A new cryptographic key assignment scheme using a one-way hash function

Hui-Feng Huang[1] **and Chun-Chieh Liu**[2]

[1]Department of Information Management
National Taichung Institute of Technology, Taichung 404, Taiwan, R.O.C.
E-mail:phoenix@ntit.edu.tw
[2]Institute of Computer Science and Information Technology
National Taichung Institute of Technology, Taichung 404, Taiwan, R.O.C.
E-mail: s18943110@ntit.edu.tw

Abstract

Access control in a hierarchy plays an important role in today's communication networks. To guarantee the quality of communication services it is necessary to construct a low-computation dynamic access control scheme for the use to obtain the secure information. Therefore, we propose an efficient identity-based cryptographic key assignment scheme in which the cryptographic key of each user is calculated through a one-way hash function. The key generation and key derivation are quite simple and the number of the public/secret parameters for each authenticated user is small; which will make our scheme very suitable for the hardware-limited users such as mobile units
.

Keywords : *Access control; One-way hash function; key assignment*

1. Introduction

Owing to the growth of computer networks, and the fast progress of computer technologies on multi-user systems, sharing resources has become widespread over an open network. The administration of these resources is very important in a multi-user computer environment. Hence, access control through authorization becomes more important in the information security. The problem of access control for a hierarchy arises in organization which has a hierarchical structure. This can be used not only in military and government departments but also in private company. A hierarchy is constructed by dividing users into a number of disjointed security classes, $C_1, C_2 ..., C_m$, which are partially ordered with a binary relation"$\leq$". In a hierarchy, the class $C_j \leq C_i$ means that the security level of class C_j is lower than that of class C_i. The meaning of $C_j \leq C_i$ is that users in class C_i can access any information held by a user in class C_j, while the opposite is not allowed.

The cryptographic key assignment of access control in a partial-order hierarchy (called key assignment, hereafter) was firstly proposed by Akl and Taylor in 1983 [1]. Their key assignment is to assign a cryptographic key K_i to user U_i so that he can use K_i to derive K_j, if and only if, the user $U_j \leq U_i$. Hence, the user U_i can use K_j to decrypt the data, which are encrypted with K_j. However, their scheme is impractical when there are large number of users in a hierarchy because a great amount of key storage is required for each user. In addition, Akl and Taylor's scheme is unable to implement dynamic access control because the whole system has to be reestablished once a user insertion or deletion occurs. To reduce key storage and allow dynamic change of class, most researchers on the topic have concentrated on proposing schemes that either have better performance or allow inserting and deleting classes in the hierarchy [2, 3, 4, 5, 6, 7]. However, these previous key assignment schemes require several modular exponentiations to derive the lower class of a user's key, or they require a large amount of storage for public parameters. Recently, Yang and Li proposed a cryptographic key management solution to solve the above problem in the hierarchy [8]. In their scheme, each key belonging to a user in the hierarchy is computed through one-way hash functions. They provide a more efficient method to deal with the key derivation and generation in the hierarchy. However, they should construct many different one-way hash functions for key generation and derivation in the hierarchy. The number of hash functions is not fixed, it is dependent on the maximum number of direct child users in the hierarchy [8]. It is not convenient for applications because the one-way function has to be reestablished once a user is added or deleted. Therefore, we proposed a new key assignment scheme based on identify for dynamic access control in a hierarchy. By using our method, a one-way hash function is constructed and fixed which easily solves the dynamic access control problems such as adding/deleting users in the hierarchy. Each key that a user has in the hierarchy is derived through a one way hash function. Our scheme is very suitable for the limited computation capacities of users such as smart cards or mobile units.

The rest of this paper is organized as follows: In the next section, we propose our key management method for a hierarchy. The key assignment scheme for dynamic access control in a hierarchy is presented in Section 3. The security analysis is discussed in Section 4. Finally, a brief conclusion is given in Section 5.

