Efficient Key Assignment for Hierarchical Access Control using One-way Hash Function

Shaohua Tang
School of Computer Science and Engineering
South China University of Technology
Guangzhou, Guangdong Province 510640
CHINA

Abstract: - An efficient hierarchical key assignment scheme for access control based on one-way hash function is proposed, which allows the user in a security class to derive the keys of its subordinating classes, so that the superior class can access the information encrypted and owned by its subordinating classes. This scheme is also flexible to change the hierarchical relationship among security classes, for example, it is not difficult to add new security classes to the hierarchy, or to delete security classes from the hierarchy dynamically. By analysis and comparison, it is shown that some features, such as efficiency, extensibility, and security, are possessed by this scheme.

Key-Words: - key assignment, hierarchy, access control, multilevel security, hash function, cryptography

1 Introduction
Users of a complex information system are often organized in a hierarchical structure for access control. In such a system, each user belongs to a security class, which is associated with a profile of access privileges. To achieve this goal, each security class is assigned a key for authentication, access control, and encryption, etc. It is often desirable to allow the user in a superior class to derive the keys of its subordinating classes. In this way, users in a superior security class don't need to keep the keys of its subordinating classes in order to access the information possessed by the subordinating classes. On the other hand, a user should never be able to derive the key of a security class that is not subordinate to its own class.

The hierarchical cryptographic key assignment problem was first studied by Akl and Taylor[1]. Sandhu's scheme[2] is very efficient, but it applies to a tree hierarchy only. Lin's scheme[3] and Shen-Chen's scheme[4] are highly flexible, but the extra storage needed is generally not small. Recently, some efficient schemes using one-way hash function appear. Both Chen-Chung's scheme[5] and Chen-Huang's scheme[6] take advantage of one-way hash function, but strong encryption algorithms are invoked in their schemes, which means that the computation overhead will be slightly higher than the ones using hash function only. Yang-Li's scheme[7] invokes hash functions only, where cryptographic algorithms with heavy computation overhead are not necessary. But there are some minor drawbacks in Yang-Li's scheme, for example, it is not easy to extend to large scale hierarchy, and a deleted node may be able to derive the keys of its former child nodes. We will give a detailed analysis in Section 5.3 of this paper.

Basically, a good cryptographic key assignment scheme in a hierarchy must satisfy the following requirements[4]: 1) The algorithms for the generation and derivation of secret keys should be very simple and efficient. 2) The system should be able to withstand the attacks made by collaborating some users to derive their predecessor's or sibling's keys. 3) The size of public parameters should be kept as small as possible. 4) The system should be flexible enough to handle the dynamic key management problems in an existing hierarchy.

In this paper, we present an efficient key assignment scheme for hierarchical access control using one-way hash function only. The basic requirements for hierarchical key assignment are satisfied. In addition, the features of our scheme include efficiency, extensibility, and security. The rest of this paper is organized as follows: some definitions and notations are given in Section 2. The key management scheme is presented in Section 3. In Section 4, we show some examples. The security properties, the time complexities, and the comparison with related works are analyzed in Section 5. Finally, we draw conclusions in Section 6.

2 Definitions and Notations
Let \( C_1, C_2, \ldots, C_n \) be \( n \) disjoint security classes. There is a partial order \( \preceq \) on the set \( S=\{C_i|1 \leq i \leq n\} \),
where $C_i \equiv C_j$ represents that $C_i$ is a successor of $C_j$, or, equivalently, $C_j$ is a predecessor of $C_i$. The dead-end node in the hierarchy is defined as the security class who has no predecessors. The non-dead-end node is the security class that has at least one predecessor.

If $C_i \equiv C_k \equiv C_j$, then we say that $C_k$ is an intermediate successor of $C_j$ between $C_i$ and $C_j$, and that $C_i$ is an intermediate predecessor of $C_j$ between $C_i$ and $C_j$. A security class is an immediate successor (resp., immediate predecessor) of another class, if there is no intermediate successor (resp., immediate predecessor) of another class.

The relationship between a security class $C_i$ and its immediate successor $C_j$ is a link, which is denoted as $<C_i, C_j>$. The link can be represented as a directed edge of a graph. The link $<C_j, C_i>$ is an incoming edge of node $C_i$, and an outgoing edge of node $C_j$.

The identity of security class $C_i$ is denoted as $ID_i$, and the identity can be represented as bit string or number in computer. The identities are public to all participants, and the identity of each security class should be unique, i.e., $ID_i \neq ID_j$, if $i \neq j$.

