disjoint sets of security classes, and each user is assigned to a security class called the user's 'security clearance'. Let $SC_1, SC_2, \ldots, SC_n$ be n disjoint security classes. Assume that " $\leq$ " is a binary partially ordered relation on the set $SC = \{SC_1, SC_2, \ldots, SC_n\}$. Then, in the partially ordered set (poset) $(SC, \leq)$, $SC_j \leq SC_i$ means that the users in the security class SC_i have a security clearance higher than or equal to those in the security class SC_j. In other words, users in the security class SC_i can read and store information items that belong to users in the security class SC_j. In contrast, the users in SC_j cannot read or store information items in SC_i.

IEE Proceedings online no. 19990311

DOI: 10.1049/ip-cdt:19990311

Paper received 4th February 1999

F.-H. Kuo is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan

V.R.L. Shen is with the Department of Electrical Engineering, Mingchi Institute of Technology, P.O. Box 3-28, Hsinchuang City, Taipei 243, Taiwan
E-mail: rlshen@ccson.mit.edu.tw

T.S. Chen is with the Department of Computer Science and Information Engineering, Da Yeh University

F. Lai is with the Department of Electrical Engineering and Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

Fig. 1 shows an example of a poset connected to a user hierarchy. The arrowhead in Fig. 1 indicates that the higher level security classes have a security clearance higher than the lower level ones. $SC_j \leq SC_i$ means that SC_i is the predecessor of SC_j; and SC_j a successor of SC_i. Furthermore, if there is no security class SC_k in SC such that $SC_j \leq SC_k \leq SC_i$, then SC_i is called an *immediate predecessor* of SC_j and SC_j an *immediate successor* of SC_i.

In many application areas there is a need to carry out classified data management, so access control problems become very important. Access control in a hierarchy can be used not only in military and government departments but also in private corporations. Key assignment schemes are for instance applicable in database management systems [1–3], computer networks [4, 5] and operating systems [6].

The main problem is how the user in a security class can derive the secret key for users in a lower security class. Obviously, the easiest way to accomplish this is that the users keep all the keys of their successors in a database. As shown in Fig. 1, if the user in SC_1 wants to retrieve the data which are encrypted under his own key or his successors' keys, he needs a set of secret keys $\{SK_1, SK_2, SK_3, SK_4, SK_5, SK_6\}$. However, when the hierarchy becomes quite large, the users with the higher privileges need a great number of secret keys, and the quantity of keys thus becomes inconvenient to administer and hazardous to keep secure. Therefore, our goal is to find a method in which each user only needs to hold one key to retrieve all information which he is entitled to access, and to still retain the security of the system.

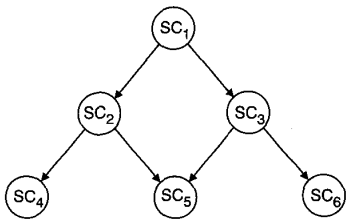

Fig. 1 *The poset in a user hierarchy*

When a user in security class SC_i wants to store the plaintext M into the database or broadcast it to the network, he first uses his secret key SK_i to encrypt M, then he will store or broadcast the ciphertext $C = E_{SKi}(M)$. Only the users with SK_i can get M by calculating $M = D_{SKi}(C)$, where E and D represent the prepublished encryption and decryption algorithms, respectively. For $SC_j \leq SC_i$, SC_i can adopt his secret key SK_i and SC_j's public information to derive SC_j's secret key SK_j, and then reads the information items possessed by SC_j.

Our scheme is composed of *Rabin's cryptosystem* and the *Chinese remainder theorem* (CRT). Rabin's cryptosystem is provably as secure as factoring [7], and the CRT has been shown to increase the deciphering speed in RSA public-key cryptosystem fourfold [8]. This motivates us to propose the novel scheme which has the following properties.

1. Key generation and key derivation algorithms are simple.
2. Dynamic access control problems, such as adding/deleting classes, adding/deleting relationships, and changing secret keys, can be overcome easily.
3. Users can change their secret keys easily.
4. The system can withstand the attack of collusion.