2. The proposed key management method in a hierarchy

The system chooses a public one-way hash function $H(\ )$, and there is a central authority (CA) that generates and assigns a key to each user in a hierarchy. All secret parameters are managed by the CA. We assume that the partially ordered binary relationship of "$\le$" and ID_i is the identity number of user U_i. First, we will state some notations and terminology used in our scheme.

There are inheritance relationships among the nodes (users) in a hierarchy, the son U_i of a node U_j is defined as a direct child node of user U_j and U_j is defined as a direct parent node of U_i. Furthermore, the inheritance relation is transitive, that is, if node U_i is the son of U_j and U_j is the son of node U_k, then U_i is called as an indirect child node of U_k and U_k is called as indirect parent node of U_i. In order words, if it has relations $U_i \le U_j$ and $U_j \le U_k$, then it provides the relation $U_i \le U_k$. That is, a high-level node in the hierarchy can derive the keys of its direct or indirect child nodes.

We give an example of access control in a hierarchy as shown in Fig.1, where the user U_1 is a direct parent node of users U_2 and U_3, node U_2 is a direct parent node of users U_4 and U_5. Similarly, node U_3 is also a direct parent node of users U_5 and U_6. In this example, U_5 has two direct parents U_2 and U_3. Moreover, in a hierarchy, we call a node a dead-end node if it has no direct parent nodes, and other nodes are called non-dead-end nodes. For instance as shown in Fig.1, the user U_1 is a dead-end node, the other nodes U_2, U_3, U_4, U_5, and U_6 are non-dead-end nodes. The key K_i of user (node) U_i in the hierarchy is generated by a CA. Next, we will describe the proposed scheme with ID-base (identity base) as follows.

2.1. Key assignment:

(1) For a dead-end node, CA gives it an arbitrary key.

(2) If node U_j only has one direct parent node of U_i, then the key of $K_j = H(K_i, ID_j)$.

(3) If the node U_j has more than one direct parent node for $(U_{j1}, U_{j2}, \dots, U_{jt})$,and the key of $U_{j1}, U_{j2}, \dots, U_{jt}$ is $K_{j1}, K_{j2}, \dots, K_{jt}$, respectively, then the key of the node U_j will be $K_j = H(H(k_{j1}, ID_j), H(K_{j2}, ID_j), \dots, H(K_{jt}, ID_j))$. In order to derive the key K_j of node U_j for those of parent nodes $U_{j1}, U_{j2}, \dots,$ and U_{jt}, the CA

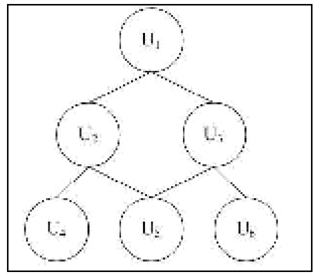

Fig.1 key of hierarchy

also delivers the information

$H(k_{j1}, ID_j), H(K_{j2}, ID_j), \dots, H(K_{j(r-1)}, ID_j), H(K_{j(r+1)}, ID_j),$

$H(K_{j(r+2)}, ID_j), \dots,$ and $H(k_{jt}, ID_j)$

to each direct parent U_{jr}, where $r = 1, 2, \dots, t$.

For example: As shown in Fig.1, the user U_1 is a dead-end node. CA gives an arbitrary key K_1 to U_1. Users U_2 and U_3 only have one direct parent U_1, then the keys for U_2 and U_3 are $K_2 = H(K_1, ID_2)$ and $K_3 = H(K_1, ID_3)$, respectively. The user U_4 also has one direct parent U_2, then CA assigns $K_4 = H(K_2, ID_4)$ to the user U_4. Because user U_5 has two direct parents U_2 and U_3, the key of U_5 is $K_5 = H(H(K_2, ID_5), H(K_3, ID_5))$. By the way, CA sends information $H(K_3, ID_5)$ and $H(K_2, ID_5)$ to the users U_2 and U_3, respectively.

3. Dynamic access control in a hierarchy

In this section, we will describe the adding and deleting of a user in our dynamic access control scheme.

