A New Dynamic Key Generation Scheme for Access Control in a Hierarchy

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
http://www.cyut.edu.tw/∼mshwang/

October 31, 2012

Abstract

In this paper, we propose a new dynamic cryptographic key generation scheme for access control in a hierarchy. Our method can achieve the following three goals. First, the storage space needed to store public information is smaller than that required in the previous work. Second, when a security class is added to the hierarchy, we assign a secret key and a public derivation key to the security class without affecting the keys of the other security classes in the hierarchy. Third, when a security class is deleted from the hierarchy, we simply erase the keys of that security class in the hierarchy and change the derivation key of its immediate ancestor.

CR Categories: D.4.6.

Keywords: Access control, cryptography, data security, multilevel.

1 Introduction

Many modern society are based on hierarchical structure of users such as military, government organizations, school systems, private corporations, and computer management systems. An example of a three-level hierarchical structure is shown in Figure 1. 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_6. C_1 is the security class with top level which possesses the greatest authority.C_4, C_5, and C_6 are the security classes with bottom level which have the least authority. It is an extremely important research topic that how to control access to information items in such an environment [1].

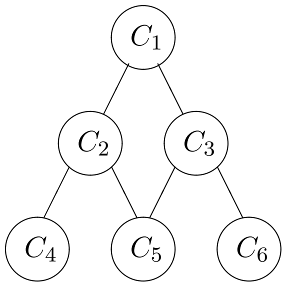

Figure 1: An example of a partially ordered hierarchy.

In the past decade, many key generation schemes for access control in the hierarchy are proposed [1, 2, 3, 5, 6, 9, 10, 11, 12, 14, 15, 16]. However, while inserting a new security class to or deleting an existed security class from a hierarchy, their schemes still have to recompute almost all keys.

In this paper, we propose a new scheme for access control in a partially-ordered hierarchy. Our method allows new keys to be determined easily whenever a security class is inserted into or removed from a hierarchy and also reduces the amount of storage space needed to store public information.

This paper is organized as follows. In the next section, the new scheme is presented. In section 3, we verify the security of our scheme. In section 4, we discuss the computation needed to generate keys and the storage required for public information. Section 5 is the conclusion of the paper.

2 A dynamic key generation scheme for access control in a hierarchy

In this section, a dynamic access control scheme is developed for a hierarchical system. Suppose a conventional cryptosystem, such as DES, is available. Let E be an enciphering procedure and let D be a deciphering procedure of the available cryptosystem. Let K be the secret key. Given a plaintext M, we get $C = E_K(M)$ and $M = D_K(C)$, where C is the ciphertext of the plaintext M. We assume that there is a central authority (CA) in the system. The main task of the CA is to generate secret keys, public identity integers, and derivation keys for all security classes. Let $C_1, C_2, \cdots, C_n$ be n security classes in a hierarchy. Each C_i owns three values: the secret key K_i, public identity integer which is an ordinary index, and public derivation key PD_i which is encrypted in association with K_i. Only an ancestor of security class C_i can derive the secret key K_i of C_i. This means that C_j can derive K_i if and only if $C_i \leq C_j$.

In order to show how the proposed scheme works, we shall first introduce a theorem, which will play an important role in the development of our scheme.

Theorem 2.1 *Assume that $K > \max(a_j)$, for $j = 1, 2, \cdots, n$, where a_j is the j-th item of the sequence $\{a_j | j = 1, 2, \cdots, n\}$. There exists an integer M such that a_j is equal to $\lfloor M/K^{j-1} \rfloor \bmod K$.*

Proof. Let M be the number of

$$\begin{aligned} M &= \sum_{j=1}^{n} a_j K^{j-1} \\ &= a_n K^{n-1} + a_{n-1} K^{n-2} + \cdots + a_2 K + a_1. \end{aligned}$$

Then we need to prove

$$\lfloor M/K^{j-1} \rfloor \bmod K = a_j, \quad 1 \leq j \leq n$$

where $\lfloor \cdot \rfloor$ indicates a floor function. We have

$$\begin{aligned} & \lfloor M/K^{j-1} \rfloor \bmod K \\ = \ & \lfloor (KQ + a_j + R) \rfloor \bmod K \end{aligned}$$

where $Q = a_{j+1} + a_{j+2}K + \cdots + a_n K^{n-j-1}$, and $R = (a_{j-1}/K) + (a_{j-2}/K^2) + \cdots + (a_1/K^{j-1})$.

Since

$$\begin{aligned} & a_{j-1}K^{j-2} + a_{j-2}K^{j-3} + \cdots + a_2 K + a_1 \\ \leq \ & (K-1)(K^{j-2} + K^{j-3} + \cdots + K + 1) \\ = \ & K^{j-1} - 1 \\ < \ & K^{j-1}, \end{aligned}$$

this implies that

$$\lfloor R \rfloor = 0.$$

Since KQ and a_j are integers, therefore

$$\begin{aligned} & \lfloor (KQ + a_j + R) \rfloor \bmod K \\ = \ & (KQ + a_j + \lfloor R \rfloor) \bmod K \\ = \ & a_j \bmod K + \lfloor R \rfloor \bmod K \\ = \ & a_j. \end{aligned}$$

Therefore, the theorem holds. Q.E.D.

According to Theorem 2.1, we can construct a cryptographic key generation scheme for access control in a hierarchy by taking M as derivation key (SD_i) and K as secret key (K_i) of C_i and a_j as secret key (K_{ij}) of the j-th immediate descendent

(C_{ij}) of C_i. The algorithm for generating the secret key and derivation key for each security class is stated as follows.

Algorithm Key-Generation

Step 1: Select and publish an existing available cryptosystem, such as DES.

Step 2: Assign every node C_i a random secret key K_i such that $K_j \leq K_i$ for all $C_j \leq C_i$.

Step 3: Suppose C_i has r immediate descendants, denoted by $C_{i1}, C_{i2}, \cdots, C_{ir}$. Assign every descendant a public integer j such that $1 \leq j \leq r$ and every two descendants have distinct integers. Denote a descendant with an integer j by C_{ij}.

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

Step 4.1: Compute the secret derivation key SD_i for C_i as follows.

$$SD_i = \sum_{j=1}^{r} K_{ij} K_i^{j-1}, \tag{1}$$

where K_{ij} for $j = 1, 2, \cdots, r$, is a secret key of C_{ij}, the immediate descendent of C_i.

Step 4.2: Compute the public derivation key PD_i of C_i as follows.

$$PD_i = E_{K_i}(SD_i), \tag{2}$$

where E is an enciphering procedure of the available cryptosystem.

Note that $K_{ij} < K_i$ for all $C_{ij} < C_i$ of the above algorithm. To realize this, the existing cryptosystem used in our scheme must be remain secure even if a certain number, say l, of the most significant key bits are fixed and known. We may divide the security classes of the hierarchy in a totally ordered set of ranks, such that if $C_j < C_i$ then C_i has a higher rank than C_j. E.g. in Figure 1, C_1 are of the highest rank, C_2 and C_3 of the second rank, and C_4, C_5, and C_6 of the third rank. Now the existing cryptosystem can accommodate 2^l ranks numbered by $2^l - 1, \cdots, 1, 0$ from the highest to the lowest. If C_i is of rank t, then fix the l most significant bits of K_i to be the binary representation of t, and generate the rest of the key bits randomly. Then automatically, $K_j < K_i$ if $C_j < C_i$. In addition, the keys K_i are not necessarily integers in DES, but in the key generation and derivation they must be interpreted as integers.

For each security class C_i, once the secret key K_i, derivation key, and public identity integer have been determined, we can easily derive the immediate descendent's secret key. The key derivation algorithm is stated as follows.

Algorithm Key-Derivation

Step 1: Let C_i's immediate descendents be $C_{i1}, C_{i2}, \cdots, C_{ir}$, with public identity integer j.

Step 2: Compute $SD_i = D_{K_i}(PD_i)$, where D is the deciphering procedure of the available cryptosystem.

Step 3: Derive the secret key K_{ij} of C_{ij} as

$$K_{ij} = \lfloor SD_i / K_i^{(j-1)} \rfloor \bmod K_i. \tag{3}$$

Note that the security class C_i can compute the keys of lower security classes that are not its immediate descendent by performing the key derivation algorithm iteratively.

When a new security class is inserted into the hierarchy, the corresponding secret key, derivation key, and public identity will be determined immediately by algorithm Key-Generation without changing any previously defined secret keys and public information. Only the derivation key (SD_i) of the immediate ancestor of the new security class need be changed, as follows:

$$SD_{inew} = SD_{iold} + K_{i(r+1)} K_i^r, \tag{4}$$

where $K_{i(r+1)}$ denotes a secret key of $C_{i(r+1)}$, the $(r+1)$-th immediate descendent of C_i.

When an existing security class C_{ij} is removed from the hierarchy, the secret key, derivation key, and public identity of C_{ij} are simply dropped, without changing any previously defined secret keys and public information. Only the derivation key (SD_i) of the immediate ancestor (C_i) of the security class being removed must be changed, as follows:

$$SD_{inew} = SD_{iold} - K_{ij}K_i^{j-1}. \tag{5}$$

The public identity number j is then reserved for a future security class.

3 The security

There are two cases to consider concerning security. One is that a security class C_i should be able to derive the keys of the other security classes C_j, if $C_j \le C_i$, using its own cryptographic key. Conversely, this is prohibited if $C_i \not\geq C_j$. Case two is that the scheme must provide security against two or more security classes collaborating to derive a higher level key. In the following, we prove that our method is secure.

Theorem 3.1 *The proposed scheme satisfies $C_j \le C_i$ if and only if K_j can be derived by C_i with K_i, where K_i and K_j are the keys of C_i and C_j, respectively.*

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

Case 1: If $C_j \le C_i$ then K_j can be derived by C_i with K_i.

When C_i is an immediate ancestor of C_j, from step 2 and step 3 of algorithm Key-Derivation in Section 2 we know that

$$\begin{aligned} SD_i &= D_{K_i}(PD_i), & (6)\\ K_{ij} &= \lfloor SD_i/K_i^{(j-1)} \rfloor \bmod K_i. & (7) \end{aligned}$$

Thus, it is well known that K_{ij} can be derived by C_i with K_i. By transitivity, K_j can also be derived by C_i with K_i when C_i is an ancestor of C_j.

Case 2: If K_j can be derived by C_i with K_i then $C_j \le C_i$

This case is equivalent to stating that if $C_j \not\leq C_i$ then K_j cannot be derived by C_i with K_i. Two methods may be in use to derive K_j by C_i with K_i. One is that to decompose Equation (1) in Section 2. The other is that to attack the existing cryptosystem from Equation (2) in Section 2. Even if two or more security classes C_{ij} (for all $C_{ij} \not\geq C_j$) collaborating, it is hard to get both SD_j and K_j by solving one equation in two variables from Equation (1). In addition, since we assume that the existing cryptosystem used in our scheme is of high security, C_i cannot evaluate the secret derivation key directly or indirectly from step 2 of algorithm Key-Derivation in Section 2. Q.E.D.

From step 4.1 of algorithm Key-Generation in Section 2, we know that

$$K_i^{r-1} < SD_i < K_i^r, \tag{8}$$

where r is the number of immediate descendents of C_i. Whenever illegal users can obtain the secret derivation key (SD_i) of all C_i's, it is easy to evaluate the secret key K_i of C_i from Equation (8). However, the illegal users need to solve Equation (6) for getting SD_i. It is well known that existing cryptosystems can withstand known-plaintext attacks [4]. Our scheme thus can also withstand two or more security classes collaborating to derive a higher level secret derivation key.

4 Complexity analysis and comparisons

In this section, we investigate the time complexity of the secret derivation key SD_i in the above algorithm. Let $N(a^b)$ denote the number of multiplications needed to compute a^b. Knuth [13] showed that the upper bound of $N(a^b)$ is identical to $2\lfloor \log_2 b \rfloor$ by using addition chain methods. Since the derivation key is defined in Equation (1) as $SD_i = \sum_{j=1}^{r} K_{ij} K_i^{j-1}$, we have $N(SD_i) = \sum_{j=1}^{r} 1 + N(K_i^{j-1}) \leq r + 2r\lfloor \log(r-1) \rfloor = O(r \log r)$, where r denotes the number of immediate descendents of C_i. If we ignore the overflow problem of key values, the time complexity of the derivation key SD_i for C_i in Equation (1) is upper bounded by $(r \log r)$.

Finally, we investigate the storage space needed for the public derivation key PD_i. From Equation (1), $SD_i = \sum_{j=1}^{r} K_{ij} K_i^{j-1} \leq \sum_{j=1}^{r} (K_i - 1) K_i^{j-1} = K_i^r - 1$. Therefore, the secret derivation keys in our scheme required storage space of $O(r \log K_i)$. Since DES effectively transforms a 64-bit plaintext block into a 64-bit ciphertext block using a 56-bit key, it has the advantage of no message expansion. In general, the length of the key is constant (56 bits for DES) in conventional cryptosystems [4]. Thus, the storage space needed for the public derivation key PD_i for C_i is $O(r)$.

Next, we compare our method with other cryptography-based hierarchy schemes in the literature. Previous methods are briefly reviewed as follows.

Akl and Taylor [1] assigned each security class C_i an associated distinct prime e_i and calculated the public parameter PB_i as follows:

$$PB_i = \prod_{C_j \not\leq C_i} e_j. \tag{9}$$

Thus, the secret key for all security classes can be computed as follows:

$$K_j = K_0^{PB_j} \bmod m, \tag{10}$$

where m is the product of p and q, which are two random large primes, and K_0 is a random secret key, where $2 \leq K_0 \leq m - 1$, $\gcd(K_0, m) = 1$. A descendant's key can be derived from an ancestor's key by the formula

$$K_j = K_i^{PB_j / PB_i} \bmod m, \text{ iff } C_j \leq C_i. \tag{11}$$

The scheme of MacKinnon, Akl, and Taylor is the same as the Akl-Taylor scheme, except that the former use the chain decomposition to determine the primes to be used. Each public parameter PB_i of a descendant in the chain is the power exponential of PB_j owned by the ancestor node.

Harn and Lin [5] presented a cryptography-based hierarchy scheme, based on factoring a product of two large primes which is difficult. Instead of using the top-

Table 1: The computational complexity.

Poset hierarchy scheme	Appendability	Removability	Key generation	Key derivation
Akl and Taylor's	$O(n)$	$O(n)$	$O(n \log n)$	$O(n \log n)$
MacKinnon et al.'s	$O(n)$	$O(n)$	$O(n \log n)$	$O(n \log n)$
Harn and Lin's	$O(\log n)$	$O(1)$	$O(n \log n)$	$O(n \log n)$
Our method	$O(1)$	$O(1)$	$O(r \log r)$	$O(r \log r)$

down design approaches, as in the Akl-Taylor scheme, the Harn-Lin scheme used a bottom-up key generating procedure.

The computational and storage complexity of each method introduced are summaries in Tables 1, 2.

Table 1 shows four aspects of the computational complexity: appendability, removability, key generation, and key derivation. In the key generation column and key derivation column, $O(n \log n)$ must be recomputed in the previous methods, where n denotes the total number of security classes in the system. Since the space of the n'th prime is $(n \log n)$ bits, the time complexity of the Equation (10) is $O(n \log n)$ by using addition chain methods. However, our scheme needs $O(r \log r)$ only, where r denotes the number of immediate descendents of security class. In general, $r << n$.

In the appendability column and removability column, $O(n)$ keys must be recomputed in Akl-Taylor's scheme and MacKinnon et al.'s scheme. Whenever, a new security class is added to or an exiting security class is removed from the system, these schemes need reconstruct the secret key for all security classes. In the same case, Harn-Lin's scheme need reconstruct the secret key for $(\log n)$ security classes in average. When an existing security class is removed from the hierarchy, Harn-Lin's scheme simple drop the keys of the removed class. However, when a new security class is added to or removed from the hierarchy, our scheme needs only update the derivation key of its immediate ancestor.

Table 2 shows three aspects of storage space: number of primes, maximal pub-

Table 2: The space complexity.

Poset hierarchy scheme	Number of primes	Maximal public parameter	Storage spaces
Akl and Taylor's	n	$\prod_{i=1}^{n} p_i$	$O(n^3 \log n)$
MacKinnon et al.'s	c	$\prod_{i=1}^{n} p_i$	$O(n^3 \log c)$
Harn and Lin's	n	$\prod_{i=1}^{n} p_i$	$O(n^3 \log n)$
Our method	–	$\prod_{i=1}^{r} K_i$	$O(rn)$

lic key, and storage spaces. Akl-Taylor's and Harn-Lin's scheme needs n primes. MacKinnon et al.'s scheme needs c primes, the number of chains. However, our scheme needs no prime. In the maximal public key column, there are only r keys (K_i) to be multiplied in our scheme. In the storage spaces column, our scheme total requires $O(rn)$ bits. However, Akl-Taylor's and Harn-Lin's schemes total require $O(n^3 \log n)$ bits, and MacKinnon et al.'s scheme total requires $O(n^3 \log c)$ bits, where $O(n^3 \log n) > O(n^3 \log c) \gg O(rn)$.

5 Conclusions

We have proposed a new dynamic cryptographic key generation scheme for access control in a partially-ordered hierarchy. The scheme can be used in multilevel database encryption systems [7, 8]. Our scheme has the following advantages:

1. When a new security class is added to the hierarchy, we need only compute the keys for the new class and update the derivation key of its immediate ancestor. The keys of the other security classes remain unaffected.

2. When an existing security class is removed from the hierarchy, we simply drop the keys of the removed class and update the derivation key of its immediate ancestor. Again, the keys of the other security classes are unaffected.

3. The size of public derivation keys generated is small, so the scheme utilizes memory space efficiently.

4. The key generation and derivation procedure is simple.

Acknowledgements

The authors wish to thank many anonymous referees for their suggestions to improve this paper. The material in this paper was partially presented at the 9th IEEE TENCON, Singapore, 1994. Part of this research was supported by the National Science Council, Taiwan, R.O.C., under contract no. NSC88-2213-E-324-002.

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*, 1(3):239–248, Jul 1983.

[2] C.C. Chang, C.H. Lin, and R.C.T. Lee. Hierarchy representations based on arithmetic coding for dynamic information protection systems. *Information Sciences*, 64:35–48, 1992.

[3] C.C. Chang, R.J. Hwang, and T.C. Wu. Cryptographic key assignment scheme for access control in a hierarchy. *Information Systems*, 17(3):243–247, 1992.

[4] D.E.R. Denning. *Cryptography and Data Security*. Addison-Wesley, Massachusetts, 1982.

[5] L. Harn and H.Y. Lin. A cryptographic key generation scheme for multilevel data security. *Computers & Security*, 9(6):539–546, Oct. 1990.

[6] M.S. Hwang, C.C. Chang, and W.P. Yang. Modified Chang-Hwang-Wu access control scheme. *IEE Electronics Letters*, 29(24):2095–2096, Nov. 1993.

[7] M.S. Hwang and W.P. Yang. A two-phase encryption scheme for enhancing database security. *Journal of Systems and Software*, 31(12):257–265, Dec. 1995.

[8] M.S. Hwang and W.P. Yang. Multilevel database security with subkeys. *Submitted for Publication*, Nov. 1995.

[9] M.S. Hwang. A cryptolographic key assignment scheme in a hierarchy for access control. *Mathematical and Computer Modelling*, 26(2):27–31, 1997.

[10] M.S. Hwang. An improvement of a dynamic cryptographic key assignment scheme in a tree hierarchy. Accepted and to appear in *Computers & Mathematics with Applications.*

[11] M.S. Hwang. An improvement of novel cryptographic key assignment scheme for dynamic access control in a hierarchy. *IEICE Transactions on Fundamentals*, E82-A(3):548–550, Mar. 1999.

[12] M.S. Hwang. Extension of CHW cryptographic key assignment scheme in a hierarchy. Accepted and to appear in *IEE Proceedings Computers and Digital Techniques.*

[13] D.E. Knuth. *The Art of Computer Programming, Vol. 2 (Seminumberical Algorithm), 2'nd ed.* Addison-Wesley, Massachusetts, 1980.

[14] H.T. Liaw, S.J. Wang, and C.L. Lei. A dynamic cryptographic key assignment scheme in a tree structure. *Computers and Math. with Applic.*, 25(6):109–114, 1993.

[15] S.J. Mackinnon, P.D. Taylor, H. Meijer, and S.G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Transactions on Computers*, 34(9):797–802, Sep. 1985.

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