2 Review of previous research

Under the property that $t_i | t_j$ (i.e. t_j is divisible by t_i) if and only if $SC_j \leq SC_i$, Akl and Taylor [9] first proposed that each security class SC_i was assigned a public integer t_i. Then the secret key SK_i of SC_i is computed according to $SK_i = SK_0^{ti}$ (mod m), where SK_0 is the secret key of central authority (CA) and m is a product of a secret pair of large prime numbers. If $SC_j \leq SC_i$, t_j/t_i is an integer and SC_i can derive SK_j by the deduction:

$$SK_j = SK_0^{tj} = SK_0^{ti*(tj/ti)} = SK_i^{(tj/ti)} (\text{mod } m).$$

On the contrary, if SC_j (not $\leq$) SC_i, then t_j/t_i is not an integer and the key derivation will fail. However, the size of the public information t_i will increase dramatically when the number of security classes becomes large.

The advantage of Akl and Taylor's scheme is that the key generation and key derivation algorithms are quite simple; however, several drawbacks exist. First, when the number of security classes in the hierarchy is large, a large amount of storage is required for the public parameters. Secondly, as to the dynamic access control problems, such as adding/deleting classes or relationships in the existing system, the whole system has to be reestablished. Third, it is rather difficult to change the secret key.

Hence, MacKinnon *et al* proposed an improved method to reduce the values of t_i by using a canonical assignment method [10]. Later, Harn *et al* presented another method which uses a bottom-up key generation scheme to replace the top-down one [9, 11]. As a result, the values of t_i indeed become smaller than those in the Akl-Taylor scheme.

Whenever a new security class is added into the user hierarchy system, the schemes mentioned above cannot satisfy the security requirements. Unless the issued keys are changed, it is impossible to add a security class to the system.

Therefore, several key management schemes are proposed to solve the dynamical adding or deleting of the security class, and to keep the size of public information items as small as possible [12–15]. In these schemes, the user with higher security clearance must *iteratively* perform the key derivation process for deriving the secret key of the user who is not the immediate successor. And the key derivation is inefficient. Consequently, some other schemes, proposed to extend the Akl-Taylor scheme, allow the users to change their keys and to derive any secret keys belonging to the successors efficiently [16, 17].

Meanwhile, each aforementioned scheme allows any security class to derive the keys of his successors from his own secret key, and to prevent the possibility of collaboration by the successors to derive his own secret key.

3 Proposal for a new scheme

Based on Rabin's public key system and the CRT, our scheme is intended to solve dynamic access control problems in user hierarchies.

3.1 Rabin's public key system

In Rabin's public key system [18], there are a pair of public keys (b, m) for encryption, and a pair of secret keys (p, q) and b for decryption. The key point for Rabin's cryptosystem is to select a pair of secret and large prime numbers (p, q) and to let $m = p*q$. The encryption procedure E is shown as follows:

$$C = E(M) = M(M + b) \bmod m, \tag{1}$$

where M is a plaintext, and C is a ciphertext. The decryption procedure D is to obtain solutions for M through the following congruence:

$$M^2 + Mb - C = 0 \bmod m. \tag{2}$$

Since the module m is equal to $p*q$, eqn. 2 will be equivalent to the following two congruences:

$$M^2 + Mb - C = 0 \bmod p, \tag{3}$$

$$M^2 + Mb - C = 0 \bmod q. \tag{4}$$

Since eqns. 3 and 4 are quadratic congruences, the plaintext M can thus be recovered through the following four square roots modulo pq:

$$M = \frac{-b}{2} + \sqrt{\left(\frac{-b}{2}\right)^2 + C} \bmod p,$$

$$M = \frac{-b}{2} - \sqrt{\left(\frac{-b}{2}\right)^2 + C} \bmod p,$$

$$M = \frac{-b}{2} + \sqrt{\left(\frac{-b}{2}\right)^2 + C} \bmod q,$$

$$M = \frac{-b}{2} - \sqrt{\left(\frac{-b}{2}\right)^2 + C} \bmod q.$$

Obviously, Rabin's system is popular in the public key cryptosystems. The *encryption function* of the system is based on a second-order polynomial, which computes the modulus of the product of two large primes (p, q); and the *decryption function* on a square-root function, which computes the modulus of one of these two primes. Thus, Rabin's system makes our scheme elegant and practical in computation.

If a user wants to derive the secret key SK, first he must own the secret data p and q of the decryption procedure D. In other words, the security of the system, based on the

difficult computation of the factoring of the product of two large primes, will be broken when one pair of primes (p, q) exists. Therefore, in order to keep the security of the system, we need to give each security class a distinct pair of primes. Dirichlet's theorem [19] thus offers the distinct pair of primes as follows:

Let $b \geq 1$, then there will exist infinite integers $r \geq 1$ such that $r*2^b + 1$ is one prime, and also exist infinite integers $r' \geq 1$ such that $r'*2^b - 1$ is the other prime.

In order to generate its own secret primes p and q under the security of the system, each security class should hold two secret information items (r, r') and a public datum b. In other words, when a user U_i with a higher security clearance wants to decrypt the information items held by his successor U_j, the secret data (r_j, r_j') of U_j should first be obtained; then p_j and q_j, which perform the decryption function, can be derived. As for U_i, holding all his successors' information items (r, r') is as awkward and insecure as holding all his successors' keys. Consequently, we include the CRT to overcome this difficulty.

3.2 Chinese remainder theorem (CRT)

Let $n_1, n_2, \ldots, n_t$ be pairwise relative primes, i.e. $\gcd(n_i, n_j) = 1$, for $i \neq j$. Let $r_1, r_2, \ldots, r_t$ be integers. Then the congruent equation is of the form: $H \equiv r_i (\text{mod } n_i)$.

Let $N = n_1\, n_2\, \ldots\, n_t$. There is only a unique solution H between 0 and $N - 1$. The solution of H is of the form:

$$H = \sum_{i=1}^{t} r_i X_i (\text{mod } N)$$

where $X_i = y_i*(N/n_i)$ and y_i satisfies $y_i*(N/n_i) \equiv 1 \ (\text{mod } n_i)$.

3.3 Proposed scheme

In order to find the true plaintext, we propose adding some information into the original plaintext M, such as *sender identity code, receiver identity code, date*, and *time*. In our scheme, the secret key SK concatenated with the specific identity code (ID) is used to replace the message M in the above encryption procedure E, so eqn. 1 is modified to

$$C = (SK \| ID)((SK \| ID) + b) \bmod m,$$

where "$\|$" represents the concatenation operation. Undoubtedly, the secret key SK can be recovered correctly. Also, a 4:1 ambiguity in the decrypted messages [7] can be avoided in this manner.

3.3.1 Key generation algorithm.

We suppose that the existing system has n security classes $SC_1, SC_2, \ldots,$ and SC_n in the user hierarchy. The algorithm of generating the secret key SK_i for each security class SC_i is stated as follows:

Step 1: Randomly select n coprime integers $n_1, n_2, \ldots, n_n$ to be as small as possible.

Step 2: Compute $N = \prod_{i=1}^{n} n_i$.

Step 3: Take a node SC_i from the hierarchy by preorder traversal.

Step 4: CA (i.e. Central Authority) randomly selects three positive integers r_i, r_i' and b_i for security class SC_i such that $(r_i*2^{bi} + 1)$ and $(r_i'*2^{bi} - 1)$ are equal to prime numbers p_i and q_i, respectively.

Step 5: CA chooses y_i such that $(N/n_i)*y_i \bmod n_i = 1$ for $1 \leq i \leq n$, and lets $X_i = (N/n_i)*y_i$. CA then computes each security class SC_i's derivation information

$$H_i = \Sigma_j (r_j \| r_j')*X_j - (r_i \| r_i')*X_i (\text{mod } N) \text{ for } SC_j \leq SC_i$$

Step 6: When the security class SC_i receives the data of (r_i, r_i', X_i, H_i) and its own public parameters (b_i, n_i), CA performs the following steps:

6.1 Compute $p_i = (r_i*2^{bi} + 1)$, $q_i = (r_i'*2^{bi} - 1)$, let $m_i = p_i*q_i$, and keep p_i, q_i, and m_i secret.

6.2 Choose one secret key SK_i for convenience, then concatenate the secret key SK_i with its identity code ID_i, and let its range be in $[1, m_i]$, In other words, let M_i be equal to SK_i concatenated with ID_i, i.e. $M_i = SK_i \| ID_i$, where $1 \leq M_i \leq m_i - 1$.