3.1. Adding a user

When a new user is added, CA will decide he is a dead-end node or a non-dead-end node. If it is a dead-end node, CA will give a new arbitrary key for him, and all keys of its direct or indirect child nodes will be computed again. For Instance, In Fig.2, a new dead-end node U_n is added. CA gives a new key for U_n and all the keys of $U_1, U_2, ..., U_6$ will be regenerated according to our key management method.

If the newly added user U_j is not a dead-end node, then CA will only derive the key of U_j from its direct parent nodes. At the same time, the keys of U_j's direct and indirect child users will be updated. The remainder keys for other users will not be changed.

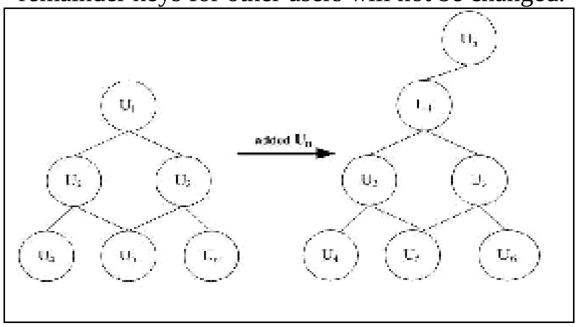

Fig.2 added a dead-end node

For instance, in Fig.3, if a new non-dead-end node U_7 is added in Fig.1, the key of user U_7 will be derived from its parent node U_1, that is $K_7 = H(K_1, ID_7)$, and the key of U_7's direct and indirect child nodes U_3, U_5, and U_6 will be regenerated by using our key assignment method. In this case, their keys will be $K_3 = H(K_7, ID_3)$, $K_5 = H(H(K_2, ID_5), H(K_3, ID_5))$, and $K_6 = H(K_3, ID_3)$. The other keys will not be updated.

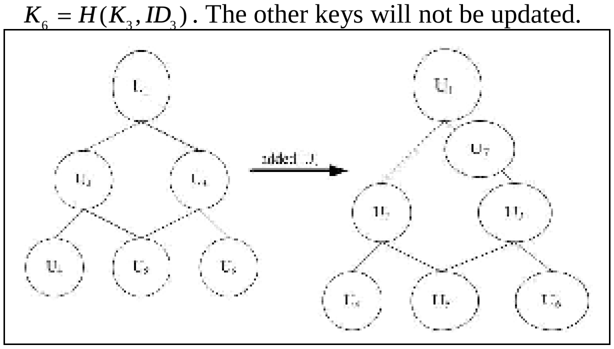

Fig.3 added a dead-end node

3.2. Deleting a node

If the deleted user U_j is a dead-end node, then it will have new dead-end node in the hierarchy after the deletion. CA will update the keys of U_j's direct or indirect child nodes. For example, as shown in Fig.4 we assume U_1 is deleted by Fig.1 and nodes U_2 and U_3 are new dead-end nodes after the deletion. Then, the keys of U_2, U_3, U_4, U_5, and U_6 will be regenerated by using our key assignment method.

If the deleted user U_j is not a dead-end node, then CA will only update the key of U_j's direct and indirect nodes according to our key assignment method. For instance, in Fig.5, we assume U_2 is deleted from Fig1, then the new keys for U_4 and U_5 are $H(K_1, ID_4)$ and

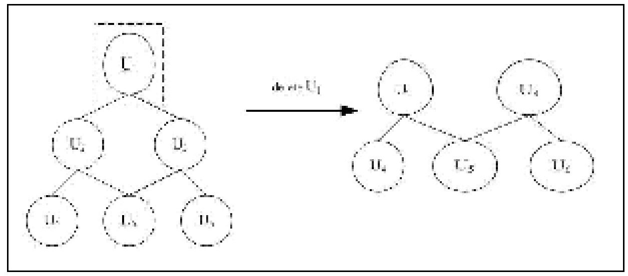

Fig.4 delete a dead-end node

$H(H(K_1, ID_5), H(K_3, ID_5))$, respectively. The other keys will not be changed.

