ELSEVIER

The Journal of Systems and Software 67 (2003) 99–107

The Journal of Systems and Software

www.elsevier.com/locate/jss

# Controlling access in large partially ordered hierarchies using cryptographic keys

Min-Shiang Hwang [a,*], Wei-Pang Yang [b]

[a] Department of Information Management, Chaoyang University of Technology, 168, Gifeng E.Rd., Wufeng, Taichung County, 413 Taiwan, ROC
[b] Department of Computer and Information Science, National Chiao Tung University, Hsinchu, 300 Taiwan, ROC

Received 17 August 2000; received in revised form 6 December 2001; accepted 30 April 2002

**Abstract**

The problem of access control in a hierarchy is present in many application areas. Since computing resources have grown tremendously, access control is more frequently required in areas such as computer networks, database management systems, and operating systems. Many schemes based on cryptography have been proposed to solve this problem. However, previous schemes need large values associated with each security class. In this paper, we propose a new scheme to solve this problem achieving the following two goals. One is that the number of keys is reduced without affecting the security of the system. The other goal is that when a security class is added to the system, we need only update a few keys of the related security classes with simple operations.


*Keywords:* Access control; Cryptography; Data security; Hierarchy; Key management

## 1. Introduction

Many social organizations are based on hierarchical structures, for example, the military, government organizations, school systems, private corporations, computer management systems (Lu and Sundareshan, 1992; Lu and Sundareshan, 1988; McCullough, 1987; McHugh and Moore, 1986), operation systems (Friam, 1983; Mcilroy and Reeds, 1992), and database management systems (Denning, 1984; Denning et al., 1986). In the Bell–LaPadula model (Bell and LaPadula, 1975), a subject is allowed a read access to an object only if its security class is greater than or equal to the corresponding security class of the object. A subject is allowed write access to an object only if its security class is less than or equal to that of the object.

An example of a four-level hierarchical structure is shown in Fig. 1, with top, second, third and bottom levels. The top level (i.e., a general or a president) possesses the greatest authority. The bottom level (i.e., workers or employees) has the least authority. Now, assume that the user, in the security class 1000 ($C_{1000}$) in Fig. 1, encrypts a message with his/her own privacy key ($K_{1000}$). Because of the access control in a hierarchical structure, only the user (in $C_{1000}$) and his/her direct superiors (i.e., $C_7$, $C_3$, $C_1$) holding the key, $K_{1000}$, can decipher this message. Obviously, each user must hold a set of keys. Table 1 shows the keys held by each user in the hierarchy shown in Fig. 1. In this case, we need a key management scheme to manage those keys.

One way to solve this problem is to use a master key instead of many subordinate keys. Many authors have proposed methods of solving this problem based on the concept of the master key. The first of these methods, proposed by Akl and Taylor (1983), is based on cryptography to control access to any information among a group of users in a hierarchy. In such a hierarchy, the users and their own information items are divided into a number of disjointed sets of security classes, $C_1, C_2, \ldots, C_n$. Assume that a binary relationship "$\leqslant$" partially orders the set of security class, $C = \{C_1, C_2, \ldots, C_n\}$. In the partially ordered set $(C, \leqslant)$, $C_i \leqslant C_j$ means that the users in $C_i$ have a security clearance lower than or equal to those in $C_j$. In other words, users in $C_j$ can derive the keys of users in $C_i$ and access information held by users in $C_i$, but the users in $C_i$ cannot access the information

* Corresponding author. Tel.: +886-4-23323000x4288; fax: +886-4-23742337.
*E-mail address:* mshwang@cyut.edu.tw (M.-S. Hwang).

0164-1212/03/$ - see front matter © 2002 Elsevier Inc. All rights reserved.
doi:10.1016/S0164-1212(02)00091-2

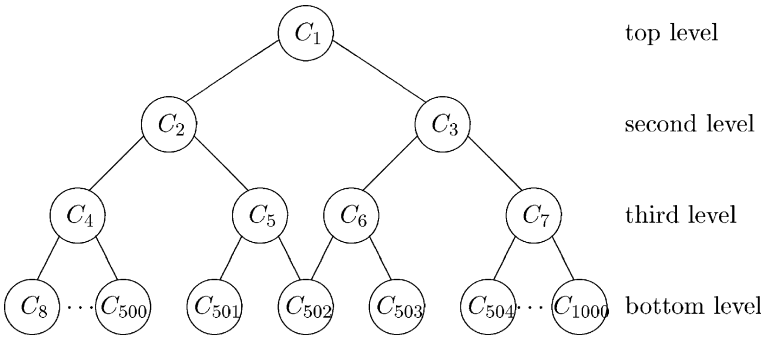


Fig. 1. An example of a hierarchical structure.

Table 1
The keys held by each user

| User | Keys held | Number of keys held |
|---|---|---|
| $C_1$ | $K_1, K_2, \ldots, K_{1000}$ | 1000 |
| $C_2$ | $K_2, K_4, K_5, K_8, \ldots, K_{502}$ | 498 |
| $C_3$ | $K_3, K_6, K_7, K_{502}, \ldots, K_{1000}$ | 502 |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $C_8$ | $K_8$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $C_{1000}$ | $K_{1000}$ | 1 |

held by the users in $C_j$. For the partially ordered set (poset, for short) structure, $C_i \leqslant C_j$, $C_i$ is called a descendant of $C_j$, and $C_j$ is called an ancestor of $C_i$. If there is no existing $C_k$ such that $C_i \leqslant C_k \leqslant C_j$, $C_i$ is called an immediate descendant of $C_j$, and $C_j$ is called an immediate ancestor of $C_i$. If there is no existing $C_i$ such that $C_i \leqslant C_j$, $C_j$ is called a leaf security class.