6.3 Calculate the public information $C_i = (M_i)* (M_i + b_i) \bmod m_i$.

Step 7: Repeat steps 3 to 7 until all nodes of the hierarchy are traversed.

3.3.2 Key derivation algorithm.

By the SC_i successors' public parameters, SC_i can derive any successor SC_j's secret key SK_j, then, by SK_j, SC_i decrypts the information items possessed by SC_j. The algorithm to derive the secret key SK_j is stated as follows:

Step 1: Compute $(r_j \| r_j') = H_i \bmod n_j$.

Step 2: Compute $p_j = r_j*2^{bj} + 1$; $q_j = r_j'*2^{bj} - 1$.

Step 3: Due to the same reasons as in Section 3.1, compute

$$M_j = \frac{-b_j}{2} + \sqrt{\left(\frac{-b_j}{2}\right)^2 + C_j} \bmod p_j,$$

$$M_j = \frac{-b_j}{2} - \sqrt{\left(\frac{-b_j}{2}\right)^2 + C_j} \bmod p_j,$$

$$M_j = \frac{-b_j}{2} + \sqrt{\left(\frac{-b_j}{2}\right)^2 + C_j} \bmod q_j,$$

$$M_j = \frac{-b_j}{2} - \sqrt{\left(\frac{-b_j}{2}\right)^2 + C_j} \bmod q_j.$$

Step 4: According to the identity code ID_j, extract SK_j from the true plaintext M_j.

4 Dynamic access control

This section presents the dynamic access control problems, including adding/deleting classes, adding/deleting relationships, and changing secret keys.

4.1 Adding classes

To add a new security class SC_{n+1} into an existing hierarchy, CA must update the H_i of those security classes SC_i which serve as predecessors. All other keys in the system can stay the same. Suppose that a new security class SC_{n+1} is added to the existing system. CA performs the following steps:

Step 1: Select a new prime number n_{n+1}.

Step 2: $N = N*n_{n+1}$.

Step 3: Calculate y_{n+1} such that $(N/n_{n+1})* y_{n+1} \bmod n_{n+1} = 1$, and let $X_{n+1} = (N/n_{n+1})* y_{n+1}$.

Step 4: Randomly select three positive integers r_{n+1},

r'_{n+1} and b_{n+1} such that

$$p_{n+1} = (r_{n+1} * 2^{bn+1} + 1),$$

$$q_{n+1} = (r'_{n+1} * 2^{bn+1} - 1),$$

and let $m_{n+1} = p_{n+1} * q_{n+1}$.

Step 5: For all the predecessors SC_i of SC_{n+1}, after CA adds the new security class SC_{n+1}

$$H_i = H_i + (r_{n+1} \| r'_{n+1}) * X_{n+1} (\text{mod } N).$$

Step 6: Derive $H_{n+1} = \Sigma_{j \in J} (r_j \| r'_j) * X_j - (r_{n+1} \| r'_{n+1}) * X_{n+1} (\text{mod } N)$, where $J = \{j | (SC_j \leq SC_{n+1})$ AND NOT $(SC_j \leq SC_i)\}$.

Step 7: Compute $C_{n+1} = (M_{n+1}) * (M_{n+1} + b_{n+1})$ (mod m_{n+1}), where $M_{n+1} = SK_{n+1} \| ID_{n+1}$.