$H(x)$ is a secure one-way hash function that takes the input parameter $x$, where $x$ can be applied to a block of data or bit string in any size. The properties of a secure one-way hash function include: 1) It is computationally infeasible to derive $x$ according to the value of $H(x)$. 2) For any given variable $x$, it is computationally infeasible to find $y \neq x$ so that $H(y)=H(x)$. 3) It is computationally infeasible to find $x$ and $y$ so that $H(x)=H(y)$.

$H(x,y)$ is defined as $H(x,y)=H(x \oplus y)$, where $\oplus$ denotes the exclusive-or (XOR) operator. Obviously, $H(x,y)$ is also a secure one-way hash function that takes two input parameters $x$ and $y$.

Similarly, $H(x_1,x_2, ..., x_k)$ is defined as $H(x_1,x_2, ..., x_k)=H(x_1 \oplus x_2 \oplus ... \oplus x_k)$, where $k$ can be any positive integer.

The SA denotes the system authority, which is trusted by the whole system.

3 Key Assignment Scheme

3.1 Hierarchy Setup

The key for each security class $C_i$ in the hierarchy is generated by the SA according to the following rules:

1) For the dead-end nodes, the SA chooses random numbers as their keys.

2) For each link $<C_i, C_j>$, assuming that $C_j$ is the only immediate predecessor of $C_i$ as shown in Fig 1, and $C_j$’s key is $k_j$, then the key for security class $C_i$ is computed by the following equation:

\[
k_i = H(H(k_j, ID_i)) = H(H(k_j \oplus ID_i))
\]  

(1)

Fig 1. The security class $C_i$ has only one immediate predecessor $C_j$.

3) If the security class $C_i$ has more than one immediate predecessors: $C_{j_1}, C_{j_2}, ..., C_{j_m}$, as illustrated in Fig 2. Then the key for $C_i$ is calculated by the following equation:

\[
k_i = H(H(k_{j_1}, ID_i), H(k_{j_2}, ID_i), ..., H(k_{j_m}, ID_i))
\]

\[
= H(H(k_{j_1} \oplus ID_i) \oplus ... \oplus H(k_{j_m} \oplus ID_i)),
\]

(2)

where $k_{j_h}$ is the key of $C_{j_h}$, and $1 \leq h \leq m$.

Fig 2. The security class $C_i$ has more than one immediate predecessors.

3.2 Key Derivation

Suppose $C_i \equiv C_j$, $C_i$ can derive $C_j$’s key $k_j$ according to the following rules:

1) If $C_i$ is the only immediate predecessor of $C_j$ as shown in Fig 1, then $C_i$ can compute

\[
k_j = H(H(k_j, ID_i)) = H(H(k_j \oplus ID_i)).
\]

2) If $C_i$ has more than one immediate predecessors, assuming that $C_i$’s immediate predecessors are $C_{j_1}, C_{j_2}, ..., C_{j_m}$, where $m \geq 1$, then $C_j$ gets the secret parameters $H(k_{j_1}, ID_i)$, $H(k_{j_2}, ID_i)$, ..., and $H(k_{j_m}, ID_i)$ from the SA, where $k_{j_h}$ is the key of security class $C_{j_h}$, and $1 \leq h \leq m$. $C_j$ should keep the secret of $H(k_{j_1}, ID_i)$, ..., $H(k_{j_m}, ID_i)$ and not to expose them to others.

After that, $C_i$ can compute the key of the security class $C_j$ by the following equation:

\[
k_i = H(H(k_j \oplus ID_i), H(k_{j_1} \oplus ID_i), ..., H(k_{j_m} \oplus ID_i))
\]

\[
= H(H(k_j \oplus ID_i) \oplus H(k_{j_1} \oplus ID_i) \oplus ... \oplus H(k_{j_m} \oplus ID_i)).
\]
3.3 Changing Link Relationships

3.3.1 Adding a Link

Suppose $C_a$ and $C_b$ are security classes in the hierarchy. If a new link $<C_a, C_b>$ is added to the hierarchy, then the keys of $C_b$ and all $C_b$’s successors should be updated to adapt to the new link relationships according to the Rule 2) or 3) in Section 3.1.

3.3.2 Removing a Link

If a link $<C_a, C_b>$ is removed from the hierarchy, and none of the existing security classes become new dead-end nodes, then the keys of $C_b$ and all $C_b$'s successors should be updated to adapt to the new link relationships according to the Rule 2) or 3) in Section 3.1.

