A Cryptographic Key Assignment Scheme in a Hierarchy for Access Control *

Min-Shiang Hwang

Department of Information Management
Chaoyang University of Technology
168, Gifeng E. Rd., Wufeng,
Taichung County, TAIWAN 413, R.O.C.
Email: mshwang@mail.cyut.edu.tw
Fax: 886-4-3742337

September 14, 2001

*This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no.: NSC86-2621-E-324-001.

Abstract

Access control is one of mechanisms for data protection in a computer system. Many literatures based on cryptography have been proposed to solve the problem of access control in hierarchic structures. Recently, Liaw and Lei proposed an optimal heuristic algorithm for multilevel data security. But, their heuristic algorithm can only be used in a tree structure, which is a special case of a partially-ordered hierarchy. In this article, we present a modification of their algorithm that enables the algorithm to be used in a partially-ordered structure.

Keywords: Access control, Cryptography, Data security, Multilevel.

1 Introduction

In 1983, Akl and Taylor [1] proposed an elegant solution for controlling access to information among a group of users in a hierarchy. In such a hierarchy, the users and the information items they own are divided into a number of disjoint sets of security classes, $C_1, C_2, \cdots, C_n$, and the relationships among security classes correspond to a partially-ordered set (poset, for short) hierarchy, as shown in Figure 1.

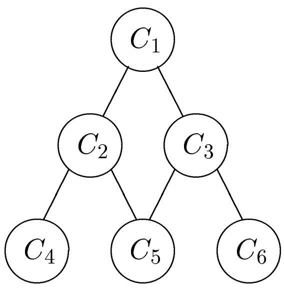

Figure 1: An example of a partially ordered hierarchy.

For the partially-ordered structure, let C_i and C_j be nodes in this structure. Furthermore, $C_i \leq C_j$ where C_i is called the descendant of C_j and C_j is called the ancestor of C_i. If there is another existing node C_k such that $C_i \leq C_k \leq C_j$, C_i is called an immediate descendant (or children) of C_j, and C_j is called an immediate ancestor (or father) of C_i. If there is no another node C_i such that $C_i \leq C_j$, then C_j is called a leaf security class.

In the Akl-Taylor scheme [1], each security class C_i is 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. The information items owned by C_i are encrypted by an available symmetric (one-key) cryptosystem with the enciphering key K_i. This information can only be retrieved by the security class C_j, where $C_i \leq C_j$. Using the public parameters, PB_i and PB_j, and the secret key K_j, C_j can derive K_i to decipher the information items owned by C_i. The Akl-Taylor scheme suggests an elegant solution in a poset hierarchy for the access control problem. However, a large amount of storage to store the public parameters is required. In 1985, MacKinnon et al. [2] presented an improved algorithm for the Akl-Taylor scheme, called the canonical assignment, to reduce the value of public parameters. This scheme certainly reduces the number of distinct primes. However, a large amount of storage is still needed to store the PB_i. In addition, it is difficult to find an optimal canonical algorithm, which can make the assignment optimal for an arbitrary poset hierarchy. In 1988, Sandhu [3] 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) name and its immediate ancestor's secret key through a one-way function. In the scheme, no extra public parameter is needed for the key derivation. However, there are two drawbacks: one is that computational overhead is incurred in deriving keys. The other is that the proposal is only implemented in a tree hierarchy, which is less flexible in applications. In 1990, Harn and Lin [4] 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 to store the public parameters for most security classes is 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, Chang et al. [5] and in 1993 Liaw et al. [6] proposed other schemes. Their schemes are 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 [7, 8].

Recently, Liaw and Lei [9] presented an optimal heuristic algorithm for assigning cryptographic keys in a tree structure for multilevel data security. The Liaw-Lei scheme uses a top-down design approach, as does the Akl-Taylor scheme [1]. 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 heuristic algorithm can only be used in a tree structure, which is a special case of a partially-ordered hierarchy. In this article, we present a modification of their algorithm that enables the algorithm to be used in a partially-ordered structure; that is, a user at a higher level can derive the keys of other users below him from his own cryptographic key.

Table 1: Algorithm for assigning a prime to each node.