Example 1: Let SC_7 be a new security class which is added as a successor of SC_1, as shown in Fig. 2. Since only SC_1 has the access right to the new class SC_7, CA needs to reconstruct H_1 of the security class SC_1 after the secret information X_7 of SC_7 has been generated. CA will select these parameters (n_7, y_7, r_7, r'_7, b_7) and recompute the following equations.

$$N = N * n_7, X_7 = (N/n_7) * y_7$$

$$p_7 = (r_7 * 2^{b7} + 1), q_7 = (r'_7 * 2^{b7} - 1), m_7 = p_7 * q_7$$

$$H_1 = H_1 + (r_7 \| r'_7) * X_7 (\text{mod } N)$$

$$H_7 = [(r_7 \| r'_7) * X_7 + (r_6 \| r'_6) * X_6 - (r_7 \| r'_7) * X_7] (\text{mod } N)$$

$$C_7 = (SK_7 \| ID_7) * ((SK_7 \| ID_7) + b_7) (\text{mod } m_7)$$

4.2 Deleting classes

CA will perform the following steps to delete a security class SC_k in the existing system. For all the predecessors SC_i of SC_k, CA needs to have to recompute N and H_i. Then, CA will discard the secret data and public parameters of SC_k.

Step 1: $N = N/n_k$.

Step 2: For all the predecessor SC_i of SC_k, before CA deletes the SC_k,

$$H_i = H_i - (r_k \| r'_k) * X_k (\text{mod } N).$$

Step 3: Discard the secret data and public parameters of SC_k.

Example 2: Suppose that SC_3 in Fig. 1 is deleted from the hierarchy; the resulting hierarchy is shown in Fig. 3. CA will recompute the following equations.

$$N = N/n_3, H_1 = H_1 - (r_3 \| r'_3) * X_3 (\text{mod } N)$$

Then, CA will discard the secret data (r_3, r'_3, X_3, H_3) and the public parameters (b_3, n_3).

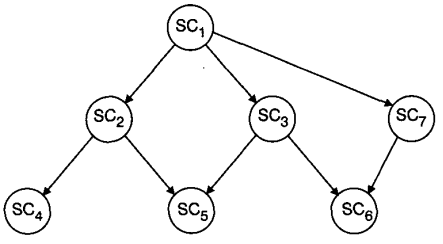

Fig. 2 *Adding a new security class SC_7*

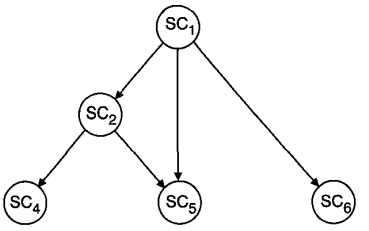

Fig. 3 *Deleting the security class* SC_3

4.3 Adding a relationship

Suppose that a new relationship between SC_b and SC_a is added into a user hierarchy such that $SC_b \leq SC_a$. CA only recomputes the public parameters $H'_i s$ for $SC_b \leq SC_i$, while other parameters remain unchanged. Then, after CA adds the relationship of $SC_b \leq SC_a$, it will perform the following procedure:

Step 1: For $SC_b \leq SC_a$ and $SC_j \leq SC_b$, compute

$$H_a = H_a + (r_b \| r'_b) X_b + \Sigma_{j \in J} (r_j \| r'_j) X_j (\text{mod } N),$$

where $J = \{j | (SC_j \leq SC_b)$ AND NOT $(SC_j \leq SC_a)\}$.

Step 2: For all the security classes SC_i that have the relationship $SC_a \leq SC_i$, do

$$H_i = H_i + \Sigma_{j \in J} (r_j \| r'_i)^* X_j (\text{mod } N),$$

where $J = \{j | (SC_j \leq SC_a)$ AND NOT $(SC_b \leq SC_i)$ AND NOT $(SC_j \leq SC_i)\}$.

Example 3: Suppose that a new relationship is added between SC_5 and SC_6 in Fig. 1, and $SC_6 \leq SC_5$; then the resulting hierarchy is shown in Fig. 4. {SC_1, SC_2, SC_3, SC_5} all have the access right to SC_6; therefore, CA will compute:

$$H_5 = H_5 + (r_6 \| r'_6)^* X_6 (\text{mod } N)$$

$$H_2 = H_6 + (r_6 \| r'_6)^* X_6 (\text{mod } N)$$

In the updated hierarchy, SC_5 can derive the secret key of SC_6. Since $SC_5 \leq SC_2$ before the updating, SC_2 can have the access right to SC_6 after the relationship $SC_6 \leq SC_5$ is added.

