A Cryptographic Key Generation Scheme Without A Trusted Third Party For Access Control In A Hierarchy

Min-Shiang Hwang[†] Chwei-Shyong Tsai[‡]

Department of Management Information system[‡]
National Chung Hsing University
250 Kuo Kuang Road, 402 Taichung, Taiwan, R.O.C.

Department of Information Management[‡]
National Taichung Institute of Technology
129 Sec. 3, San-min Rd., Taichung, Taiwan 404, R.O.C.

Abstract

In this paper, we propose a new dynamic cryptographic key generation scheme for access control in a hierarchy with frequently inserted and deleted security classes. Our scheme, based on the Chinese remainder theorem and an available symmetric cryptosystem, can achieve the following four goals: First, a so-called trusted third party for generating keys is not needed in the system. Second, each security class can decide and choose a secret key independently of the other classes. Third, when a security class is inserted to or deleted from the hierarchy, we recompute only the derivation key of its immediate ancestor, without affecting the keys of the other classes in the hierarchy. Fourth, the storage space needed for the public information of each class can be as far as possible reduced.

Keywords: Access control, Cryptography, Data security, Partially-ordered hierarchy.

1 Introduction

It is a major issue to prevent important information from being destroyed, altered, disclosed or copied by unauthorized users in computer protection systems. For this reason, an access control model was introduced for its easy implementation and was widely used in computer protection systems [14].

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 hierarchy, as shown in Figure 1. In the Akl-Taylor

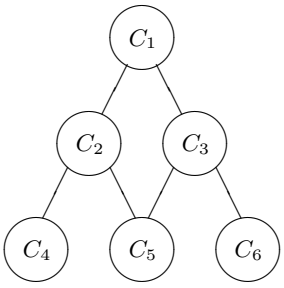

Figure 1: An example of partially ordered hierarchy

scheme [1], each security class C_i is assigned a distinct prime for public information, PI_i. The secret key, SK_i, for each security class C_i is calculated using the public information PI_i by a trusted third party in the system. The information items owned by C_i are encrypted by an available symmetric (one-key) cryptosystem with the enciphering key SK_i. This information can only be retrieved by a security class C_j, where $C_i \leq C_j$. Using the public information, PI_i and PI_j, and the secret key SK_j, C_j can derive the secret key SK_i and decipher the information items owned by C_i. However, a large amount of storage is required for storing the public information. In the past decade, many authors have proposed ways to reduce the storage space needed for storing the public information.

In 1985, MacKinnon et al. [17] presented an improved algorithm for the Akl-Taylor scheme to reduce the size of the public information. In 1988, Sandhu [20] used one-way functions to create a cryptographic implementation of a tree hierarchy for access control. This approach has one drawback: computational overhead is incurred in deriving keys.

In 1990, Harn and Lin [5] proposed an approach somewhat similar to the Akl-Taylor scheme. Instead of using a top-down design approach, as in the Akl-Taylor scheme, however, Harn and Lin presented a bottom-up key generating scheme. In 1993 Liaw et al. [15, 16] proposed other schemes. Their schemes are based on Newton's interpolation method and a predefined one-way function. However, their schemes are insecure against cooperative attacks [6, 7, 8, 9]. In 1997, Shen, Chen, and Lai [21] presented a novel cryptographic key assignment scheme for dynamic access control in a hierarchy. Their scheme requires a large computation time to generate and derive keys [10, 11, 12].

In 1998, Yeh et al. proposed an efficient cryptographic key assignment scheme for solving the access control problem in a hierarchy [22]. Their 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 [13].

In this paper, we propose a new scheme without trusted third party for access control in a hierarchy. This paper is organized as follows. In the next section, the new dynamic scheme, based on the Chinese remainder theorem, is presented. In Section 3, we verify the security of our scheme. The computation for generating and deriving keys and the storage of public information are discussed in Section 4. Finally, conclusions are given in Section 5.