In the Akl–Taylor scheme (Akl and Taylor, 1983), each security class $C_i$ was assigned a distinct prime to be its public parameters, $PB_i$. The secret key, $K_i$, for each security class $C_i$ is calculated with the public parameters $PB_i$ by a central authority in the system. Information items owned by $C_i$ are encrypted by an available symmetric (one-key) cryptosystem (Schneier, 1996) with the enciphering key $K_i$. This information can be retrieved only by the security class $C_j$, where $C_i \leqslant C_j$. Using the public parameters, $PB_i$ and $PB_j$, and the secret key $K_j$, $C_j$ can derive $K_i$ to decipher information items owned by $C_i$. The Akl–Taylor scheme employs an elegant solution in a poset hierarchy for the access control problem. However, a large amount of storage for the public parameters is required. Moreover, new security classes cannot be added to or removed from the system once all the security keys have been issued. In 1985, Mackinnon et al. (1985) presented an improved algorithm for the Akl–Taylor scheme, called the canonical assignment, to reduce the value of public parameters. This scheme markedly reduces the number of distinct primes. However, a large amount of storage is still required for the $PB_i$. In addition, it is difficult to find an optimal canonical algorithm which can make an optimal assignment for an arbitrary poset hierarchy.

In 1988, Sandhu (1988) used one-way functions to create a cryptographic implementation of a tree hierarchy for access control (The tree hierarchy is a special case of a poset hierarchy). Each secret key $K_i$ for security class $C_i$ is generated with its own identity (ID) and its immediate ancestor's secret key through a one-way function. In the scheme, no extra public parameters are needed for key derivation. However, there are two drawbacks: one is that computational overhead is incurred in deriving keys. The other is that this proposal can only implemented for a tree hierarchy, which limits its applications. In 1990, Harn and Lin (1990) proposed an approach similar to the Akl–Taylor scheme. But, instead of using a top-down design approach as in the Akl–Taylor scheme, Harn and Lin presented a bottom-up key generating scheme. In the Harn–Lin scheme, the size of storage space needed to store the public parameters for most security classes is much smaller than that is needed for the MacKinnon et al. and Akl–Taylor schemes. However, in the Harn–Lin scheme when there are many security classes in the system, a large amount of storage space is required to store the public parameters.

In 1992 and 1993, both Chang et al. (1992) and Liaw et al. (1993) proposed other schemes based on Newton's interpolation method and a predefined one-way function. However, the computations needed for key generation and derivation in their schemes are time consuming. Furthermore, their schemes are insecure against cooperative attacks (Hwang et al., 1993; Hwang, 1999a; Hwang, 1999b). In 1993, Liaw and Lei proposed an optimal heuristic algorithm for assigning cryptographic keys in a tree structure (Liaw and Lei, 1993). The Liaw–Lei scheme not only reduces the amount of storage required for storing public parameters, but also is simple and efficient in generating and deriving keys. However, their algorithm can only be used in a tree structure (Hwang, 1997). In 1998, Yeh, Chow and Newman proposed a new cryptographic key assignment scheme (Yeh et al., 1998). The Yeh–Chow–Newman scheme enforces

access control policies in a user matrix model, which is more flexible than that in a user hierarchy. The user matrix model not only can model the access control policies in the user hierarchy model, but also more complicated policies with anti-symmetrical and transitive exceptions. However, their scheme is insecure (Hwang, 2000a). In 2000, Hwang proposed a new access control scheme for a totally ordered hierarchy that can be used in asymmetric cryptosystem (Hwang, 2000b).

We propose the following four criteria for evaluating the effectiveness and efficiency of a cryptography-based poset hierarchy scheme.

1. The effort needed to revise keys when a user is added or deleted.
2. The space needed to store public parameters.
3. The security of the system needed to achieve computational security and to withstand the cooperative attacks.
4. The effort needed to compute key generation and derivation.

The space needed to store public parameters and the computation of key generation and derivation are related to the number of primes. Thus, we shall also consider the number of primes to be used in the system.

The purpose of this paper is to propose a new method for generating small public parameters in a poset hierarchy for access control. In terms of the above criteria, our method has a good overall performance. This paper is organized as follows. In the next section, we present the new scheme. In Section 3, our method is compared with previous methods. Finally, Section 4 presents our conclusions.

## 2. An effective scheme

In this section, we present a new cryptography-based key assignment scheme for access control in a poset hierarchy. Our scheme can simultaneously reduce the public value and the number of primes.

### 2.1. Basic concept

The basic concept behind our scheme is to assign a set of primes to each security class so that the following three properties are satisfied.

1. Uniqueness: The public parameters of the set of primes of each security class are unique. Assume that the set of public parameters of $C_i(C_j)$ is $z_i(z_j)$, where $z_i(z_j)$ is a set of primes for $C_i(C_j)$. Then $z_i \neq z_j$ for any $i \neq j$.
2. Containment: The set of public parameters of descendant security classes is a subset of the public parameters of the ancestor security classes. That is, $z_i \subseteq z_j$ if $C_i \leqslant C_j$, for any $i$ and $j$. If there is no ancestor/descendant relationship between security classes $C_i$ and $C_j$, then these classes have no existing containment property. That is, $z_i \not\subseteq z_j$ if $C_i \not\leqslant C_j$. For example, there is no ancestor/descendant relationship between $C_4$ and $C_5$ in Fig. 1. Therefore, $z_4 \not\subseteq z_5$ or $z_5 \not\subseteq z_4$.
3. Minimal size: The number of elements in the set of public parameters is the smallest number such that the above properties hold.