4.4 Deleting a relationship

Suppose that a relationship between SC_b and SC_a is deleted from a user hierarchy, then the relationship of $SC_b \leq SC_a$ will not exist. CA only recomputes the public parameters $H'_i s$ for $SC_b \leq SC_i$, while other parameters remain unchanged. Before CA deletes the relationship of $SC_b \leq SC_a$, it will perform the following procedure:

Step 1: For $SC_b \leq SC_a$ and $SC_j \leq SC_b$,, compute

$$H_a = H_a - (r_b \| r'_b) X_b - \Sigma_{j \in J} (r_j \| r'_j) X_j (\text{mod } N),$$

where $J = \{j | (SC_j \leq SC_b)$ AND NOT $(SC_j \leq SC_a)\}$.

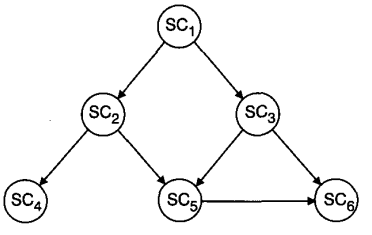

Fig. 4 *The resulting hierarchy of adding the relationship $SC_6 \leq SC_5$*

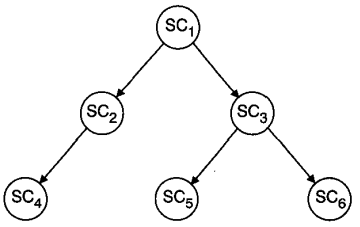

Fig. 5 *The resulting hierarchy of deleting the relationship $SC_5 \leq SC_2$*

Step 2: For all the security classes SC_i that have the relationship $SC_a \leq SC_i$, do

$$H_i = H_i - \Sigma_{j \in J}(r_j \| r'_j)^* X_j (\text{mod } N),$$

where $J = \{j | (SC_j \leq SC_a)$ AND NOT $(SC_b \leq SC_i)$ AND NOT $(SC_j \leq SC_i)\}$.

Example 4: Suppose that the relationship $SC_5 \leq SC_2$ in Fig. 1 is deleted, the resulting hierarchy is shown in Fig. 5. CA only needs to compute:

$$H_2 = H_2 - (r_5 \| r'_5)^* X_5 (\text{mod } N)$$

and other parameters remain unchanged. SC_2 cannot derive the secret key of SC_5 in the updated hierarchy. It means that SC_2 no longer has the access authority to SC_5's information items.

4.5 Changing a secret key

When a user requests CA to change his secret key, CA must derive all the users' secret keys. In the security classes, we are able to change our secret keys freely in the issued system, and CA only needs to update the public information C of the class. For example, if the key SK_i of the security class SC_i is changed to SK'_i, the public information C_i only needs to be updated as $C'_i = (SK'_i \| ID_i)((SK'_i \| ID_i) + b_i) \bmod m_i$, and all other keys or information items need not be changed.

5 Security analysis and discussions

In this section, we present the security, the efficiency of our scheme, and the functional comparisons in sequence.

5.1 Security

As shown in the key derivation process, any security class with its secret information H can derive his successors' keys easily; the opposite of this is not allowed because each security class only knows the public information b, n, and C of its predecessor. For example, as shown in Fig. 1, if the SC_3 wants to use a single key to retrieve information from all his successors SC_5 and SC_6, SC_3 must firstly obtain H_3. Since H_3 consists of H_5 and H_6, which are made up of $(r_5 \| r'_5)$ and $(r_6 \| r'_6)$, respectively; SC_3 can thus derive SK_5 and SK_6 by using $(r_5 \| r'_5)$ and $(r_6 \| r'_6)$. Certainly, SC_3 is able to use a single key to retrieve information from all his successors SC_5 and SC_6.

If two or more users at a lower level in the system collaborate to derive a higher level key, from Step 5 of the key generation algorithm, we understand that all successors of some security class can only collaborate to obtain H of the class by using all of their own (r, r') and X. Even if they obtain their predecessor's H, it cannot provide them with enough information to reveal the secret key because they can only know what they own. Thus, only under the disclosure of its corresponding data (r, r') can the predecessor's secret key SK be revealed.