Input:	A hierarchy H
Output:	A set of n_{R_i}, primes or the products of several primes, assigned to the nodes of H
	/* Compute n_{R_i} for each node R_i starting from the leaves and going up to the root.*/
1.	For each non-root node R_i do
2.	if R_i has only one child and no sibling
3.	then $n_{R_i} = p_1 \prod n_{\underline{R_i}}$
4.	else $n_{R_i} = p_e$, where p_e is the smallest prime differs from all other primes previously used in $\underline{R_i}$ and lR_i;
5.	$n_{R_0} = 1$;

2 A Modified Liaw-Lei Scheme

In this section, we introduce two algorithms created by modifying the algorithm Assignment in [9]. We assume that, as in the Akl-Taylor, Hanr-Lin, and other schemes, there is a central authority (CA, for short) in the system. The main task of the CA is to generate secret keys, public identity integers, and derivation keys for all security classes. In Table 1, corresponding to phase 1 of the algorithm Assignment in [9], we assign each node R_i of the hierarchy a prime or the product of several primes, n_{R_i}, starting from the leaves and working up to the root. In Table 2, corresponding to phase 2 of the algorithm Assignment in [9], we compute a public parameter t_{R_i} for each node R_i, starting from the root and working down to the leaves. For convenience, we use the following notation in our proposed algorithms:

$\overline{R_i}$: denotes the father node of R_i,

$\underline{R_i}$: denotes the children nodes of R_i,

lR_i: denotes the sibling nodes of R_i, and

$\overline{lR_i}$: denotes the uncle nodes of R_i. That is, the relationship of $\overline{lR_i}$ to $\overline{R_i}$ is that of sibling.

We call a node crossbred if the node has two or more father nodes.

Whenever all nodes have been assigned a public parameter by the algo-

Table 2: Algorithm for computing public parameters.

Input:	A hierarchy H with primes assigned
Output:	A set of t_{R_i}, public parameters, assigned to each node of H
	/* Compute t_{R_i} for each node R_i starting from the root and going down to leaves.*/
1.	$t_{R_0} = 1$;
2.	For each non-root node R_i do
3.	if R_i is non-crossbred node
4.	then $t_{R_i} = p_1 t_{\overline{R_i}} \prod n_{lR_i}$
5.	else $t_{R_i} = p_1 lcm(t_{\overline{R_i}}) \prod n_{lR_i}$, and
6.	for all $R \in \overline{lR_i}$ do $t_R = t_R n_{R_i}$;

rithms in Table 1 and Table 2, the CA computes the secure key K_{R_i} for each node R_i as follows.

$$K_{R_i} = K_0^{t_{R_i}} \bmod m, \tag{1}$$

where m is a multiplication of two large prime p and q; K_0 is a random secret integer. K_0 and m are relatively prime. K_0, p, and q are kept secret and m is public.

Once the secret node keys K_{R_i} and the corresponding public information parameter t_{R_i} are generated, the node R_i can derive the secret key K_{R_j} for its descendant R_j by computing the following:

$$K_{R_j} = K_{R_i}^{t_{R_j}/t_{R_i}} \bmod m \quad \text{if} \quad R_j \leq R_i. \tag{2}$$

Next, we will show that the key derivation in Equation(2) is correct as follows.

$$\begin{aligned} & K_{R_i}^{t_{R_j}/t_{R_i}} \bmod m \\ = \ & K_0^{t_{R_i} \frac{t_{R_j}}{t_{R_i}}} \bmod m \\ = \ & K_j. \end{aligned}$$

3 Security Analysis

In this section, we prove the correctness of our security enforcement scheme. We need to satisfy the following two statements:

- A user at a higher level can derive the keys of the other users below him from his own secret key, while the opposite is not allowed.
- Two or more users collaborating at a lower level of the system cannot derive a higher-level key to which they are not entitled.

Akl and Taylor [1] proved the following two conditions:

$$t_{R_b} | t_{R_a} \text{ if and only if } R_a \leq R_b; \tag{3}$$

$$\gcd_{R_b \ngeq R_a} t_{R_b} \not| t_{R_a}. \tag{4}$$

To ensure the security of our scheme, we also insert a limit,

$$\frac{\min_{R_j \ngeq R_i} t_j}{\gcd_{R_j \ngeq R_i} t_j} = 1,$$

as in [9], to prevent collaborated attacks. We prefer to reference [9] for details. Thus Equation (4) is rewritten as follows:

$$\text{if } \frac{\min_{R_b \ngeq R_a} t_{R_b}}{\gcd_{R_b \ngeq R_a} t_{R_b}} \neq 1 \qquad \text{then collaborating users have no privileges}$$
$$\text{else } \gcd_{R_b \ngeq R_a} t_{R_b} \not| t_{R_a}. \tag{5}$$

In the following we verify the security of our scheme.

Lemma 3.1 *The proposed scheme satisfies Equation (3).*

Proof. We divide the proof into the following two cases.

Case 1: if $R_a \leq R_b$ then $t_{R_b} | t_{R_a}$.

For convenience, we rewrite step 4 and step 5 in Table 2 as follows:

$$\begin{cases} t_{R_i} = t_{\overline{R_i}} \prod n_{lR_i} p_1, \\ t_{R_i} = lcm(t_{\overline{R_i}}) \prod n_{lR_i} p_1. \end{cases} \tag{6}$$

From the above equations, we know that

$$t_{\overline{R_i}} | t_{R_i}. \tag{7}$$

Thus, this case satisfies Equation (3).