To satisfy the above properties, we propose a simple approach that assigns a set of public parameters to each security class. Our scheme is a bottom-up key generating scheme. To implement our concept, the central authority (CA) first chooses a prime in each security class as a public parameter and products these primes (which we refer to as composed primes) to public parameters of leaf classes in the poset hierarchy. The public parameters of all classes except leaf security classes are assigned by taking the union of these composed primes from the immediate descendant, from those of the bottom through the root of the poset hierarchy level by level. We shall discuss the above three properties in detail in Section 2.4.

### 2.2. Key generation procedures

We also assume that there is a CA in the system similar to that in Akl–Taylor's and Harn–Lin's scheme. The CA then executes the following steps:

Step 1. CA chooses two large prime numbers $p$ and $q$ and computes the parameter $m = p \cdot q$, where $p$ and $q$ are kept secret and $m$ is public.

Step 2. CA chooses another parameter, for example a random number $K_0$, between 2 and $(m-1)$ such that $K_0$ and $m$ are relatively prime.

Step 3. CA selects a set of primes, $e_i$, and calculates the multiplicative inverse, $d_i$, for each prime. That is $d_i = e_i^{-1} \bmod \phi(m)$. The number of these primes will be discussed in Section 2.4.

Step 4. CA assigns each leaf security class in the hierarchy a distinct composed prime set, $z_i$, which is not a subset of $z_l$ if $C_i \not\leqslant C_l$. $C_i$ and $C_l$ are leaf security classes. This step satisfies the three properties in Section 2.1. We shall also discuss it in Section 2.4.

Step 5. CA assigns each non-leaf security classes $C_j$ a distinct prime $e_j$. Next, CA generates a distinct composed prime set $x_j$ by taking the union of these composed prime sets $z_i$ of the immediate descendant class and $e_j$. For any two classes $C_i$ and $C_j$, if $C_i$ is not an ancestor of $C_j$, $x_j$ is not a subset of $x_i$. This property will resist the conspiracy attack.

Step 6. CA calculates a set of public parameters $PB_i$ and secret keys $K_i$ for all leaf security classes $C_i$ in the hierarchy such that $PB_i = \prod e_l$ and $K_i = K_0^{f(C_i)\prod d_l \bmod \phi(m)} \bmod m$, for all $l$ such that $e_l \in z_i$, where $f(\cdot)$ denotes a one-way function. If $PB_i K_i \bmod \phi(m) = 1$, discard $(PB_i, K_i)$ and repeat Step 3. Otherwise, the second type of common modulus attack will arise. We shall discuss it in Section 2.8.

Step 7. CA calculates a set of public parameters $PB_j$ and secret keys $K_j$ for all non-leaf security classes in the hierarchy such that $PB_j = \prod e_h$ and $K_j = K_0^{\prod d_h \bmod \phi(m)} \bmod m$, for all $h$ such that $e_h \in x_j$.

### 2.3. Key derivation procedures

Assume that there are $C_i$ and $C_j$ in the poset hierarchy such that $C_i \leqslant C_j$, and keys $K_i$ and $K_j$ which are generated by the key generation procedures for $C_i$ and $C_j$, respectively. Thus the user $u_j$, where $u_j \in C_j$, can derive the key $K_i$ of $C_i$ by the following formula:

$$K_i = \begin{cases} K_j^{(PB_j/PB_i)f(C_i)} \bmod m & \text{if } C_i \text{ is a leaf class,} \\ K_j^{PB_j/PB_i} \bmod m & \text{if } C_i \text{ is not a leaf class.} \end{cases} \tag{1}$$

Here, $C_j$ need only use his/her secret key $K_j$ together with both public parameters of $C_i$ and $C_j$ to derive the secret key of $C_i$. For this, $C_j$ does not need to know $\phi(m)$.

Suppose $C_j$ wants to derive the secret key $K_i$ of a descendant $C_i$ which is a leaf class. In Step 6 of the Key Generation Procedures in Section 2.2, we have seen that $PB_i = \prod e_l$, $PB_j = \prod e_h$, $K_i = K_0^{f(C_i)\prod d_l \bmod \phi(m)} \bmod m$, and $K_j = K_0^{\prod d_h \bmod \phi(m)} \bmod m$, for all $l$ and $h$ such that $e_l \in z_i$ and $e_h \in x_j$. Now, we substitute $PB_i$ and $PB_j$ to Eq. (1) and show that the key derivation in Eq. (1) is correct, as follows.

$$\begin{aligned} & K_j^{(PB_j/PB_i)f(C_i)} \bmod m \\ &= \left(K_0^{\prod d_h \bmod \phi(m)}\right)^{\left(\prod e_h / \prod e_l\right) f(C_i)} \bmod m, \qquad (2) \\ &\quad \text{for all } l \text{ and } h \text{ such that } e_l \in z_i \text{ and } e_h \in x_j, \\ &= K_0^{f(C_i)\left(1/\prod e_l\right)} \bmod m, \qquad (3) \\ &\quad \text{for all } l \text{ such that } e_l \in z_i, \\ &= \left(K_0^{f(C_i)\prod d_l \bmod \phi(m)}\right) \bmod m, \qquad (4) \\ &= K_i. \end{aligned}$$

Since $(K_0^{(\prod d_h \bmod \phi(m))\cdot(\prod e_h)} \bmod m)$ in Eq. (2) is equal to $K_0$, we thus get Eq. (3). Since $ed \bmod \phi(m) = 1$, $K_0^{d \bmod \phi(m)} \bmod m$ is equal to $K_0^{1/e} \bmod m$. Therefore, we get Eq. (4) from Eq. (3).

### 2.4. The key assignment method

Two problems remain from Section 2.1. One is how many primes are used in the system, and the other is how many primes are in $z_i$ for each $C_i$. We will give solutions to both these problems in this subsection.

We give a poset hierarchy with $n$ security classes. The term "leaf-group" denotes a security class whose immediate descendants are the leaf security classes. For example, there are four leaf-groups in Fig. 1: $C_4$, $C_5$, $C_6$ and $C_7$. Let $LG_i$ be the $i$th leaf-group; the number of members of $LG_i$ be $m_{il}$. Thus, the CA can select the number of primes using the following formula:

$$y = \left(\sum_{\text{for all } LG_i} g_i\right) + n_a + n_t, \tag{5}$$

where:

- $g_i$ is the number such that $\binom{g_i}{k} \geqslant m_{il}$, where $\binom{g_i}{k}$ indicates $g_i$ chooses $k$. Here, $k$ is the number of primes that distinguish the leaf security classes for each leaf-group,
- $n_a$ denotes the number of leaf security classes that have two or more immediate ancestors, and
- $n_t$ denotes the number of non-leaf security classes.

To minimize the space required for public parameters, we need to minimize the number $g_i$. For example, assuming $m_{il} = 400$, then $g_i = 11$ is the minimal number for the hierarchy, and $k$ can be taken to be 5. Now, we will show by means of the following theorem that Eq. (5) satisfies the three properties in Section 2.1.

**Theorem 2.1.** *The number of primes, $y$, in Eq. (5) satisfies the three properties described in Section 2.1.*

**Proof.** When the hierarchy is a totally ordered with $t$ ordered levels, $t$ distinct primes are needed to satisfy these properties described in Section 2.1. There are no primes needed to be assigned for distinguishing leaf security classes in the leaf-group. One distinct prime for each leaf security class is needed to satisfy the uniqueness property. Thus $t$ primes are needed. Obviously, $y$ satisfies the three properties in totally ordered hierarchy.

When the hierarchy is a partially ordered with $r$ leaf-groups, find the set of $(g_i, k)$, $i = 1, 2, \ldots, r$, which satisfies $\binom{g_i}{k} \geqslant m_{il}$, so that $g_i$ distinct primes make the composite primes which identify members of each leaf-group. Therefore, the uniqueness property is satisfied. By the bottom-to-top key assignment method, the containment property is satisfied. Finding the minimal number of $g_i$ such that $\binom{g_i}{k} \geqslant m_{il}$ satisfies the minimal property. □

### 2.5. An example

For the poset hierarchy with one thousand security classes $C_i$, $(i = 1, 2, \ldots, 1000)$, shown in Fig. 1, Table 2 shows the keys assigned to each security class using our method.

Now, assume that user 4, $u_4$, belonging to security class $C_4$, wishes to access the information of user 8, $u_8$, where $u_8 \in C_8$. Since $C_8 \leqslant C_4$, $u_4$ can use his/her own secret key $K_4$ to derive $K_8$, the key of $u_8$, as follows:

$$\begin{aligned} & K_4^{f(C_8)PB_4/PB_8} \bmod m \\ &= (K_0^{d_4 d_8 d_9 \cdots d_{23} \bmod \phi(m)})^{f(C_8)e_4 e_8 e_9 \cdots e_{23} e_8 e_9 e_{10}} \bmod m \\ &= (K_0^{d_4 d_8 d_9 \cdots d_{23} \bmod \phi(m)})^{f(C_8)e_4 e_{11} e_{12} \cdots e_{23}} \bmod m \\ &= K_0^{f(C_8) d_8 d_9 d_{10} \bmod \phi(m)} \bmod m \\ &= K_8. \end{aligned}$$

Only 42 primes are required for the above hierarchy. However, Harn–Lin's scheme and Akl–Taylor's scheme would require 1000 primes for the above hierarchy.

### 2.6. Dynamic ability

When adding a new security class to a system, we assign a distinct public parameter to the class. If the new class belongs to an existing leaf-group, there are two cases must be considered. First, assume that the leaf-group is not the group with the maximal number of leaf security classes in the system such that more distinct composed primes can be assigned to this new class. In this case, CA only assigns a distinct composed prime to the new class, but does nothing for all security classes. Second, assume that the leaf-group is the group with the maximal number of leaf security classes in the system such that more distinct composed primes can be assigned to this group. For example, if the system has 105 members, the maximal number of leaf-groups, $(g, k) = (15, 2)$ to distinguish these leaf security classes. In this case, when adding a new security class to the system, CA assigns a distinct composed prime by adding a prime to this new class. It then changes the public parameters by multiplying the new prime and corresponding secret keys for the ancestor node of that new class. The number of security classes which need to be updated by the public parameters and secret keys is $l$, where $l$ is the level of the hierarchy.

If the new security class does not belong to an existing leaf-group, there are two cases must be considered. First, assume that the new class is a leaf. CA assigns a new composite prime to the new class and calculates public parameters and secret keys for the new class and all ancestor classes of the new class. Second, assume that the new class is not a leaf. CA assigns a new prime to the new class and calculates public parameters and secret keys for the new class and all ancestor classes of the new class.

When deleting an existing security class from the system, the CA just drops the entity of the security class. CA does nothing for all security classes.