Through Dirichlet's theorem on primes, there are infinite integers that we can choose for r and r' when $b \geq 1$. Hence, what we are concerned about is how to choose these two integers r and r' in our scheme. Let r and r' be integers of 10 digits, thus the entire exhaustive searching space needs 10^{20} digits for m. Suppose that there are 10^{11} digits being searched per day, the entire computation would take about 10^9 days. Therefore, the security of our scheme is determined by the careful selection of (r, r'). Furthermore, a small choice for modulus n is possible since this value is kept secret in this scheme unlike Rabin's public key scheme where the modulus is public.

5.2 Efficiency

Each security class needs to store the secret information (r, r', X, H, SK), and the public information (b, n, C). Compared with other schemes [9–11, 20], our scheme seems to need more storage for storing b, r, r', X, n, and H. However, the whole storage required for actual practice is small.

First, by the security classes or CA, (r, r') and b's values can be randomly selected. In order to reduce the values of the primes (p, q), the values should be selected to be as small as possible. But, it may be argued that the primes (p, q), generated by the small (r, r') and b of 10-digit, will make the scheme insecure because Rabin's scheme is based on the difficult computation of factoring the product of two primes of 100-digit [4]. However, from the key generation and derivation algorithms, we know that m will not show up any more after generating the public information C. This means that our scheme is secure enough to avoid the attack of factoring m by using m as a 20-digit number. Therefore, the storage space of secret key SK and the public information C, where SK and C are 20 digits, is much smaller than that needed in [9–11, 13, 15, 20].

Next, the storage of an equation H is proportional to the number of the security class SC's successors. However, it is a time-space tradeoff in practice. We generate H for the non-leaf security class so that the key derivation process will become more efficient, and the derivation path will be found by deriving the successors' key step by step. Let k be the number of the security class SC's successors, then the storage needed is about $k * \lceil \log P \rceil$ bits for H. Therefore, the total storage required is also small.

Finally, in our key generation or derivation and the insertion of new security classes, both the computation of Rabin's public key cryptosystem and the construction of the equation H are the key steps. In Rabin's system, encryption function is a second-order polynomial that computes the modulus of the product of two primes, and the decryption function is a square-root function that computes the modulus of one of the two primes. Thus, it is easy to see that the computational complexity of the key assignment dominated the cost of finding y_i's values in the equation H. Knuth [21] shows that the time complexity of the finding of y_i is $O(\log^3 n_i)$.

5.3 Functional comparisons

In Table 1, we list the functions of our scheme compared with the previously proposed schemes.

Table 1: Functional comparisons

Comparison items	AT	HL	WH	PRO
Hierarchy type	poset	poset	poset	poset
Key deriving method	direct	direct	direct	indirect
Key assignment complexity	exponential operation	exponential operation	CRT + one-way function	CRT + one-way function
Key deriving complexity	exponential operation	exponential operation	modular operation	modular operation + square root
Adding/deleting classes	updating entire parameters	updating entire parameters	updating entire parameters	updating partial parameters
Adding/deleting relationship	updating entire parameters	updating entire parameters	updating partial parameters	updating partial parameters
Changing secret key	updating entire parameters	updating entire parameters	updating entire parameters	updating partial parameters
Storage for public information	large	large	large	fixed and small

Remark: AT: Akl and Taylor's scheme [9]; HL: Harn and Lin's scheme [11]; WH: Wu and He's scheme [17]; PRO: our proposed scheme.

6 Conclusion

For an arbitrary poset user hierarchy, we have presented a dynamic access control scheme. It can ensure that the predecessor's secret key will not be revealed by conspiracy on the part of any successors. In other words, our scheme has been shown to be secure.

The proposed scheme contains the following three characteristics:

(1) The key generation and derivation algorithms are simple. For the user, the successors' keys can be derived directly rather than stepwise.
(2) The updating of the existing keys can be avoided when a new class is added into the system or an old one is deleted from the system.
(3) Since any security class can select its own secret key for convenience and change its secret key for security reasons, the key selection becomes more flexible.

7 Acknowledgments

The authors would like to thank the anonymous referees for their constructive comments which improved the quality of this paper. This work was supported in part by the National Science Council under grant NSC 88 - 2516 - S131 - 003.

8 References

1 DAVIDA, G.I., WELLS, D.L., and KAM, J.B.: 'A database encryption system with subkeys', *ACM Trans. Database Syst.*, 1981,**6**, (2), pp. 312–328
2 DENNING, D.E.: Cryptography and Data Security, Chap. 2 (Addison-Wesley, Massachusetts, 1983)
3 DENNING, D.E., AKL, S.G., MORGENSTERN, M., and NEUMANN, P.G.: 'Views for multilevel database security', Proc. 1986 IEEE Symp. on *Security and Privacy*, 7-9, April 1986, Oakland, CA, pp. 156–172
4 McCULLOUGH, D.: 'Specifications for multilevel security and a hook-up property', Proc. 1987 IEEE Symp. on *Security and Privacy*, 27-29, April 1987, Oakland, CA, pp. 161–166
5 McHUGH, J., and MOORE, A.P.: 'A security policy and formal top level specification for a multilevel secure local area network', Proc. 1986 IEEE Symp. on *Security and Privacy*, 7-9, April 1986, Oakland, CA, pp. 34–39
6 FRAIM, L.J.: 'Scomp: a solution to multilevel security problem', *IEEE Comput.*, 1983, pp. 26–143
7 WILLIAMS, H.C.: 'A modification of the RSA public-key encryption procedure', *IEEE Trans. Inf. Theory*, 1980, **IT-26**, (6), pp. 729–729
8 QUISQUATER, J.J., and COUVREUR, C.: 'Fast decipherment algorithm for RSA public-key cryptosystem', *Electron. Lett.*, **18**, pp. 905–907, 1982
9 AKL, S.G., and TAYLOR, P.D.: 'Cryptographic solution to a problem of access control in a hierarchy', *ACM Trans. Comput. Syst.*, 1983, **1**, (3), pp. 239–247
10 MacKINNON, S.T., TAYLOR, P.D., MEIJER, H., and AKL, S.G.: 'An optimal algorithm for assigning cryptographic keys to control access in a hierarchy', *IEEE Trans. Comput.*, 1985, **C-34**, (9), pp. 797–802
11 HARN, L., and LIN, H.Y.: 'A cryptographic key generation scheme for multilevel data security', *Comput. Secur.*, 1990, **9**, pp. 539–546
12 CHANG, C.C., and BUEHRER, D.J.: 'Access control in a hierarchy using a one-way trapdoor function', *Comput. Math. with Appl.*, 1993, **26**, (5), pp. 71–76
13 CHANG, C.C., HWANG, R.J., and WU, T.C.: 'Cryptographic key assignment scheme for access control in a hierarchy', *Inf. Syst.*, 1992, **17**, (3), pp. 243–247
14 CHICK, G.C., and TAVARES, S.E.: 'Flexible access control with master keys', *in* 'Advances in Cryptology - CRYPTO'89, Springer-Verlag, 1990, pp. 316–322
15 SANDHU, R.S.: 'Cryptographic implementation of a tree hierarchy for access control', *Inf. Proc. Lett.*, 1988, **27**, pp. 95–98
16 TSAI, H.M., and CHANG, C.C.: 'A cryptographic implementation for dynamic access control in a user hierarchy', *Comput. Secur.*, 1995, **14**, (2), pp. 857–959
17 WU, T.C., WU, T.S., and HE, W.H.: 'Dynamic access control scheme based on the Chinese remainder theorem', *Comput. Syst. Sci. Eng.*, 1995, **10**, (2), pp. 92–99
18 RABIN, M.O.: 'Digitized signatures and public-key function as intractable as factorization'. Technical Report, MIT/LCS/TR-212, MIT Lab. for Computer Science, 1979
19 RIBENBOIM, P.: 'The Book of Prime Number Records', 2nd edn., (Springer-Verlag, 1989)
20 LAIH, C.S., and HWANG, T.L.: 'A branch oriented key management solution to dynamic access control in a hierarchy', *IEEE Trans. Softw. Eng.*, 1991, **17**, (3), pp. 422–429
21 KNUTH, D.E.: 'The Art of Computer Programming, Seminumerical Algorithms', 2nd edn., (Addison-Wesley, Reading, MA, 1981)