Case 2: if $R_a \not\leq R_b$ then $t_{R_b} \not| t_{R_a}$.

We divide the proof into the following three subcases.

Case 2.1: neither R_a or R_b is a crossbred node.

In this subcase, the proof is the same as Lemma 1 in [9].

Case 2.2: R_a is a crossbred node.

We divide the proof into the following three subcases.

Case 2.2.1: $R_b < R_a$

Since t_{R_b} has the factor of $t_{\overline{R_b}} p_1$, by transitivity, t_{R_b} has the factor of $t_{R_a} p_1^i$, where $i \geq 1$. Thus, $t_{R_b} \not| t_{R_a}$.

Case 2.2.2: $R_b \leq \overline{lR_a}$, that is, R_b is a uncle node of R_a or descendant node of $\overline{lR_a}$.

From step 6 in Table 2, we know that t_{R_b} has the factor of n_{R_a}, but t_{R_a} does not contain n_{R_a}. Thus $t_{R_b} \not| t_{R_a}$.

Case 2.2.3: $R_b > \overline{lR_a}$.

Let the common ancestor of R_a and R_b be $\overline{\overline{R_{ab}}}$. Assume that $\overline{\overline{R_a}}$, the ancestor of R_a, and $\overline{\overline{R_b}}$, the ancestor of R_b, are children nodes of $\overline{\overline{R_{ab}}}$. From Equation (6), we can easily see that both t_{R_a} and t_{R_b} contain the product $t_{\overline{\overline{R_{ab}}}} p_1$. Thus, after division by t_{R_a} and t_{R_b}, t_{R_a} does not contain $n_{\overline{\overline{R_a}}}$, but t_{R_b} does. Thus, $t_{R_b} \not| t_{R_a}$ and this subcase satisfies Equation (3).

Case 2.3: R_b is a crossbred node.

We divide the proof into the following three subcases.

Case 2.3.1: $R_a > R_b$

In this subcase, the proof is the same as in case 2.2.1.

Case 2.3.2: $R_a \leq \overline{lR_b}$.

From Equation (6), we know that

$$t_{\overline{R_b}} | t_{R_b} \tag{8}$$

$$t_{\overline{R_b}} \not| t_{\overline{lR_b}} \tag{9}$$

The above equations implying $t_{R_b} \not| t_{\overline{lR_b}}$. From Equation (6), we know that t_{R_b} contains $n_{\overline{lR_b}}$, but R_a does not. Thus, $t_{R_b} \not| t_{R_a}$ and this subcase satisfies Equation (3).

Case 2.3.3: $R_a > \overline{lR_b}$.

In this subcase, the proof is similar to that for Case 2.2.3. **Q.E.D**

Lemma 3.2 *The modified scheme satisfies Equation (5).*

Proof. The proof is similar to that of Lemma 2 in [9]. **Q.E.D**

From Lemma 3.1 and 3.2, the following theorem is derived directly.

Theorem 3.1 *The modified scheme satisfies Equation (3) and (5).*

4 Conclusions

We have proposed an access control scheme for a partially-ordered hierarchy by modifying Liaw-Lei's scheme, which can only be used in a tree structure. We have also shown that the modified scheme prevents cooperative attacks.

References

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

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

[3] Sandhu, R.S., "Cryptographic Implementation of a Tree Hierarchy for Access Control", *Information Processing Letters*, Vol. 27, 1988, pp. 95–98.

[4] Harn, L. and Lin, H.Y., "A Cryptographic Key Generation Scheme for Multilevel Data Security", *Computers & Security*, Vol. 9 No. 6, Oct. 1990, pp. 539–546.

[5] Chang, C.C., Hwang, R.J., and Wu, T. C., "Cryptographic Key Assignment Scheme for Access Control in a Hierarchy", *Information Systems*, Vol. 17 No. 3, 1992, pp. 243–247.

[6] Liaw, H.T., Wang, S.J., and Lei, C.L., "A Dynamic Cryptographic Key Assignment Scheme in a Tree Structure", *Computers and Math. with Applic.*, Vol. 25 No. 6, 1993, pp. 109–114.

[7] Hwang, M.S. and Yang, W.P., "Attacks on A Dynamic Cryptolographic Key Assignment Scheme in a Tree Structure", *Submitted for publication*, 1994.

[8] Hwang, M.S., Chang, C.C., and Yang, W.P., "Modified Chang-Hwang-Wu Access Control Scheme", *IEE Electronics Letters*, Vol. 29 No. 24, Nov. 1993, pp. 2095–2096.

[9] Liaw, H.T. and Lei, C.L., "An Optimal Algorithm to Assign Cryptographic Keys in a Tree Structure for Access Control", *BIT*, Vol. 33, 1993, pp. 46–56.