Hence, the proposed scheme can be easily implemented as a dynamic access control because most of the keys need not be changed once a user is added or deleted. In this situation, the bit-length of a key is also equal to that of function *H*( ) which is not affected.

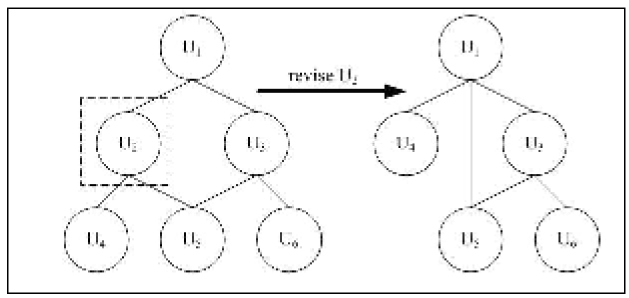

Fig.5 delete a non dead-end node

4. Security Analysis

In this section, we discuss the security analysis of the proposed key assignment method. In the proposed method, the key generation and key derivation in a hierarchy are obtained through a one-way hash function. By using our method if a user U_i only has one direct parent node U_j then the key of U_i will be $K_i = H(K_j, ID_i)$, where K_j is the key of U_j. Therefore, it is difficult for the user U_i to derive the key K_j of its parent node U_j from $H(K_j, ID_i)$. Because K_j is protected under the one-way hash function *H*( ). On the other hand, if a user S is one of the direct parent nodes of node *R*, then a user S may have some information to derive the key of user *R*. By using our method, user *S* cannot obtain the keys of other direct parent nodes of R. Because the keys of other direct parent nodes of

R are also protected under the one-way hash function *H*( ). Moreover, if a user is deleted, then CA will regenerate the key of his direct and indirect child nodes according to our key management method. Thus, when a user resigns from this system, he cannot decrypt the message with his old key. Therefore, based on the one-way hash function, the proposed scheme is secure. In addition, our key generation and key derivation are derived only through a one-way function. It is very simple and efficient for a user to obtain the information.

5. Conclusions

In this paper we presented a new cryptographic key assignment scheme with ID-base to reduce storage for the public (secret) parameters of each authenticated user. Moreover the proposed scheme has the following properties.

(1)Key generation and key derivation algorithms are quite simple and efficient.

(2)It can be easy to implement dynamic access control scheme because most other keys of the system need not to be immediately changed once a user is added or deleted.

(3)The bit-length of key is independent of the number of users in the hierarchy.

Hence, our scheme is very suitable for the low-computation users such as mobile users.

6. References

[1] Akl SG, and Taylor PD, "Cryptographic solution to a multilevel security problem", *Advances in cryptology*. pp. 237-249, 1982.

[2] S.J. Mackinnon, P.D. Taylor, H. Heojer, and S.G. Akl, "An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy", *IEEE Transactions on Computers*, vol. 34, no. 9, pp. 797-802, 1985.

[3] R.S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control", *Information Processing Letters*, no. 27, pp. 95-98, 1988.

[4] L. Harn and H.Y. Lin, "A Cryptographic Key Generation Scheme for Multilevel Data Security", *Computers and security*, vol. 9, no. 6, pp. 539-546, 1990.

[5] C.C. Chang, R.J. Hwang, and T.C. Wu, "Cryptographic Key Assignment Scheme for Access Control in a Hierarchy", *Information systems*, vol. 7, no. 3, pp. 243-247, 1992.

[6] D.P. Maher, "Crypto Backup and Key Escrow", *Communication of ACM*, vol. 39, no. 3, pp. 48-53, 1996.

[7] Nicolas Sklavos and Odysseas Koufopavlou," Access Control in Networks Hierarchy: Implementation of Key Management Protocol", International Journal of Network Security, Vol. 1, No. 2, pp. 103-109, 2005.

[8] C. Yang, C. Li "Access control in a hierarchy using one-way hash functions", *Computers and security* vol. 23, pp. 659-664, 2004.