2 The proposed Scheme

In this section, we present an implementation scheme for access control in a hierarchy that is based on the Chinese Remainder Theorem (CRT). Suppose a conventional cryptosystem, such as DES-like cryptosystems [18], is available. Since DES-like is a secure private-key cryptosystem, the outlaws cannot break our scheme. Let E be an enciphering procedure and let D be a deciphering procedure of the available cryptosystem. Let SK be the secret key. Given a plaintext message M, we obtain $X = E_{SK}(M)$ and $M = D_{SK}(X)$, where X is the ciphertext of the plaintext M. Let $C_1, C_2, \cdots, C_n$ be n security classes in a hierarchy. Each C_i possesses three values: the secret key SK_i, a prime P_i and public information PI_i. Only an ancestor of security class C_i can derive the secret key SK_i.

Since our access control scheme is based on the CRT, we first introduce the theorem.

Theorem 2.1 (Chinese Remainder Theorem [3]) *Let $P_1, P_2, \ldots, P_r$ be pairwise relatively prime integers and let $y = P_1 \times P_2 \times \cdots \times P_r$. Then the system of equations*

$$H = H_i \bmod P_i, \text{ for } i = 1, 2, \ldots, r$$

has a common solution H in the range $[0, y-1]$.

By the CRT, we can construct a cryptographic key generation scheme for access control in a hierarchy by taking H as the secret information (SI_i), P_i as public information (P_{ij}), and H_i as the secret key (SK_{ij}) of the j-th immediate descendant (C_{ij}) of C_i. For convenience, we assume that $C_{ij}, j = 1, 2, \cdots, r$, denotes the j-th immediate descendant of C_i in the following algorithms. The algorithm for generating the secret key, secret information, and the corresponding public information for each security class is stated as follows.

Algorithm Key-Generation

Step 1: Select and publish an existing available symmetric cryptosystem E, such as DES-like.

Step 2: Each security class, C_i, chooses his public information, a large prime P_i, and secret key SK_i, such that $SK_i \leq P_i$ and $\gcd(P_i, P_j) = 1$, for $i \neq j$.

Step 3: Find a node C_i in the partially-ordered hierarchy by in-order traversal.

Step 4: If C_i is not a leaf node, then do the following:

Step 4.1: Compute the secret information SI_i for C_i by CRT as follows:

$$SI_i = SK_{ij} \bmod P_{ij}, \text{ for } j = 1, 2, \cdots, r, \qquad (1)$$

where SK_{ij} and P_{ij}, for $j = 1, 2, \cdots, r$, are the secret key and public information of C_{ij}, the immediate descendants of C_i. Since $SK_{ij} < P_{ij}$, SI_i can been evaluated by CRT as follows [3]:

$$SI_i = \sum_{j=1}^{r} SK_{ij} G_j G'_j \bmod PP, \qquad (2)$$

where $PP = \prod_{j=1}^{r} P_{ij}$, $G_j = PP/P_{ij}$, and G'_j is an integer such that $G_j G'_j \bmod P_{ij} = 1$.

Step 4.2: Compute the public information PI_i of C_i as follows:

$$PI_i = E_{SK_i}(SI_i)$$

where SK_i is the secret key of C_i. Thus, only the security class C_i can decipher the public information PI_i, which is an encrypted version of the secret information SI_i.
Step 5: Repeat from Step 3 until all nodes of the hierarchy are completely examined.

Note that we do not need a trusted third party to generate keys in the above algorithm. Each class C_i generates its own secret key and public information and computes the secret information associated with his immediate descendant's secret keys. Each C_i possesses three values: the secret key SK_i which is kept secret, a prime P_i and public information PI_i. P_i and PI_i are public.

For each security class C_i, once the secret key SK_i, secret information, and public information have been determined, we can easily derive the secret key of C_i's immediate descendants. We assume that C_i's immediate descendants are $C_{i1}, C_{i2}, \cdots, C_{ir}$ and that their public information is P_{ij}, $j = 1, 2, \cdots, r$. The key derivation algorithm is stated as follows.

Algorithm Key-Derivation
Step 1: Compute $SI_i = D_{SK_i}(PI_i)$, where D_{SK_i} is a decipher function of DES-like under the secret key of C_i. Since PI_i is a public information, C_i gets secret information, SI_i, by the decipher function.
Step 2: Derive the secret key SK_{ij} of C_{ij} as

$$SK_{ij} = SI_i \bmod P_{ij}. \tag{3}$$

Note that the security class C_i also has the ability to derive any of its descendant's secret keys by performing the key derivation algorithm iteratively.

Our scheme achieves true dynamic control in the sense that only a single key need to be modified each time we insert a new security class or relationship among classes, delete an existing security class or relationship among classes, or update a security class or relationship among classes. When a new security class or relationship among classes is inserted into the hierarchy, the corresponding secret key, secret information, and public information will be determined immediately by algorithm Key-Generation without changing any previously defined secret keys or public information. Only the secret information (SI_i) and the public information of the immediate ancestor of the new security class are changed as follows:

$$\begin{aligned} SI_{inew} &= [SI_{iold} + SK_{i(r+1)} G_{(r+1)} G'_{(r+1)}] \bmod PP, \\ PI_{inew} &= E_{SK_i}(SI_{inew}). \end{aligned}$$

where $SK_{i(r+1)}$ denotes the secret key of $C_{i(r+1)}$, which is the $(r+1)$th immediate descendant of C_i and the inserted class, $PP = P_{i1} \times P_{i2} \times \cdots \times P_{i(r+1)}$, $G_{(r+1)} = PP/P_{i(r+1)}$ and $G'_{(r+1)}$ is a value such that $G_{(r+1)} \times G'_{(r+1)} \bmod P_{i(r+1)} = 1$.

When an existed security class C_{ij} is removed from the hierarchy, the secret key, secret information and public information of C_{ij} is simply dropped without changing any previously defined secret keys or public information. If the security class is a leaf node, only the secret information (SI_i) and public information (PI_i) of the immediate ancestor (C_i) of the removed security class are changed as follows:

$$\begin{aligned} SI_{inew} &= [SI_{iold} - SK_{ij} G_j G'_j] \bmod PP, \\ PI_{inew} &= E_{SK_i}(SI_{inew}). \end{aligned}$$

If the security class is a non-leaf node, the corresponding secret information and public information of the immediate ancestor (C_i) of the removed security class are changed as that of the above inserting these new security classes (the immediate descendants of the removed security class).

3 Security Analysis

There are two cases to consider in the security analysis. One is whether a security class C_i can indeed derive the keys of other security classes C_j if $C_j \leq C_i$ using C_i's own cryptographic key. In contrast, this is impermissible if $C_i \not\geq C_j$. The second case is whether the scheme provides security against two or more security classes collaborating to derive a higher level key. In the following, we verify the security of our method in respect to these two cases.

Theorem 3.1 *The proposed scheme satisfies that $C_j \leq C_i$ if and only if SK_j can be derived by C_i with SK_i, where SK_i and SK_j are the keys of C_i and C_j, respectively.*

Proof. We divide the proof into the following two cases.
Case 1: If $C_j \leq C_i$ then SK_j can be derived by C_i with SK_i.
When C_i is an immediate ancestor of C_j, from Steps 1 and 2 of algorithm Key-Derivation in Section 2 we know that

$$\begin{aligned} SI_i &= D_{SK_i}(PI_i), \\ SK_{ij} &= SI_i \bmod P_{ij}. \end{aligned}$$

Thus, it is clear that SK_{ij} can be derived by C_i with SK_i. By transitivity, SK_j can also be derived by C_i with SK_i when C_i is an ancestor of C_j.
Case 2: If SK_j can be derived by C_i with SK_i then $C_j \leq C_i$.
This case is equivalent to showing that if $C_j \not\leq C_i$ then SK_j cannot be derived by C_i with SK_i. Since

in our scheme the secret key of each security class is determined randomly and independently by that class, no one can know the secret key of another class except by knowing directly or indirectly the secret information of the immediate ancestor of that class. C_i, $C_i \not\geq C_j$, cannot evaluate the secret derivation key (SI_j) directly or indirectly from Step 1 of algorithm Key-Derivation in Section 2 unless C_i is able to attack the existing cryptosystem. Thus, in this case the security of our scheme depends on that of the existing cryptosystem. □

If the available cryptosystem can be attacked then our scheme can be attacked. In this case, then the illegal users can obtain the secret derivation keys (SI_i) of all C_is. It is easy to evaluate the secret key SK_{ij} of the jth immediate descendant of C_i from Equation (3). Thus, our scheme is insecure.

If our scheme can be broken, the illegal users can obtain the secret information SI_i from the public information PI_i of C_i. This means that the illegal users can use known-plaintext to attack the following equation in Step 1 of the algorithm Key-Derivation in Section 2:

$$SI_i = D_{SK_i}(PI_i),$$

where D is a deciphering procedure of the existing cryptosystem.

It is well known that the existing sysmetric cryptosystem, e.g., DES-like, can withstand known-plaintext attacks [3]. Therefore, our scheme can withstand two or more security classes collaborating to derive a higher level secret information.

4 Computation and Storage Space Complexity

Since Steps 1 and 2 of the algorithm Key-Generation in Section 2 require constant time to compute, we ignore them here. Steps 3 and 5 are iterative steps for computing the public information PI_i of all security classes. Thus, the algorithm requires n times the computation for constructing the secret information of a single class if there are n security classes in our hierarchy. To compute the secret information SI_i for C_i by the CRT [3] from Step 4.1 of the algorithm Key-Generation in Section 2 requires a total of $2r$ multiplications, $(r-1)$ additions, r divisions, and one module operation. Let $t_{op}(g,h)$ denote the time cost of an "op" operation (i.e., multiplication, division, addition, or module) with two bits g and h. We assume that the length of each key is h bits. The computation of Step 4.1 of algorithm Key-Generation is as follows:

$$\begin{aligned} t_{4.1} &= 2rt_{multiplication}(rh,h) + (r-1)t_{addition}(rh,h) \\ &\quad + rt_{division}(rh,h) + t_{module}(rh,h), \end{aligned}$$

If DES-like is used as the available cryptosystem in our scheme, it partitions the data text into pieces of 64 bits each. The computation of Step 4.2 of algorithm Key-Derivation is as follows:

$$t_{4.2} = r * \lceil h/64 \rceil DES(64),$$

where DES(64) is the time required to encipher 64 bits of text (secret information) using the DES-like device. The total processing time of Step 4 of the algorithm Key-Generation is

$$t_4 = t_{4.1} + t_{4.2}.$$

Next, we investigate time complexity of the algorithm Key-Derivation in Section 2. The computation time of Step 1 of the algorithm Key-Derivation is the same as that of Step 4.2 of the algorithm Key-Generation shown in Equation (4). The computation time of Step 2 of the algorithm Key-Derivation requires only one module operation. Thus, the computation time needed for the algorithm Key-Derivation is as follows:

$$t_{KD} = t_{4.2} + t_{module}(rh,h).$$

Dirr and Taylor [4] have designed a fast and efficient hardware implementation of the CRT in residue arithmetic. Their method incurs a time cost of $70\lceil \log_2 r \rceil$ ns for computing the equation $SI_i = SK_{ij} \bmod P_{ij}$, for $j = 1, 2, \cdots, r$. It needs only 0.7 seconds to evaluate all secret information with 10^6 security classes and 1000 immediate descendants for each class. Thus, our scheme is practical to implement.

Next, we investigate the storage space needed for the public information PI_i and P_i. From Equation (2), $SI_i = \sum_{j=1}^{r} SK_{ij} G_j G'_j \bmod \prod_{j=1}^{r} P_{ij}$, so the length of SI_i is at most that of $\prod_{j=1}^{r} P_{ij}$. Since $SK_i \leq P_i$, the length of each P_i requires at least h bits if that of SK_i is h. In order to reduce the space of P_i, we modify Equation (1) as follows.

$$SI_i = SK_{ij} \bmod P_{ij}^{t_{ij}}, \text{ for } j = 1, 2, \cdots, r,$$

where t_{ij} is an integer such that the length of $P_{ij}^{t_{ij}}$ is greater than that of SK_{ij}. If the length of SK_{ij} is 512 bits, then we can choose P_{ij} with 24 bits and an integer t_{ij} with 6 bits. The space needed for public information is apparently less than that needed for the secret key SK_{ij}s.

We compare our scheme with other famous schemes, Akl-Taylor scheme [1] and Harn-Lin [5] in computation and storage space. In Akl-Taylor

scheme, the public information and secret information of each security class are

$$\begin{aligned} PI_i &= \prod_{C_j \not\leq C_i} P_j, \\ SI_i &= K_0^{PI_i} \bmod m, \end{aligned}$$

where P_j is a large prime, m is the product of p and q, which are two random large primes, and K_0 is a random secret key, where $2 \leq K_0 \leq m-1$, $\gcd(K_0, m) = 1$. In Harn-Lin scheme, the public information and secret information of each security class are

$$\begin{aligned} PI_i &= \prod_{C_j \leq C_i} P_j, \\ SI_i &= g^{\prod_{C_j \leq C_i} d_j} \bmod m, \end{aligned}$$

where (P_j, m) and d_j are the same as the public key and secret key in RSA cryptosystem [2, 19].

Since the secret information need to compute in exponential operation in Akl-Taylor and Harn-Lin schemes, their schemes are inefficient than our scheme because the secret information need to compute in multiplication and addition operations in our scheme. As to the storage space, Akl-Taylor scheme requires to store $\prod_{C_j \not\leq C_i} P_j$, Harn-Lin scheme requires to store $\prod_{C_j \leq C_i} P_j$, and our scheme requires to store $\prod_{C_j < C_i} P_j$. It is obvious that the public information space in Akl-Taylor scheme is larger than that of Harn-Lin scheme. In addition, the public information space of our scheme is less one P_j than that of Harn-Lin scheme.

5 Conclusions

We have proposed a new dynamic cryptographic key generation scheme to handle the problem of access control in a partially-ordered hierarchy. Our scheme has the following advantages:

1. Each security class can determine its own secret key without relying on a trusted third party.

2. When a new security class is added to the hierarchy, the scheme needs only to compute the keys for the new class and update the secret information of its immediate ancestor. There is no need to change the keys of every security class.

3. When an existing security class is removed from the hierarchy, the scheme merely drops the keys of that class and updates the secret information of its immediate ancestor. Again, there is no need to change the keys of every security class.

4. The number of public information generated is small. Therefore, the scheme utilizes memory space efficiently.

5. The procedures for key generation and derivation are simple.

References

[1] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transactions on Computer Systems*, vol. 1, pp. 239–248, Jul 1983.

[2] Chin-Chen Chang and Min-Shiang Hwang, "Parallel computation of the generating keys for RSA cryptosystems," *IEE Electronics Letters*, vol. 32, no. 15, pp. 1365–1366, 1996.

[3] Dorothy E. R. Denning, *Cryptography and Data Security*. Massachusetts: Addison-Wesley, 1982.

[4] William Jr. Dirr and Fred J. Taylor, "On Implementing the CRT in Residue Arithmetic," *The Journal of Computers Math.*, vol. 17, pp. 155–163, July 1985.

[5] Lein Harn and Hung Yu Lin, "A cryptographic key generation scheme for multilevel data security," *Computers & Security*, vol. 9, pp. 539–546, Oct. 1990.

[6] Min-Shiang Hwang, Chin-Chen Chang, and Wei-Pang Yang, "Modified Chang-Hwang-Wu access control scheme," *IEE Electronics Letters*, vol. 29, pp. 2095–2096, Nov. 1993.

[7] Min-Shiang Hwang, "A cryptographic key assignment scheme in a hierarchy for access control," *Mathematical and Computer Modelling*, vol. 26, no. 2, pp. 27–31, 1997.

[8] Min-Shiang Hwang, "Extension of CHW cryptographic key assignment scheme in a hierarchy," *IEE Proceedings Computers and Digital Techniques*, vol. 146, no. 4, p. 219, 1999.

[9] Min-Shiang Hwang, "An improvement of a dynamic cryptographic key assignment schemes in a tree hierarchy," *Computers & Mathematics with Applications*, vol. 37, no. 3, pp. 19–22, 1999.

[10] Min-Shiang Hwang, "An improvement of novel cryptographic key assignment scheme for dynamic access control in a hierarchy," *IEICE Transactions on Fundamentals Of Electronics, Communications and Computer Sciences*, vol. 82-A, no. 3, pp. 548–550, 1999.

[11] Min-Shiang Hwang, "A new dynamic cryptographic key generation scheme in a hierarchy," *Nordic Journal of Computing*, vol. 6, no. 4, pp. 363–371, 1999.

[12] Min-Shiang Hwang, "An asymmetric cryptographic scheme for a totally-ordered hierarchy," *International Journal of Computer Mathematics*, vol. 73, pp. 463–468, 2000.

[13] Min-Shiang Hwang, "Cryptanalysis of YCN key assignment scheme in a hierarchy," *Information Processing Letters*, vol. 73, no. 3, pp. 97–101, 2000.

[14] Min-Shiang Hwang, Wen-Guey Tzeng, , and Wei-Pang Yang, "An access control scheme based on chinese remainder theorem and time stamp concept," *Computers & Security*, vol. 15, no. 1, pp. 73–81, 1996.

[15] H. T. Liaw, S. J. Wang, and C. L. Lei, "A dynamic cryptographic key assignment scheme in a tree structure," *Computers Math. Applic.*, vol. 25, no. 6, pp. 109–114, 1993.

[16] Horng-Twu Liaw and Chin-Laung Lei, "An optimal algorithm to assign cryptographic keys in a tree structure for access control," *BIT*, vol. 33, pp. 46–56, 1993.

[17] Stephen J. Mackinnon, Peter D. Taylor, Henk Meijer, and Selim G. Akl, "An optimal algorithm for assigning cryptographic keys to control access in a hierarchy," *IEEE Transactions on Computers*, vol. 34, pp. 797–802, Sep. 1985.

[18] S. Miyaguchi, "The FEAL cipher family," in *Advances in Cryptology, CRYPTO'90*, pp. 727–638, Lecture Notes in Computer Science, Vol. 435, August 1990.

[19] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.

[20] R. S. Sandhu, "Cryptographic implementation of a tree hierarchy for access control," *Information Processing Letters*, vol. 27, pp. 95–98, 1988.

[21] Victor R.L. Shen, T. S. Chen, and F. Lai, "Novel cryptographic key assignment scheme for dynamic access control in a hierarchy," *IEICE Transactions on Fundamentals*, vol. E80-A, no. 10, pp. 2035–2037, 1997.

[22] J. H. Yeh, R. Chow, and R. Newman, "A key assignment for enforcing access control policy exceptions," in *Proceedings on International Symposium on Internet Technology*, pp. 54–59, Taipei, 1998.