Table 2
The public parameters and secret keys for security classes using our method in Fig. 1

| Security classes $C_i$ | Public parameters ($PB_i$) | Secret keys ($K_i$) |
|---|---|---|
| $C_1$ | $e_1 e_2 \cdots e_{42}$ | $K_0^{d_1 d_2 \cdots d_{42} \bmod \phi(m)} \bmod m$ |
| $C_2$ | $e_2 e_4 e_5 e_8 \cdots e_{25}$ | $K_0^{d_2 d_4 d_5 d_8 \cdots d_{25} \bmod \phi(m)} \bmod m$ |
| $C_3$ | $e_3 e_6 e_7 e_{25} \cdots e_{42}$ | $K_0^{d_3 d_6 d_7 d_{25} \cdots d_{42} \bmod \phi(m)} \bmod m$ |
| $C_4$ | $e_4 e_8 \cdots e_{23}$ | $K_0^{d_4 d_8 \cdots d_{23} \bmod \phi(m)} \bmod m$ |
| $C_5$ | $e_5 e_{24} e_{25}$ | $K_0^{d_5 d_{24} d_{25} \bmod \phi(m)} \bmod m$ |
| $C_6$ | $e_6 e_{25} e_{26}$ | $K_0^{d_6 d_{25} d_{26} \bmod \phi(m)} \bmod m$ |
| $C_7$ | $e_7 e_{27} \cdots e_{42}$ | $K_0^{d_7 d_{27} \cdots d_{42} \bmod \phi(m)} \bmod m$ |
| $C_8$ | $e_8 e_9 e_{10}$ | $K_0^{f(C_8) d_8 d_9 d_{10} \bmod \phi(m)} \bmod m$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $C_{500}$ | $e_{21} e_{22} e_{23}$ | $K_0^{f(C_{500}) d_{21} d_{22} d_{23} \bmod \phi(m)} \bmod m$ |
| $C_{501}$ | $e_{24}$ | $K_0^{f(C_{501}) d_{24} \bmod \phi(m)} \bmod m$ |
| $C_{502}$ | $e_{25}$ | $K_0^{f(C_{502}) d_{25} \bmod \phi(m)} \bmod m$ |
| $C_{503}$ | $e_{26}$ | $K_0^{f(C_{503}) d_{26} \bmod \phi(m)} \bmod m$ |
| $C_{504}$ | $e_{27} e_{28} e_{29}$ | $K_0^{f(C_{504}) d_{27} d_{28} d_{29} \bmod \phi(m)} \bmod m$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $C_{1000}$ | $e_{40} e_{41} e_{42}$ | $K_0^{f(C_{1000}) d_{40} d_{41} d_{42} \bmod \phi(m)} \bmod m$ |

### 2.7. Required storage and computational complexity

From the key assignment method and key generating procedure, the maximal number of primes to be factorial-multiplied for public parameters is equal to $y$. Assume that there are $n$ security classes in the hierarchy, then the amount of space needed to store public parameters requires $O(yn)$ primes.

It is easy to see that the computational complexity of our scheme depends on the number of primes to be multiplied. Thus, our scheme takes $O(y)$ to generate keys for each security class. For the key derivation procedure, only one division, one exponent and one module are required.

### 2.8. Security analysis

The security in our scheme is based on the difficulty of factoring a product of two large primes. As long as all the system privacy keys $\{d_i\}$, prime numbers $p$ and $q$, are kept secret, then breaking our scheme would be as difficult as breaking the RSA scheme (Chang and Hwang, 1996; Hwang et al., 2000; Changchien and Hwang, in press; Rivest et al., 1978). Thus any user outside of the system cannot discover the key of a user in the system. Any user $u_i$ in the system cannot discover the system privacy keys $d_i$, so that user $u_i$ cannot use his/her own key to calculate other secret keys. As to cooperative attacks, a conspiracy between child keys cannot discover their ancestors' keys. Because of the secret keys of the

class $C_i$ cannot be discovered by the conspiracy of others $\{K_j | C_j \not\geqslant C_i\}$.

In addition, in order to show that our scheme is secure against common modulus attacks, we briefly introduce two types of common modulus attack (DeLaurentis, 1984; Schneier, 1996). The first type of common modulus attack uses the same modulus $m$ and two different public keys $e_1$ and $e_2$ to encrypt the same plaintext $M$ to give $C_1 = M^{e_1} \bmod m$ and $C_2 = M^{e_2} \bmod m$. If $e_1$ and $e_2$ are relatively prime, then there exists $s$ and $t$ such that $se_1 + te_2 = 1$ holds (Diffie and Hellman, 1976). In this case, the plaintext $M$ can been obtained by the following computation (Schneier, 1996):

$$\begin{aligned} & C_1^s (C_2^t) \bmod m \\ &= (M^{e_1})^s (M^{e_2})^t \bmod m \\ &= (M^{(e_1 s + e_2 t)}) \bmod m \\ &= M. \end{aligned}$$

The second type of common modulus is that a subscriber can use his/her own public $e_2$ and private keys $d_2$ to generate the private key $d_1$ of another user (DeLaurentis, 1984). First, the subscriber finds the greatest common divisor $a$ of $e_1$ and $e_2 d_2 - 1$ by the Euclidean algorithm. Next, they let $b = (e_2 d_2 - 1)/a$. Since $a$ is a divisor of $e_1$, and $e_1$ is relatively prime to $\phi(m)$, $a$ must also be relatively prime to $\phi(m)$. However, $ab = e_2 d_2 - 1$ is a multiple of $\phi(m)$. In other words, $b$ must be a multiple of $\phi(m)$. Since $b$ is relatively prime to $e_1$, there exist $s$ and $t$, such that $sb + te_1 = 1$ holds. Because $b$ is a multiple of $\phi(m)$, therefore $te_1 \bmod \phi(m) = 1$. Here, $t$ is equal to $d_1$. Therefore, the subscriber can obtain the private key $d_1$ of another user.

Since each security class use his/her own secret key ($K_i$) to encrypt plaintext but not use public parameter $e_i$, our scheme is secure against the first type of common modulus attacks. The public parameters are only used to derive the secret key of immediate descendant class.

In addition, each security class only knows his/her secret key $K_i$ although he/she does not known the private key $d_i$ in our scheme. The private key $d_i$ is only known by CA. Therefore, a subscriber cannot use public $e_2$ and private keys $d_2$ to generate the private key $d_1$ of another security class. In addition, $e_i K_i \bmod \phi(m)$ is not equal to 1 in our scheme. Our scheme can withstand the second type of common modulus attacks.

### 2.9. Discussion

In Section 2.4 there are many sets of $(g, k)$'s to satisfy $\binom{g_i}{k} \geqslant m_{il}$. Although we can find the minimal $g_i$, the storage space for derived keys may not be minimized. For example, we assume that the maximal numbers of members of a leaf-group is 400, the set of $(g_i, k)$ : $\{(11,5)(12,4)(15,3)(29,2)(400,1)\}$, satisfies $\binom{g_i}{k} \geqslant m_{il}$. We can pick up any subset, i.e. $(g_i, k) = (11, 5)$, to model the structure. When $(g_i, k) = (11, 5)$, the number of primes involved is minimal, but we must select $k = 5$ distinct primes as a composed prime set. This requires more storage space. Although $(g_i, k) = (15, 3)$ is not minimal, but $k = 3$ requires less storage space, based on the fact that the hierarchy structure is a pyramid. However, the problem of determining the minimal storage space in a poset hierarchy is still an unresolved issue.

## 3. Comparisons

In this section, we compare our method with other cryptography-based hierarchy schemes in the literature, using the four criteria in Section 1. Previous methods are briefly reviewed below.

Akl and Taylor (1983) assigned to each security class $C_i$ an associated distinct prime $e_i$ and calculated the public parameter $PB_i$ as follows:

$$PB_i = \prod_{C_j \not\leqslant C_i} e_j. \tag{6}$$

Thus, the secret keys for all security classes could be computed as follows:

$$K_j = K_0^{PB_j} \bmod m, \tag{7}$$

where $m$ is the product of $p$ and $q$, which are two random large primes, and $K_0$ is a random secret key, where $2 \leqslant K_0 \leqslant m - 1$, $\gcd(K_0, m) = 1$. A descendant's key can be derived from an ancestor's key by the formula

$$K_j = K_i^{PB_j / PB_i} \bmod m, \quad \text{iff } C_j \leqslant C_i. \tag{8}$$

The scheme of Mackinnon et al. (1985) is the same as the original Akl–Taylor scheme, except that the former uses chain decomposition to determine the primes to be used. Each public parameter $PB_i$ of a descendant in the chain is the power exponential of $PB_j$ owned by the ancestor node.

Harn and Lin (1990) presented a cryptography-based hierarchy scheme, based on the difficulty in factoring a product of two large primes. Instead of using a top-down design approach, as in the Akl–Taylor scheme, the Harn–Lin scheme used a bottom-up key generation procedure.

An example is given to illustrate the above methods.

**Example 3.1.** Given the poset structure with one thousand security classes shown in Fig. 1, we find the prime to be used, using the above methods as shown in Tables 3 and 4.

Each method introduced above has its own way of dealing with the four criteria we use to evaluate a poset hierarchy, as summarized in Tables 5–7. Table 5 shows

Table 3
Public parameters for each security class using Akl–Taylor method in Fig. 1

| Security classes | Public parameters ($PB_i$) | Number of primes |
|---|---|---|
| $C_1$ | 1 | 1 |
| $C_2$ | $e_1e_3e_6e_7e_{502}\cdots e_{1000}$ | 503 |
| $C_3$ | $e_1e_2e_4e_5e_8\cdots e_{502}$ | 499 |
| $C_4$ | $e_1e_2e_3e_5e_6e_7e_{501}\cdots e_{1000}$ | 506 |
| ⋮ | ⋮ | ⋮ |
| $C_7$ | $e_1e_2e_3e_4e_5e_6e_8\cdots e_{503}$ | 502 |
| $C_8$ | $e_1\cdots e_7e_9\cdots e_{1000}$ | 999 |
| ⋮ | ⋮ | ⋮ |
| $C_{1000}$ | $e_1\cdots e_{999}$ | 999 |

Table 4
Public parameters for each security class using Harn–Lin method in Fig. 1

| Security classes | Public parameters ($PB_i$) | Number of primes |
|---|---|---|
| $C_1$ | $e_1e_2\cdots e_{1000}$ | 1000 |
| $C_2$ | $e_2e_4e_5e_8\cdots e_{502}$ | 498 |
| $C_3$ | $e_3e_6e_7e_{502}\cdots e_{1000}$ | 502 |
| $C_4$ | $e_4e_8\cdots e_{500}$ | 494 |
| ⋮ | ⋮ | ⋮ |
| $C_7$ | $e_7e_{504}\cdots e_{1000}$ | 498 |
| $C_8$ | $e_8$ | 1 |
| ⋮ | ⋮ | ⋮ |
| $C_{1000}$ | $e_{1000}$ | 1 |

the structure and access type used. The structure of the tree hierarchy is a special case of the poset hierarchy. The tree hierarchy is thus less flexible in application. Direct access means that any security class can directly access any lower security class. Indirect access means that the security class can only access any immediate descendant class. When an individual wants to access other security classes, he/she must derive the secret key of each immediate descendant class level by level. Obviously, the complexity of the indirect access takes considerable time. In Table 5, we compare our scheme with other schemes that have the same structure and access type, i.e. same poset hierarchy structure and same direct access.

Table 6 shows four aspects of the computational complexity: append ability, remove ability, key generation, and key derivation. The computation of append ability is required to analyze the number of security classes needed to update keys when a new security class is added to partially ordered hierarchies. The computation of remove ability is required to analyze the number of security classes need to update keys when removing an existed security class from a partially ordered hierarchy. For append ability, the computational complexity is $O(l)$ which must be re-computed in Harn–Lin's scheme where $l$ is the levels of hierarchy. For remove ability, the computational complexity is $O(1)$ for both Harn–Lin and our proposed scheme. For key generation, the computational complexity is only $O(y)$ for our proposed scheme, where $O(y) \ll O(n)$. In terms of key derivation column, the computational complexity of all schemes is $O(1)$.