If a link $<C_a, C_b>$ is removed, and some of the existing security classes become new dead-end nodes, then the SA should choose a new random number as the key of the new dead-end node. After that, the SA should update the keys of $C_b$ and all $C_b$’s successors to adapt to the new link relationships according to the Rule 2) or 3) in Section 3.1.

3.3.3 Moving a Link

If a link is moved from $<C_a, C_b>$ to $<C_a', C_b>$, it is equal to removing the link $<C_a, C_b>$ and then adding a new link $<C_a', C_b>$. Therefore, we can handle this situation by updating the keys by the regulations specified in Section 3.3.2 and 3.3.1.

3.4 Addition of a Security Class

Security classes can be added to the hierarchy dynamically, but the keys of some other security classes may be affected. The adjustment to the keys should be made by the SA according to the following regulations.

1) If the added security class is a dead-end node, the SA will assign a new key to the newly added security class and re-generate the keys of all the successors of the new security class according to the Rule 2) or 3) in Section 3.1.

2) If the newly added security class is not a dead-end node, the SA will derive the key of this added node from its immediate predecessors. Also, the keys of all the added node's successors should be re-calculated according to the Rule 2) or 3) in Section 3.1.

3.5 Deletion of a Security Class

In order to prevent the deleted security class from deriving the key of its former successors, the keys of all its former successors and other affected security classes should be updated according to the following rules:

1) If the deleted security class is a dead-end node, then all its outgoing edges will be removed. If there are some security classes become the new dead-end nodes, the SA should choose a new random number as the keys of the new dead-end nodes. The keys of those nodes affected by the removed links should be updated according to the Rule 2) or 3) in Section 3.1.

2) If the deleted security class is a non-dead-end node, then all its incoming and outgoing edges are removed. Besides, some new links may be added to the hierarchy. If there are some security classes become new dead-end nodes, the SA should choose a new random number as the key of the new dead-end node. The keys of those nodes that are related with the removed links and added links should be updated according to the Rule 2) or 3) in Section 3.1.

4 Example

The diagram in Fig 3 demonstrates an example of security classes in a hierarchy. In this example, there are 6 security classes, and their identities are $ID_1$, $ID_2$, ..., and $ID_6$, respectively. We will take this example to illustrate the process of key assignment and key derivation.

![Fig 3. An example of security classes in a hierarchy.](image-url)

4.1 Key Assignment Example

In this example, $C_1$ is dead-end node. Thus, the SA chooses a random number $k_1$ as the key of $C_1$. For the security classes $C_2$, $C_3$, $C_4$ and $C_6$, since they have only one immediate predecessor, thus, the SA can compute their keys by the following equations:

$$k_2=H(H(k_1, ID_2))=H(H(k_1, ID_2)),$$
$$k_3=H(H(k_1, ID_3))=H(H(k_1, ID_3)),$$
$$k_4=H(H(k_2, ID_4))=H(H(k_2, ID_4)),$$
$$k_6=H(k_5, ID_6))=H(H(k_5, ID_6)).$$
The security class \( C_5 \) has more than one immediate predecessors, therefore, the SA calculates the key of \( C_5 \) by the equation
\[
k_5 = H(H(k_2, ID_5), H(k_3, ID_3)) = H(H(k_2 \oplus ID_5) \oplus H(k_3 \oplus ID_3)).
\]

4.2 Key Derivation Example

We take the security class \( C_2 \) as an example. Assume that the security class \( C_2 \) wants to derive the keys of \( C_4 \) and \( C_5 \).

Since the security class \( C_4 \) has only one immediate predecessor, \( C_2 \) can get the public information \( ID_4 \), and then calculates the key of \( C_4 \) by the equation
\[
k_4 = H(H(k_2, ID_4)) = H(k_2 \oplus ID_4).
\]

For another security class \( C_5 \), however, it has two immediate predecessors \( C_2 \) and \( C_3 \). \( C_2 \) should get the parameter \( H(k_3, ID_3) \) from the SA through a reliable channel firstly, then computes the key of \( C_5 \) by the equation
\[
k_5 = H(H(k_2, ID_5), H(k_3, ID_3)) = H(H(k_2 \oplus ID_5) \oplus H(k_3 \oplus ID_3)).
\]

5 Analysis

5.1 Security Analysis

Illegal derivation of keys can be prevented. By using one-way hash function, the users of a security class cannot derive the keys of their predecessors or siblings. We believe that as soon as the hash function is secure, then our scheme is secure based on the following considerations.

