Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp1065-1069)

Efficient and Scalable Hierarchical Key Assignment Scheme

Chin-Chen Chang[12], Chia-Lin Kao[2], and Chia-Chen Lin[3*]
[1]Department of Information Engineering and Computer Science,
Feng Chia University, Taichung, 407, Taiwan, R.O.C.

[2]Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi, 621, Taiwan, R.O.C.
[3]Department of Computer Science and Information Management,
Providence University, Taichung, 433, Taiwan, R.O.C.

Abstract: - The access privileges in distributed systems are often organized as a hierarchy structure. Chien and Jan proposed an efficient hierarchical key assignment scheme without using the public key cryptosystem in 2003. Nevertheless, their scheme must use the smart card and the information published on the public board contains the essential secrets of the server and the corresponding users. We therefore propose a novel hierarchical key assignment scheme which can preserve the advantages of Chien and Jan's scheme without adopting the smart card. Moreover, even legal entities can not use the public information announced on the authenticated board to derive the secret keys of other members.

Key-Words: Hierarchical Key Assignment, Time-bound Property, Scalability, Efficiency

1 Introduction

A hierarchy tree structure is always used to represent members' access rights in the government, the army, or the company. People at higher levels can learn the secret data possessed by those who are their subordinates in the hierarchy. This mechanism not only supports users to protect their secret from being stolen or tampered by unrelated participants but also makes managers easy to access secret data of their subordinates in an organization.

Since Akl and Taylor applied the hierarchy concept to present an access control scheme in 1983 [1], researchers have proposed many improvements [4, 5, 6, 7, 8]. However, most proposed schemes are not computationally efficient because of using public-key infrastructure (PKI), which results in high computational loads and implementation costs.

In 2001, Lin [6] proposed an efficient hierarchical key assignment scheme without using public-key cryptography. Although Lin's scheme is efficient, each system member requires a tamper-resistant hardware. Later, Chien and Jan [2] presented another new hierarchical key assignment scheme. But, a low-cost tamper-resistant device is still required to store each user's secret key and to perform simple arithmetic operations.

Tzeng [9] proposed a time-bound key assignment scheme for access control in 2002. His time-bound property allows the system manger to control the valid time of each member's secret key. In 2004, Chien [3] soon proposed another time-bound version for hierarchical key assignment. Chien's scheme is efficient and scalable, but it still needs a smart card to store each user's secret key and two parameters referred to time-bound issue. We thus propose a new hierarchical key assignment scheme. In the scheme, each user only needs to keep a password as his secret key. Since the password is chosen by a user himself, he can easily remember it without a smart card. Without adopting the smart card to store users' related information, our scheme still has time-bound property, and preserves the advantages of the previous works.

The rest of this paper is organized as follows. Section 2 gives a brief review of the related works. Section 3 describes our hierarchical key assignment scheme. In Section 4, two extensions of our proposed scheme are presented. Section 5 examines the security and evaluates the performance of the proposed scheme. Finally, some conclusions are given in Section 6.

2 Related Works

In 2003, Chien and Jan [2] proposed a hierarchical assignment without using public keys in their approach. Suppose a hierarchical tree which consists of n disjoint classes, $S = \{C_1, C_2, ..., C_n\}$, where each $C_i, 1 \le i \le n$, corresponds to one node in the tree. An edge "Ci Cj" means that the entities belonging to C_i are entitled to derive the key of C_j. In the initialization phase, Trusted Agent (TA), first of all,

generates n secret keys K_i and distributes those K_i's to the entities belonging to the node C_i through a secure channel, where $1 \le i \le n$. Then, TA issues each entity of node C_i a tamper-resistance device which contains TA's secret key X and the identity ID_i of C_i. For each directed edge "C_i π C_j," TA publishes a public value $r_{ij}=h(X||ID_i||ID_j||K_i)$ K_j on an authenticated public board, where $h()$ is a secure one-way hash function, || denotes the string concatenation and denotes the bit-wise XOR operation.

Suppose an entity belonging to C_i wants to derive the secret K_i of C_i. He just inputs the public values r_{ij}'s, ID_j, and his secret key K_j. The tamper-resistant device then derives the secret K_j by computing as follows.

$K_j = r_{ij}$ $h(X||ID_i||ID_j||K_i)$.

While C_j changes his secret key K_j to K_j', C_j has to submit the new secret key K_j' to TA through a secure channel. Then, TA updates the related public value with $r_{ij} = h(X||ID_i||ID_j||K_i)$ K_j' on the authenticated public board.

Suppose "C_j π C_i" and "C_k π C_j" are two adjacent directed edges and Cj is to be deleted from the hierarchy such that "C_k π C_i" becomes the directed edge. TA has to remove public value rij and change all the keys and their related public values for those nodes that are children of class C_j so as to prevent the class C_j from accessing the unauthorized resources by using the old keys. Assume "C_i π C_j" is an existing directed edge, and a new class Ck with secret key kk is to be inserted into the hierarchy such that C_i π C_k π C_j. Then, TA has to remove the old value r_{ji}, and add two new values r_{jk} and r_{ki}.

Later, Chien [3] proposed an efficient time-bound hierarchical key assignment scheme. The secret key deriving steps are the same as the previous work [2] as we describe above. For time period t, C_j will use $K_{j,t}$ to encrypt/decrypt his data. $K_{j,t}$ can be computed as follows.

$K_{j,t}= h(K_j$ $h^t(a)$ $h^{z-t}(b))$.

Here, $h()$ is a secure one-way hash function, $h^t(x)$ means applying hash function $h()$ t times, a and b are two random secret values kept by TA, and z is the upper bound of time period. Suppose an entity belonging to C_i wants to learn the data possessed by C_j at time t. He can derive the secret key K_j by computing as follows.

$K_j = r_{ij}$ $h(X||ID_i||ID_j||K_i)$

He also can obtain $K_{j,t}$ by computing as follows.

$h^t(a) = h^{t-t1}(h^{t1}(a))$,

$h^{z-t}(b) = h^{t2-t}(h^{z-t2}(b))$,

and $K_{j,t}= h(K_j$ $h^t(a)$ $h^{z-t}(b))$.

Here, $h^{t1}(a)$ and $h^{z-t2}(b)$ are stored in the tamper-resistance device and interval [$t1$, $t2$] is the valid time period of C_i.

3 Problem Solution

In this section, we propose an efficient and scalable hierarchical key assignment scheme without using smart card. Furthermore, our scheme allows all users to choose their favorite passwords as their secret keys. Notations used in our scheme are described as follows. $f(x)$ is a one-w*ay pseudo-random generator, x's length is n bits and* $f(x)$'s length is $2n$ bits. $f_0(x)$ and $f_1(x)$ are the left and right halves of the output of $f(x)$, respectively. C_j C_i means that the entities belonging to C_i have the rights to access the secret data held by the entities belonging to C_j.

3.1 Initialization Phase

First of all, TA must set up an authenticated public board. Assume that only TA can update the public board, but all system users are able to read it. Let C_i be the root user of the hierarchy. In our scheme, users have no tamper-resistant devices to perform computations for them. Therefore, TA provides an input device for users. The input device can derive the secret keys for users after they enter their secret keys and related information.

3.2 Adding a New Class

If C_j needs to be added into Figure 1(a) such that C_j becomes the left child node of C_i, and C_k and C_l become the left child node and the right child node of C_j, respectively, C_j must select a password as his secret key and sends his secret key K_j to TA. This result is shown in Figure 1(b) after the addition. TA then generates mask M_j and announces two masks M_k and M_l on the public board. $K_j = f_0(K_i)$ M_j, $K_k = f_0(K_j)$ M_k, and $K_l = f_1(K_j)$ M_l.

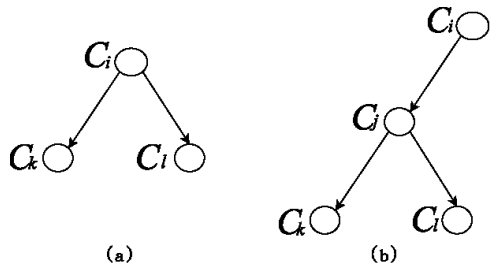

Figure 1: (a) Before inserting C_j, (b) after inserting C_j

3.3 Deriving the Secret Key

Suppose an entity belonging to C_i wants to derive the secret key K_j of C_j. He has to enter his secret key K_i and the mask M_j into the input device which is

provided by TA first. Next, the device can derive the secret key K_j by computing $K_j = f_0(K_i) \quad M_j$. As a result, the entity belonging to C_i can access the resources protected by the secret key K_j. We now consider a complex case, as shown in Figure 2. Suppose C_i is the parent node of C_j. C_k and C_l denote the left child node and the right child node of C_j, respectively. Suppose an entity belonging to C_i wants to derive the secret key K_l of C_l. He needs to enter his secret key K_i and two masks M_j and M_l into the input device. Then he can derive the secret key K_l by computing $K_l = f_1(f_0(K_i) \quad M_j) \quad M_l$.

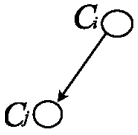

Figure 2: Derive the secret key of C_l by an entity in C_i

3.4 Updating the Secret Key

As shown in Figure 1(b), if C_j wants to change his secret key K_j to K'_j, he has to submit his new secret key K'_j to TA. TA will derive a new M'_j by computing $K'_j = f_0(K_i) \quad M'_j$. Then, TA will replace M_j with M'_j on the authenticated public board. Furthermore, TA also obtains M'_k and M'_l as follows, and then TA replaces M_k and M_l with M'_k and M'_l on the board. $K_k = f_0(K'_j) \quad M'_k$, and $K_l = f_1(K'_j) \quad M'_l$.

3.5 Deleting an Existing Class

If C_j has to be deleted from Figure 1(b), TA has to remove M_j from the public board first. Then TA updates M_k and M_l so that $K_k = f_0(K_i) \quad M_k$, and $K_l = f_1(K_i) \quad M_l$. Furthermore, all descendant nodes of C_j have to update their secret keys for security reason.

4 Two Extensions

We observe that sometimes the organization structure is more complex, which means each node could have three or more children nodes rather than only two children nodes. Besides, the time-bounded property is an important property to restrict each user's access period. In this section, we further describe two feasible extensions of our scheme for above two scenarios.

4.1 Three or More Children

In Section 3, we have described how to apply our scheme to the binary hierarchy. Furthermore, our scheme also can be applied to other hierarchical structures. Suppose C_i has three children nodes C_j, C_k, and C_l from left to right, respectively, as illustrated in Figure 3(a).

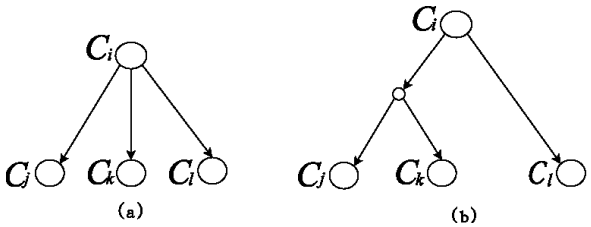

Figure 3: (a) Three children nodes belonging to C_i, (b) after inserting a virtual node

We can simply add a virtual node as the parent node of C_j and C_k as shown in Figure 3(b). The virtual node and C_l thus becomes the left child node and right child node of C_i, respectively. Next, C_i can derive the secret keys of C_j and C_k by computing $K_j = f_0(f_0(K_i)) \quad M_j$, and $K_k = f_1(f_0(K_i)) \quad M_k$. Using virtual nodes, our scheme can be easily applied to any kind of organization hierarchy structures.

4.2 Time-bound Issue

To avoid using the smart card and public-key infrastructure, we propose the following method to achieve the time-bound property. Assume that there are C_i and C_j in the hierarchy. TA has to publish four time parameters for each node on the public board. For C_i, TA computes $M_{i,t1}$ and $M_{i,t2}$ as follows. $M_{i,t1} = f_1(KT_i) \quad h^{ti,1}(x)$, and $M_{i,t2} = f_2(KT_i) \quad h^{z-ti,2}(y)$. For C_j, TA has to compute $M_{j,t1}$ and $M_{j,t2}$ as follows. $M_{j,t1} = f_1(KT_j) \quad h^{tj,1}(x)$, and $M_{j,t2} = f_2(KT_j) \quad h^{z-tj,2}(y)$. Here, $h()$ is a one-way hash function, x and y are the secret random numbers kept by TA, KT_i and KT_j are secret keys of C_i and C_j used for time-bound property, z is the upper bound of time period, and $[t_{i,1}, t_{i,2}]$ and $[t_{j,1}, t_{j,2}]$ are the valid time periods of C_i and C_j, respectively.

Suppose an entity belonging to C_i wants to derive the time-bound secret key K_{jt} of C_j at time t where $t_{i,1} \le t \le t_{i,2}$. He has to perform the following steps.

Step 1: Use his secret K_i to derive the secret key K_j of C_j by computing $K_j = f_0(K_i) \quad M_j$.

Step 2: Derive $h^{ti,1}(x)$, $h^{z-ti,2}(y)$, $h^t(x)$, and $h^{z-t}(y)$ by computing $h^{ti,1}(x) = M_{i,ti,1} \quad f_1(KT_i)$, $h^{z-ti,2}(y) = M_{j,ti,2} \quad f_2(KT_j)$, $h^t(x) = h^{t-ti,1}(h^{ti,1}(x))$, and $h^{z-t}(y) = h^{ti,2-t}(h^{z-ti,2}(y))$.

Step 3: Generate the time-bound secret key $K_{j,t}$ by computing $K_{j,t} = h(K_j \quad h^t(x) \quad h^{z-t}(y))$.

5 Security Analysis and Performance Evaluation

In this section, we will discuss the possible attacks from either outsiders or insiders first. Then, we will evaluate our performance in terms of the storage space, the implementation cost, and the computation loads.

5.1 Security Analysis

Attack by an outsider In our scheme, TA is in charge of computing and announcing the masks on the authenticated public board to help entities to access their subordinates. An outside attacker who has no right to access the authenticated public board might try to access these masks. However, it is infeasible for the attacker to get these masks since he cannot pass through the authentication mechanism. Although he can break the authentication mechanism, he still cannot use these masks to derive a secret key, because he has no chance to get any user's secret key. In our scheme, users keep their secret keys in their mind rather than in the tamper-resistant devices.

Attack by subordinates Assume C_j and C_k are the left child node and right child node of C_i, respectively. If an inside attacker A_j that belongs to C_j tries to derive the secret key K_i of C_i. He can use his secret K_j and mask M_j to derive $f_0(K_i)$ by computing $f_0(K_i) = K_j \quad M_j$. However, he can only drive the left half of $f()$, which can not help an attacker to derive the secret key K_i. Even though all the child nodes of C_i cooperate together to derive the secret key of C_i, they still can not learn K_i by inverting $f(K_i)$, because of the one-way property of $f()$.

5.2 Performance Evaluation

In this subsection, we will compare our scheme with other related works in terms of the time complexity, the storage space and the implementation costs. As shown in Table 1, our scheme needs n pseudo-random generators and n bit-wise XOR operations for deriving the secret key while Chien and Jan's scheme requires n hash functions and n bit-wise XOR operations, and Lin's scheme requires $(5n + 3)$ hash functions and $4n + 4$ bit-wise XOR operations. The time complexity of the pseudo-random generator, the hash function, and the bit-wise XOR are denoted as T_g, T_h, and T_{XOR}, respectively.

Table 2 summarizes three time-bound hierarchical key assignment schemes. We compare our method with others in terms of the time complexity, the storage space and the implementation costs. The comparison results show that the time complexity of our scheme is higher than that of Chien's, but lower than that of Tzeng's. Because we do not adopt a tamper-resistant device to store data, our scheme needs more storage space to store the public information than other schemes do. The notations used in Table 2 are described as follows. n is the number of nodes in the hierarchy, r is the number of child nodes of C_k. C_i, C_j, and C_k have the following relations $C_k \pi C_i$ and $C_j \pi C_k$. C_i is the user's class, and C_j is the target class. T_h denotes one hash function, Te represents one modular exponentiation, Tm means one modular multiplication, and T_l denotes one Lucas function.

6 Conclusion

In this paper, we propose an efficient and scalable hierarchical key assignment without using a tamper-resistant device. Our scheme allows all users to choose their favorite passwords to be their secret keys. The TA is in charge of generating the corresponding masks for entities' subordinates and announcing them on an authenticated public board. Therefore, legal entities can easily derive their subordinates' secret keys.

Our scheme not only successfully protects secret keys from being derived by outside attackers but also makes legal users unable to derive the secret keys by colluding with others. Meanwhile, our scheme is easily extended to support a complex hierarchy structure and to achieve the time-bound property.

References:

[1] S. G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," *ACM Transactions on Computer System*, Vol. 1, No. 3, Aug. 1983, pp. 239-248.

[2] H. Y. Chien, and J. K. Jan, "New Hierarchical Assignment without Public-Key Cryptography," *Computers and Security*, Vol. 22, No. 6, Sep. 2003, pp. 523-526.

[3] Hung-Yu Chien, "Efficient Time-Bound Hierarchical Key Assignment Scheme," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 10, Oct. 2004, pp. 1301-1304.

[4] L. Harn and H.Y. Lin, "A Cryptographic Keys Generation Scheme for Multilevel Data Security," *ACM Transactions on Computer Security*, Vol. 9, No. 6, Oct. 1990, pp. 539-546.

[5] C. H. Lin, C.C. Chang, and R.C.T. Lee, "Hierarchy Representation Based on Arithmetic Coding for Dynamic Information Protection Systems," *Information Sciences*, Vol. 64, No. 1-2, Oct. 1992, pp. 35-48.

[6] C. H. Lin, "Hierarchical Key Assignment without Public-key Cryptography," *Computers and Security*, Vol. 20, No. 7, Oct. 2001, pp. 612-619.

[7] R. S. Standhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control,"

Information Processing Letters, Vol. 27, No. 2, Feb. 1988, pp. 95-98.

[8] H. M. Tsai and C.C. Chang, "A Cryptographic Implementation for Dynamic Access Control in a User Hierarchy," *Computers and Security*, Vol. 14, No. 2, 1995, pp. 159-166.

[9] W. G. Tzeng, "A Time-bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, No. 1, 2002, pp. 182-188.

Table 1: Comparisons among our scheme, Chien and Jan's and Lin's schemes

	Ours	Chien and Jan's	Lin's
Implementation requirements	none	Low-cost smart card	Low-cost smart card
Number of public values	$n-1$	$n-1$	$2(n-1)$
N_1	$1T_g+1T_{XOR}$	$1T_h+1T_{XOR}$	$3T_h+4T_{XOR}$
N_2	$2T_g+2T_{XOR}$	$2T_h+2T_{XOR}$	$8T_h+8T_{XOR}$
N_n	nT_g+nT_{XOR}	nT_h+nT_{XOR}	$(5n+3)T_h$ $+(4n+4)T_{XOR}$

N_i: The number of operations to perform when deriving the secret key of a subordinate of i edges away, i=1,2, and n.

Table 2: Comparisons among our scheme, Chien's and Tzeng's

	Ours	Chien's	Tzeng's
Implementation requirements	none	Low-cost smart card	Public key infrastructure
Number of public values	$5n$	$n-1$	$n+6$
Number of computations when deriving the secret of one own class	$(t2-t1+1)T_h$ $+2T_g+2T_{XOR}$	$(t2-t1+1)T_h$	$(t2-t1)T_e$ $+(t2-t1)T_l+lT_h$
Number of computations when deriving the secret of one-edge distance child class	$(t2-t1+2)T_h$ $+2Tg+2T_{XOR}$	$(t2-t1+2)T_h$	$(t2-t1+r)T_e$ $+(t2-t1)T_l+lT_h$
Number of computations when deriving the secret of n-edge distance child class	$(t2-t1+1+l)T_h$ $+2T_g+2T_{XOR}$	$(t2-t1+1+l)T_h$	$(t2-t1+r)T_e$ $+(t2-t1)T_l+lT_h$