Table 5
Implementation

| Schemes | Structure | Access type |
|---|---|---|
| Akl and Taylor | Poset hierarchy | Direct |
| MacKinnon et al. | Poset hierarchy | Direct |
| Sandhu | Tree hierarchy | Indirect |
| Harn and Lin | Poset hierarchy | Direct |
| Chang et al. | Poset hierarchy | Indirect |
| Proposed method | Poset hierarchy | Direct |

Table 6
The computational complexity

| Schemes | Append ability | Remove ability | Key generation | Key derivation |
|---|---|---|---|---|
| Akl and Taylor | $O(n)$ | $O(n)$ | $O(n)$ | $O(1)$ |
| MacKinnon et al. | $O(n)$ | $O(n)$ | $O(n)$ | $O(1)$ |
| Harn and Lin | $O(l)$ | $O(1)$ | $O(n)$ | $O(1)$ |
| Proposed method | $O(1)$ | $O(1)$ | $O(y)$ | $O(1)$ |

Table 7
The space complexity

| Schemes | Number of primes | Maximum public parameter | Storage spaces |
|---|---|---|---|
| Akl and Taylor | $n$ | $\prod_{i=1}^{n} p_i$ | $O(n^3 \log n)$ |
| MacKinnon et al. | $c$ | $\prod_{i=1}^{n} p_i$ | $O(n^3 \log c)$ |
| Harn and Lin | $n$ | $\prod_{i=1}^{n} p_i$ | $O(n^3 \log n)$ |
| Proposed method | $y$ | $\prod_{i=1}^{y} p_i$ | $O(n^3 \log y)$ |

Table 7 shows three aspects of storage space: number of primes, maximum public parameter, and storage spaces. Both Akl–Taylor's and Harn–Lin's scheme need $n$ primes, the same as number of security classes in hierarchy. MacKinnon et al.'s scheme needs $c$ primes, the same as number of chains. However, our proposed scheme needs only $y$, as shown in Eq. (5), where $y \leqslant c \ll n$. We prove $y \leqslant c$ by the following theorem.

**Theorem 3.1.** *Let y have the same definition as in the Eq.* (5), *and let c be the number of chains in a totally ordered hierarchy* (*that is, for any two security classes* $C_i$ *and* $C_j$ *in the chain, where* $C_i \neq C_j$, *either* $C_i \leqslant C_j$ *or* $C_j \leqslant C_i$,). *Then* $y \leqslant c$.

**Proof.** First, since the chain is in a totally ordered hierarchy and there are no relationships among leaf security classes in any poset hierarchy, the number of chains is larger than the number of leaf security classes. Next, $g_i$ is always less than $m_{il}$ to satisfy $\binom{g_i}{k} \geqslant m_{il}$. Here, $m_{il}$ is the number of leaves in the leaf-group $LG_i$. From the above two points, the theorem holds. □

In the maximum public parameter column, there are only $y$ primes to be multiplied in our scheme. In the storage spaces column, our scheme requires $O(n^3 \log y)$ bits. However, Akl–Taylor's and Harn–Lin's schemes require $O(n^3 \log n)$ bits; while MacKinnon et al.'s scheme requires $O(n^3 \log c)$ bits, where $O(n^3 \log n) > O(n^3 \log c) > O(n^3 \log y)$.

## 4. Conclusions

In terms of the four criteria introduced earlier, our key assignment for the poset hierarchy scheme is a considerably better approach to access control than the other comparable schemes discussed above. The primary merit of our scheme is its simplicity, in terms of both the underlying idea and the algorithm for assigning public parameters. The advantages of our scheme in application include the following:

1. There are relatively few primes in the system, so we avoid using large prime numbers as public parameters and reduce the size of public parameter values.
2. The security is equivalent to that provided by the RSA cryptosystem.
3. When a security class wishes to access a permitted resource owned by other security classes, our scheme can easily derive the secret key of the security class to the other resource.
4. When a new security class is added, only the key values of the new security class need to be computed.
5. To remove a security class from the system, the scheme removes the key values of that security class only.

Since the fact that the hierarchy structure is a pyramid, our scheme is efficient than other schemes. Our proposed scheme can also be used to combine several poset hierarchy systems into a larger poset hierarchy system, called a group poset hierarchy system. The keys of the existing security classes need only to be multiplied by the group identification number. However, other methods in the literature require that the keys of the existing security classes in the poset hierarchy be recomputed using a large number as the maximal number of the lower poset hierarchy.

## Acknowledgements

The authors wish to thank many anonymous referees for their suggestions to improve this paper. Part of this research was supported by the National Science Council, Taiwan, ROC, under contract no. NSC90-2213-E-324-005.

## References

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

Bell, D., LaPadula, L., 1975. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, Hanscom Air Force Base Technical Report, Beford, MA.

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.

Chang, C.-C., Hwang, M.-S., 1996. Parallel computation of the generating keys for RSA cryptosystems. IEE Electronics Letters 32 (15), 1365–1366.

Changchien, S.W., Hwang, M.-S., in press. A batch verifying and detecting multiple RSA digital signatures. International Journal of Computational and Numerical Analysis and Applications.

DeLaurentis, J.M., 1984. A further weakness in the common modulus protocol for the RSA cryptosystem. Cryptologia 8 (3), 253–259.

Denning, D.E., 1984. Cryptographic checksums for multilevel database security. In: Proceedings of the 1984 IEEE Symposium on Security and Privacy, Oakland, pp. 52–61.

Denning, D.E., Akl, S.G., Morgenstern, M., Neumann, P.G., Schell, R.R., Heckman, M., 1986. Views for multilevel database security. In: Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, pp. 156–172.

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

Fraim, L.J., 1983. Scomp: A solution to the multilevel security problem. IEEE Computer, 26–34.

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

Hwang, M.-S., 1997. A cryptographic key assignment scheme in a hierarchy for access control. Mathematical and Computer Modelling 26 (2), 27–31.

Hwang, M.-S., 1999a. Extension of CHW cryptographic key assignment scheme in a hierarchy. IEE Proceedings Computers and Digital Techniques 146 (4), 219.

Hwang, M.-S., 1999b. An improvement of a dynamic cryptographic key assignment schemes in a tree hierarchy. Computers and Mathematics with Applications 37 (3), 19–22.

Hwang, M.-S., 2000a. An asymmetric cryptographic scheme for a totally-ordered hierarchy. International Journal of Computer Mathematics 73, 463–468.

Hwang, M.-S., 2000b. Cryptanalysis of YCN key assignment scheme in a hierarchy. Information Processing Letters 73 (3), 97–101.

Hwang, M.-S., Chang, C.-C., Yang, W.-P., 1993. Modified Chang–Hwang–Wu access control scheme. IEE Electronics Letters 29 (24), 2095–2096.

Hwang, M.-S., Lin, I.-C., Hwang, K.-F., 2000. Cryptanalysis of the batch verifying multiple RSA digital signatures. Informatica 11 (1), 15–19.

Liaw, H.-T., Lei, C.-L., 1993. An optimal algorithm to assign cryptographic keys in a tree structure for access control. BIT 33, 46–56.

Liaw, H.T., Wang, S.J., Lei, C.L., 1993. A dynamic cryptographic key assignment scheme in a tree structure. Computers and Mathematics with Applications 25 (6), 109–114.

Lu, W.P., Sundareshan, M.K., 1992. Enhanced protocols for hierarchical encryption key management for secure communication in internet environments. IEEE Transactions on Communications 40 (4), 658–660.

Lu, W.P., Sundareshan, M.K., 1988. A model for multilevel security in computer networks. In: Proceedings of the 1988 INFOCOM, New Orleans, LA, , pp. 1095–1104.

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 34 (9), 797–802.

McCullough, D., 1987. Specifications for multi-level security and a hook-up property. In: Proceedings of the 1987 IEEE Symposium on Security and Privacy, pp. 161–166.

McHugh, J., Moore, A.P., 1986. A security policy and formal top level specification for a multi-level secure local area network. In: Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, pp. 34–39.

Mcilroy, M.D., Reeds, J.A., 1992. Multilevel security in the UNIX tradition. Software—Practice and Experience 22 (8), 673–694.

Rivest, R.L., Shamir, A., Adleman, L., 1978. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM 21 (2), 120–126.

Sandhu, R.S., 1988. Cryptographic implementation of a tree hierarchy for access control. Information Processing Letters 27, 95–98.

Schneier, B., 1996. Applied Cryptography, second ed., John Wiley & Sons, New York.

Yeh, J.H., Chow, R., Newman, R., 1998. A key assignment for enforcing access control policy exceptions. In: Proceedings on International Symposium on Internet Technology, Taipei, pp. 54–59.

**Min-Shiang Hwang** received the B.S. in Electronic Engineering from National Taipei Institute of Technology, Taipei, Taiwan, Republic of China (ROC), in 1980; the M.S. in Industrial Engineering from National Tsing Hua University, Taiwan, in 1988; the Ph.D. in Computer and Information Science from National Chiao Tung University, Taiwan, in 1995. He also studied Applied Mathematics at National Cheng Kung University, Taiwan, from 1984 to 1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in field "Information Engineering", qualified as advanced technician the first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, ROC. He was also a project leader for research in computer security at TL in July 1990. He obtained the 1997, 1998, 1999, 2000, 2001 Distinguished Research Awards of the National Science Council of the Republic of China. He is currently a professor and chairman of the Department of Information Management, Chaoyang University of Technology, Taiwan, ROC. He is a member of IEEE, ACM, IEICE, and Chinese Information Security Association. His current research interests include database and data security, cryptography, image compression, and mobile communications.

**Wei-Pang Yang** was born on May 17, 1950 in Hualien, Taiwan, Republic of China. He received the B.S. degree in mathematics from National Taiwan Normal University in 1974, and the M.S. and Ph.D. degrees from the National Chiao Tung University in 1979 and 1984, respectively, both in computer engineering. Since August 1979, he has been on the faculty of the Department of Computer Engineering at National Chiao Tung University, Hsinchu, Taiwan. In the academic year 1985–1986, he was awarded the National Postdoctoral Research Fellowship and was a visiting scholar at Harvard University. From 1986 to 1987, he was the Director of the Computer Center of National Chiao Tung University. In August 1988, he joined the Department of Computer and Information Science at National Chiao Tung University, and acted as the Head of the Department for one year. Then he went to IBM Almaden Research Center in San Jose, California for another one year as visiting scientist. From 1990 to 1992, he was the Head of the Department of Computer and Information Science again. His research interests include database theory, database security, object-oriented database, image database and Chinese database systems. Dr. Yang is a full professor and a member of IEEE, ACM, and the phi Tau Phi Society. He was the winner of the 1988 and 1992 Acer Long Term Award for Outstanding M.S. Thesis Supervision, and the winner of 1990 Outstanding Paper Award of the Computer Society of the Republic of China. He also obtained the 1991–1993 Outstanding Research Award of National Science Council of the Republic of China.