1) The user of a security class cannot derive the keys of its predecessors. Suppose the subordinating class \( C_i \) in Fig 1 wishes to derive its immediate predecessor \( C_i \)'s key \( k_i \). \( C_i \) possesses its own key \( k_i \) and knows the Equation(1), but \( C_i \) cannot derive \( C_i \)'s key \( k_i \) from the Equation(1) due to the one-way property of the hash function.

Suppose the subordinating class \( C_i \) in Fig 2 wishes to derive the keys of its immediate predecessors \( C_{i_1}, C_{i_2}, ..., C_{i_s} \). \( C_i \) possesses its own key \( k_i \). Similarly, due to the one-way property of the hash function, \( C_i \) cannot derive any \( k_{i_s} \), even combining the knowledge of Equation (2).

2) Even multiple users from different security classes conspire cannot derive the keys of their predecessors due to the security properties of one-way hash function.

The deletion of a security class or a link will not expose the keys of its former successors. As described in Section 3.3.3 and Section 3.5, if a link or a security class is removed from the hierarchy, a new key will be chosen for the new dean-end node, and the keys of the affected security classes will be updated by the SA to fit the new hierarchical security relationship. Therefore, our scheme can prevent the deleted security class from deriving the key of its former successor.

5.2 Time Complexity Analysis

In order to measure the complexities of the scheme, some additional variables should be defined. Let \( T_h \) denote the average time to compute the one-way hash function \( H(\cdot) \). \( n \) is the number of security classes in the hierarchy. \( n_d \) is the number of dead-end nodes. \( L \) is the number of links in the hierarchy.

The time complexity is discussed as follows.

1) Computation overhead during scheme setup phase

It is easy to deduce that the time needed to setup the hierarchical key structure is \( T_h \times (L + 2 \times (n - n_d)) \).

2) Key derivation time

The time for a security class to derive the key of its immediate successor is \( (2 + n_s) \times T_h \), where \( n_s \) is the number of the sibling nodes of the current security class.

5.3 Comparison with Related Works

The computation of hash function is faster than other operations, such as modular multiplication, exponentiation, and strong encryption. Thus, our scheme, which is based on one-way hash function only, will be more efficient than other hierarchical key assignment schemes based on modular multiplication, exponentiation, or strong encryption.

In the aspect of security, as soon as the one-way hash function used is secure, that is, the hash function satisfies the security properties mentioned in Section 2, then our scheme will be secure.

We are now going to compare our scheme with Yang-Li’s scheme[7], which is also based on one-way hash functions. Our scheme needs the same quantitative level of computation as Yang-Li’s scheme. But our scheme is superior to Yang-Li’s scheme in the following two aspects:

1) Flexibility and extensibility

Yang-Li’s scheme is not easy to extend to large scale hierarchy, since the maximum number of immediate successors of each security class can not be greater than a constant number chosen before the hierarchy setup. Once this constant number is fixed, then immediate successors can not be arbitrarily added to a certain security class.
However, the maximum number of immediate successors of each class is not limited in our scheme, which means that arbitrary number of immediate successors can be added to a security class in the hierarchy. Therefore, our scheme is more flexible and easy to extend.

2) Security
In Yang-Li's scheme, a deleted node may be able to derive the keys of its former child nodes, because the key values of the new dead-end nodes are kept after the deletion. But there isn't this potential security threat in our scheme.

6 Conclusion
An efficient hierarchical key assignment scheme based on one-way hash function is proposed in this paper. The basic requirements for hierarchical key schemes are satisfied. In addition, the following important features are possessed by the scheme:

1) Efficiency. Only hash function is adopted. Compared with other schemes using modular multiplication, exponentiation, or strong encryption, our scheme is overwhelmed in computation speed.

2) Flexibility and extensibility. Our scheme is suitable for the general partial order hierarchy, and it is not limited to tree structure. The maximum number of immediate successors of each security class and the level of the hierarchy are not limited. Besides, it can be convenient to dynamically change the link relationships, to add security classes to the hierarchy, or to delete security classes from the hierarchy.

3) Security. The security of our scheme is based on the security properties of one-way hash function. Illegal derivation of keys can be prevented. The deletion of a security class or a link will not expose the keys of its former successors.

Acknowledgement
This work is supported by National Natural Science Foundation of China under Grant No. 60572139 and No. 60273064, and supported by Fok Ying Tung Education Foundation under Grant No. 101069.

References: