A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy

Wen-Guey Tzeng

Abstract—The cryptographic key assignment problem is to assign cryptographic keys to a set of partially ordered classes so that the cryptographic key of a higher class can be used to derive the cryptographic key of a lower class. In this paper, we propose a *time-bound* cryptographic key assignment scheme in which the cryptographic keys of a class are different for each time period, that is, the cryptographic key of class C_i at time t is $K_{i,t}$. Key derivation is constrained not only by the class relation, but also the time period. In our scheme, each user holds some secret parameters whose number is independent of the number of the classes in the hierarchy and the total time periods. We present two novel applications of our scheme. One is to broadcast data to authorized users in a multilevel-security way and the other is to construct a flexible cryptographic key backup system.

Index Terms—Access control, cryptographic key assignment, secure broadcasting, cryptographic key backup.

1 Introduction

The hierarchical cryptographic key assignment (called *key assignment*, hereafter) problem was first studied by Akl and Taylor in 1983 [1] and, then, by other researchers for example, in [3], [7], [9], [13]. In a computer system, information items (called *data*, hereafter) are usually classified into security classes (called *classes*, hereafter) C_i, $1 \leq i \leq m$, which are partially ordered by a binary relation "$\preceq$" such that they form a partial-order hierarchy. In the partial-order hierarchy, $C_j \prec C_i$ denotes that the security level of class C_j is lower than that of class C_i and $C_j \preceq C_i$ denotes additionally that $C_j = C_i$ is possible. The key assignment problem is to assign a cryptographic key (called *key*, hereafter) K_i to each class C_i such that we can use K_i to derive K_j if and only if $C_j \preceq C_i$. In such a system, we use key K_i to encrypt the data in class C_i. We assign each user in the system into a class according to his security clearance. A user with a higher security clearance is assigned a higher class and a user with a lower security clearance is assigned a lower class. Then, the user in class C_i is given the key K_i such that he can decrypt the data in class C_i, which are encrypted with K_i. Furthermore, a user in class C_i can decrypt the data in class C_j if and only if $C_j \preceq C_i$. That is, the user uses K_i to derive K_j and then uses K_j to decrypt the encrypted data in C_j. We assume that readers are familiar with secret-key encryption algorithms, such as DES (Data Encryption Standard) [6], [14].

A key assignment scheme can solve the access control problem of a computer system, which has arisen in an organization that has a hierarchical structure. For example, the data in a government are usually classified into four classes: "unclassified," "confidential," "secret," and "top-secret" such that only top officers can access top-secret data. The classes are arranged as "unclassified" $\preceq$ "confidential" $\preceq$ "secret" $\preceq$ "top-secret." If we encrypt the top-secret data with K_4, the secret data with K_3, the confidential data with K_2, and the unclassified data with K_1 by a key assignment scheme. We can use K_i to derive K_j for $j \leq i$. If we give a top officer the key K_4, he can gain information in all classes. If we give an employee the key K_2, he can only read the information in classes "unclassified" and "confidential" only. By this, we can control the information flow by cryptographic keys. Cryptographic access control is useful not only in a computer system, but also in a communication system, in which protection against eavesdropping is important. We can also use the key assignment scheme to partially solve the multilevel security problem, which is concerned with protection of classified data and their aggregation, dissemination, update control, etc. [6].

Most researchers on the topic have concentrated on proposing schemes that either have better performance or allow inserting and deleting classes in the hierarchy. For example, MacKinnon et al. [9] proposed an optimal canonical key generating algorithm to assign keys to the scheme of Akl and Taylor when the number of classes in the hierarchy is large. Sandhu [13] proposed a key assignment scheme for a tree hierarchy, which is a special case of a partial-order hierarchy. The scheme is based on symmetric cryptosystems, as opposed to asymmetric cryptosystems, which most previously proposed key assignment schemes use. The scheme also allows us to add new classes to the tree without affecting keys for existing classes. Harn and Yin [7] proposed a key assignment scheme such that new classes can be added into the hierarchy also. Their key assignment procedure assigns keys to classes in a bottom-up way, while most previously proposed schemes [1], [3], [13] use the top-down assignment. Chang et al. [3] proposed a key assignment scheme using "Newton's interpolation" method as its mathematical basis.

• *The author is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050. E-mail: tzeng@cis.nctu.edu.tw.*

Manuscript received 18 July 1997; revised 5 May 2000; accepted 27 June 2000; posted to Digital Library 12 June 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 105396.

In this paper, we consider the situation that a *user* may be in a class for only a period of time. We propose a *time-bound* key assignment scheme in which each class C_i has many class keys $K_{i,t}$, where $K_{i,t}$ is the class key of C_i at time t. The *time* is not necessarily a real time. We actually divide time into time periods, starting at 0. Consider that data that are put into class C_j at time t are encrypted using key $K_{j,t}$. A user in class C_i from time t_1 to t_2 is given information $I(i,t_1,t_2)$, $1 \le i \le m$, $t_1 \le t_2$, such that with the information $I(i,t_1,t_2)$, the user can compute the class key $K_{j,t}$ of C_j at time t if and only if $C_j \preceq C_i$ and $t_1 \le t \le t_2$. Thus, the user can decrypt the data stored in class C_j at time t. This scheme can be considered as a generalization of previous ones in which $I(i,t_1,t_2)$ is simply K_i. Like the previous observation, a straightforward implementation of the scheme requires users to keep a large number of keys, which is not practical. We consider the time-versus-space trade-off. In our scheme, we require that the size of $I(i,t_1,t_2)$ be independent of the number of total classes in the hierarchy and the time period from t_1 and t_2.

We propose two novel and interesting applications for our time-bound key assignment scheme. The first application is concerned with broadcasting data to authorized users in a multilevel security way with optimal bandwidth. In this broadcasting system, a user can only obtain the information that he is authorized to get. On the other hand, an unauthorized user cannot get any information by listening to the broadcasting. This is useful in controlling access to a web-based electronic paper to subscribers. The second application is concerned with the file encryption problem in the computer system of an organization. We propose a cryptographic key backup system that allows employees to encrypt their files, but does not allow them to hold encrypted files as hostages. Our system is very flexible since it takes into account that a person may be employed for a period of time only. We describe these two applications in Section 5.

The rest of the paper is organized as follows: In the next section, we present the necessary mathematical backgrounds of our scheme. In Section 3, we describe our scheme in detail. In Section 4, we analyze the security of our scheme. In Section 5, we present two possible applications of our scheme. In Section 6, we discuss a possible solution to the time-bound cryptographic key assignment problem using previously proposed schemes, though it is not practical. Finally, we conclude the paper and present an open problem.

2 Mathematical Backgrounds

Our scheme uses the following mathematical backgrounds: modular exponentiation, the Lucas function, and one-way hash functions [11], [14].

2.1 Modular Exponentiation

Let p and q be two large primes that satisfy the criteria of strong primes [11], [14], $n = pq$, and $\phi(n) = (p-1)(q-1)$, where $\phi(n)$ is called the Euler's totient function. Then,

$$Z_n = \{0, 1, \ldots, n-1\}$$

and

$$Z_n^* = \{a \mid 1 \le a \le n-1, \gcd(a, \phi(n)) = 1\}.$$

For a number $e \in Z_n^*$, there is a number $d \in Z_n^*$ so that $ed \bmod \phi(n) = 1$. By Euler's generalization of Fermat's Little Theorem, $a^{ed} \bmod n = a$ for any $a \in Z_n^*$. The computational aspects of modular exponentiation over Z_n^* is as follows: The terms "feasible" or "efficient" both mean that "computable in polynomial time in the input size."

1. *Factorization* problem: For $n = pq$, the product of two large primes, we cannot compute p and q efficiently.
2. If $\phi(n)$ is known, for any given $e \in Z_n^*$, we can efficiently compute $d \in Z_n^*$ so that $ed \equiv 1 \ p \bmod \phi(n)$ using the extended Euclid's algorithm.
3. If only $e \in Z_n^*$ and n are known, we cannot compute $d \in Z_n^*$ with $ed \equiv 1 \ p \bmod \phi(n)$ efficiently.
4. *Discrete logarithm* problem: We cannot compute a solution of x for the equation $y = g^x \bmod n$, where y, g and n are given. The problem remains difficult even if n is a prime.
5. The eth *root* problem: Given e, n, and $y = x^e \bmod n$, we cannot compute x efficiently. Note that, if we know the factors p and q of n, we can solve x_1 for $y \equiv x^e \bmod p$ and x_2 for $y \equiv x^e \bmod q$ efficiently and use the Chinese Remainder Theorem to combine x_1 and x_2 as $x = x_1 q(q^{-1} \bmod p) + x_2 p(p^{-1} \bmod q) \bmod n$, which is a solution for $y = x^e \bmod n$, where $ed \bmod \phi(n) = 1$.
6. Given x, g, and n, we can compute $y = g^x \bmod n$ efficiently by the repeated square-and-multiply algorithm.
7. *Common modulus attack:* Given $y_1 = x^r \bmod n$ and $y_2 = x^s \bmod n$, if $\gcd(r,s) = 1$, we can compute $x \bmod n$ efficiently. We use the extended Euclidean algorithm to find integers a and b with $ar + bs = 1$ first. Then, we compute $y_1^a \cdot y_2^b \equiv x^{ar+bs} \equiv x \bmod n$. We remark that $\gcd(x,n)$ need not be 1 for the attack to succeed.

2.2 Lucas Function

Let p and q be two large primes and $n = pq$. Typically, p and q are of length about 512 bits and their lengths differ in a few bits. Let m be an integer. The Lucas function is defined as

$$V_i(m) = m \cdot V_{i-1}(m) - V_{i-2}(m) \bmod n$$

for $i \ge 2$ with the initial condition $V_0(m) = 2$ and $V_1(m) = m$. The sequence $\langle V_i(m) \rangle_{i=0}^{\infty}$ is called the Lucas sequence over m. For example, let $n = 15$ and $m = 4$ or 5. We have the Lucas sequences as shown in Table 1. Since the Lucas function is a second-order recurrence relation, we can compute each term $V_a(m), a \ge 0$ directly by

$$V_a(m) = \alpha^a + \alpha^{-a} \bmod n,$$

where α is a root of the characteristic polynomial

$$f(x) = x^2 - mx + 1 \bmod n.$$

The Lucas function has the following properties:

1. For any positive integers a, b, and m, we have

$$V_a(V_b(m)) = V_b(V_a(m)) = V_{ab}(m), \qquad (1)$$

and

TABLE 1
The Lucas Sequences with $n = 15$ and $m = 4$ or 5

$V_0(4)$	$V_1(4)$	$V_2(4)$	$V_3(4)$	$V_4(4)$	$V_5(4)$	$V_6(4)$	$V_7(4)$	$\cdots$
2	4	14	7	14	4	2	4	$\cdots$
$V_0(5)$	$V_1(5)$	$V_2(5)$	$V_3(5)$	$V_4(5)$	$V_5(5)$	$V_6(5)$	$V_7(5)$	$\cdots$
2	5	8	5	17	5	8	5	$\cdots$

$$V_{a+b}(m) = V_a(m)V_b(m) - V_{a-b}(m) \bmod n. \quad (2)$$

2. Let $e \geq 3$ with $\gcd(e, (p^2-1)(q^2-1)) = 1$ and d with $ed \equiv 1 \bmod (p^2-1)(q^2-1)$. For any m, we have $V_e(V_d(m)) = 1$.
3. We can compute $V_a(m)$ efficiently from given a, n and m. By (2), we have the relations

$$V_{2r}(m) = V_r^2(m) - 2 \bmod n, \text{and}$$
$$V_{2r+1}(m) = V_{r+1}(m)V_r(m) - m \bmod n.$$

We apply these two equations recursively to compute $V_a(m)$. It takes $2\lfloor \log_2 a \rfloor + 2$ modular multiplications [17].
4. We cannot compute m efficiently from given a, n, and $V_a(m)$.
5. Let T be the period of the sequence $\langle V_0(m), V_1(m), \ldots \rangle$. Then we have,

$$V_{-a}(m) = (-1)^a V_a(m) \text{ and}$$
$$V_a(m) = V_{T-a}(m).$$

2.3 One-Way Hash Function

A hash function takes as input a binary string of arbitrary length and outputs a binary string of fixed length. A hash function h is one-way if it is computationally feasible to compute $h(x)$, but computationally infeasible to find an x that satisfies $h(x) = y$, where y is given. The hash result $h(x)$ is generally considered as a good cryptographic key (possibly with truncation, folding, etc., to satisfy the key length of a cryptosystem). One-way hash functions are a standard mechanism for safeguarding valuable information and generating pseudorandom numbers.

Typical one-way functions are $h_{g,p}(x) = g^x \bmod p$, $h_n(x) = x^2 \bmod n$ and cryptographically strong hash functions, such as SHA-1, MD5, etc. [11], [14].

3 Our Scheme

There is a central authority (CA) that generates and assigns keys to classes in the hierarchy. All secret parameters are processed and managed by CA. We assume that the partial-order hierarchy has m classes C_i, $1 \leq i \leq m$, which are partially ordered by the binary relation "$\preceq$". We also assume that the maximum number of time periods is $z+1$. For simplicity, we let z be an integer, i.e., the system starts at time 0 and ends at time z. We remark that this constraint on the maximum number of time periods should not be considered as limitation of the system. For example, if each time period represents a week, $z = 5,200$ denotes roughly 100 years. Let H be a one-way hash function, which is publicly known to all. The output of H is 56-bit long if the encryption algorithm is DES, 112-bit long if the encryption algorithm is the triple DES, and 128-bit long if the encryption algorithm is AES (Advanced Encryption Standard), which is going to be announced in August, 2000.

Our scheme works as follows:

1. **Initial computation.** CA chooses four large strong primes p_1, p_2, q_1, and q_2 and let $n_1 = p_1 \cdot q_1$ and $n_2 = p_2 \cdot q_2$. CA also chooses two random numbers a and b, where $1 < a < n_1$, and $1 < b < n_2$. Numbers p_1, q_1, n_1 and a are used for the partial-order hierarchy and p_2, q_2, n_2, and b are used for the time period, i.e., the Lucas function.

   For the partial-order hierarchy, CA randomly chooses $e_1, e_2, \ldots, e_m$, each relatively prime to $\phi(n)$, and computes $d_1, d_2, \ldots, d_m$ such that

$$e_i d_i \bmod \phi(n_1) = 1$$

   for $1 \leq i \leq m$. CA then computes

$$K_i = a^{\Pi_{C_k \preceq C_i} d_k} \bmod n_1$$
$$= (a^{d_1 d_2 \cdots d_m})^{\Pi_{C_l \not\preceq C_i} e_l} \bmod n_1.$$

   For the time period, CA randomly chooses f_1 and f_2 and computes

$$w_t = V_{f_1^{z-t} f_2^t}(b),$$

   where $0 \leq t \leq z$. Note that the modulus is n_2.
2. **Key assignment.** CA selects g_1 and g_2, relatively prime to $\phi(n_1)$, and computes h_1 and h_2 such that $g_1 h_1 \equiv g_2 h_2 \equiv 1 \ p\text{mod}\phi(n_1)$. The key associated with class C_i at time t is

$$K_{i,t} = H(K_i^{h_1^t h_2^{z-t}} \bmod n_1, w_t).$$

   A key assignment for a six-class partial-order hierarchy with $z = 5$ is illustrated in Fig. 1. There are six classes C_i, $1 \leq i \leq 6$, and six time periods $0, 1, \ldots, 5$. We have

$$K_1 = a^{d_1 d_2 d_3 d_4 d_5 d_6} \bmod n_1, K_2 = a^{d_2 d_4 d_5 d_6} \bmod n_1,$$

$$K_3 = a^{d_3} \bmod n_1, \ K_4 = a^{d_4} \bmod n_1,$$
$$K_5 = a^{d_5} \bmod n_1,$$

   and $K_6 = a^{d_6} \bmod n_1$. We also have $w_0 = V_{f_1^5}(b)$,

$$w_1 = V_{f_1^4 f_2}(b), \ w_2 = V_{f_1^3 f_2^2}(b), \ w_3 = V_{f_1^2 f_2^3}(b),$$

   $w_4 = V_{f_1^1 f_2^4}(b)$ and $w_5 = V_{f_2^5}(b)$. The keys

$$K_{i,t}, \ 1 \leq i \leq 6, \ 0 \leq t \leq 5,$$

   are computed by $H(K_i^{h_1^t h_2^{z-t}} \bmod n_1, w_t)$ accordingly.
3. **Public parameters.** The public parameters are put into a public directory so that all users can access them. Since the public parameters are not changed during the life time of the system, a user can keep them in his local storage and use them, together with his $I(i, t_1, t_2)$, to compute the authorized class keys. The public parameters are

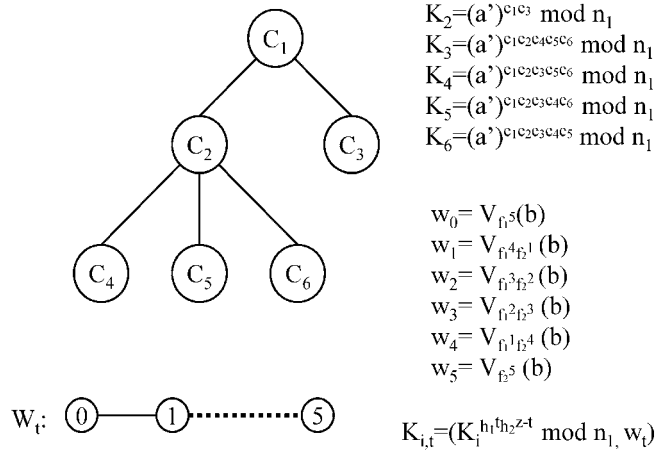

Fig. 1. An example of our key assignment scheme.

$$g_1, g_2, f_1, f_2, e_1, e_2, \ldots, e_m, n_1, n_2.$$

The other parameters are kept by CA securely so that no users can access them.

4. **Information** $I(i, t_1, t_2)$. When a user is assigned to class C_i and allowed to obtain his class keys from time t_1 to t_2, he is given the information

$$I(i, t_1, t_2) = (K_i^{h_1^{t_2} h_2^{z-t_1}} \bmod n_1, V_{f_1^{z-t_2} f_2^{t_1}}(b)).$$

Since $K_i^{h_1^{t_2} h_2^{z-t_1}} \bmod n_1$ is of size (the number of bits) equal to that of n_1 and $V_{f_1^{z-t_2} f_2^{t_1}}(b)$ is of size equal to that of n_2, the size of $I(i, t_1, t_2)$ is equal to the total size of n_1 and n_2, which is independent of the number of classes in the hierarchy and the time period from t_1 to t_2.

5. **Key derivation.** From the given information $I(i, t_1, t_2) = (x, y)$ and the public parameters, a user can compute $K_{j,t}$ if $C_j \preceq C_i$ and $t_1 \le t \le t_2$. The user first computes

$$\begin{aligned} & x^{g_1^{t_2-t} g_2^{t-t_1} \Pi_{C_k \preceq C_i, C_k \not\preceq C_j} e_k} \bmod n_1 \\ &= a^{g_1^{t_2-t} h_1^{t_2} g_2^{t-t_1} h_2^{z-t_1} \Pi_{C_l \preceq C_i} d_l \Pi_{C_k \preceq C_i, C_k \not\preceq C_j} e_k} \bmod n_1 \\ &= a^{g_1^{-t} g_2^t h_2^z \Pi_{C_l \preceq C_j} d_l \Pi_{C_k \preceq C_i, C_k \not\preceq C_j} e_k d_k} \bmod n_1 \\ &= a^{h_1^t h_2^{z-t} \Pi_{C_l \preceq C_j} d_l} \bmod n_1 \\ &= K_j^{h_1^t h_2^{z-t}} \bmod n_1 \end{aligned}$$

and

$$\begin{aligned} & V_{f_1^{t_2-t} f_2^{t-t_1}}(y) \\ &= V_{f_1^{t_2-t} f_2^{t-t_1}}(V_{f_1^{z-t_2} f_2^{t_1}}(b)) \\ &= V_{f_1^{t_2-t} f_2^{t-t_1} f_1^{z-t_2} f_2^{t_1}}(b) \\ &= V_{f_1^{z-t} f_2^t}(b) \\ &= w_t. \end{aligned}$$

Note that computing $K_j^{h_1^t h_2^{z-t}} \bmod n_1$ and w_t from $I(i, t_1, t_2)$ and the public parameters is computationally feasible. Therefore, the associated key $H(K_j^{h_1^t h_2^{z-t}} \bmod n_1, w_t)$ of C_j at time t can be computed efficiently.

3.1 Performance

Given the public parameters and $I(i, t_1, t_2)$, to derive $K_{j,t}$, we need compute

$$x^{g_1^{t_2-t} g_2^{t-t_1} \Pi_{C_k \preceq C_i, C_k \not\preceq C_j} e_k} \bmod n_1$$

and

$$V_{f_1^{t_2-t} f_2^{t-t_1}}(y).$$

We need do $t_2 - t_1 + r$ modular exponentiation operations and $t_2 - t_1$ Lucas operations, where r is the number of classes that satisfy $C_k \preceq C_i$ and $C_k \not\preceq C_j$, and each Lucas operation is roughly equivalent to a modular exponentiation operation [17]. Therefore, it takes $t_2 - t_1 + r$ modular exponentiation operations and a hash operation of H totally. In the view of complexity analysis, this is computable.

The number of public parameters is $m + 6$, which is linear in the number of classes in the hierarchy.

4 Security Analysis

We now discuss the security of our scheme. There are three security points to be addressed. The first, is the security of K_i, the second, is the security of w_t, and the third, is the security of $K_{i,t}$.

4.1 Security of K_i

For the security of K_i, there are two possible attacks. The first attack, is that a user in class C_j tries to derive K_i of class C_i, $C_j < C_i$, from K_j. Since $K_j = (K_i)^{e'}$, where e' is a product of some e_ls, $1 \le l \le m$, computing K_i efficiently is equivalent to breaking the RSA cryptosystem [12], which is now computationally infeasible [11], [14].

The second attack, is that two users collude to compute the unauthorized K_i of class C_i. Let two users be in classes C_{j_1} and C_{j_2}, respectively, where $C_{j_1} \not\succeq C_{j_2}$ and vice versa. There are three cases for C_i. In the following: We let e' and e'' be products of some e_ls, $1 \le l \le m$.

1. C_i is the least common ancestor of C_{j_1} and C_{j_2} in the partial-order hierarchy. It can be seen that $K_{j_1} = K_i^{e_i e'} \bmod n_1$ and $K_{j_2} = K_i^{e_i e''} \bmod n_1$. If $\gcd(e', e'') = 1$, one can compute $K_i^{e_i} \bmod n_1$ by the "common modulus" attack. However, even if $K_i^{e_i}$ is known, to compute K_i is as hard as to break the RSA cryptosystem. We can check this case by the example in Fig. 1 with $j_1 = 4, j_2 = 5$ and $i = 2$.
2. C_i is not in the partial-order subhierarchy rooted at C_s, where C_s is the least common ancestor of C_{j_1} and C_{j_2}. From the above case, we can see that $K_s^{e_s e'} \bmod n_1 = (a')^{e_i e''} \bmod n_1$ may be computed, where $a' = a^{d_1 d_2 \cdots d_m}$. Since $K_i = (a')^{\Pi_{C_l \preceq C_i} e_l} \bmod n_1$ does not have e_i in the exponentiation, we see no way to compute K_i from $(a')^{e_i e''} \bmod n_1$. We can check this case by the example in Fig. 1 with $j_1 = 4, j_5 = 5, s = 2$ and $i = 3$.
3. C_i is in the partial-order subhierarchy rooted at C_s, but not in the partial-order subhierarchy rooted at C_{j_1} (and C_{j_2}). Then, $K_s^{e_s e'} \bmod n_1 = (a')^{e_i e''} \bmod n_1$

can be computed. However, $K_i = (a')^{e''}$ does not contain e_i in its exponentiation e''. It is hard to compute K_i from $K_s^{e_s e'} \bmod n_1$ in which e_i appears in the exponentiation of a'. We can check this case by the example in Fig. 1 with $j_1 = 4,\ j_5 = 5,\ s = 2$, and $i = 6$.

Therefore, collusion of two (or more) users does not help in computing unauthorized K_i. It is even harder for an attacker to attack our scheme since K_i is raised to the power $h_1^{t_2} h_2^{z-t_1}$. With $K_i^{h_1^{t_2} h_2^{z-t_1}} \bmod n_1$, the attacker can compute

$$K_i^{h_1^{t_1} h_2^{z-t_1}} \bmod n_1,\ K_i^{h_1^{t_1+1} h_2^{z-t_1-1}} \bmod n_1, \ldots, K_i^{h_1^{t_2} h_2^{z-t_2}} \bmod n_1$$

which are useful for computing $K_{j,t_1}, \ldots, K_{j,t_2}$, $C_j \preceq C_i$, and no others.

4.2 Security of w_t

For the security of w_t, there are also two possible attacks. The first attack, is that, given information $I(i, t_1, t_2)$, the user wants to compute $w_{t'}$ for $t' < t_1$ or $t' > t_2$. From the given

$$x = V_{f_1^{z-t_2} f_2^{t_1}}(b),$$

if one wants to compute $w_{t'}$, one need find $f_1^{-1} \bmod (p^2 - 1)(q^2 - 1)$ or $f_2^{-1} \bmod (p^2 - 1)(q^2 - 1)$ in order to compute

$$w_{t'} = V_{f_1^{t_2 - t'} (f_2^{-1})^{t_1 - t'}}(x)$$

for $t' < t_1$ or

$$w_{t'} = V_{(f_1^{-1})^{t' - t_2} f_2^{t' - t_1}}(x)$$

for $t' > t_2$, which is computationally infeasible.

The second attack is that two users, given $I(j_1, t_1, t_2)$ and $I(j_2, s_1, s_2)$, respectively, collude to compute $w_{t'}$ for $t' < t_1$ or $t' > s_2$, that is, to compute $w_{t'}$ from $V_{f_1^{z-t_2} f_2^{t_1}}(b)$ and $V_{f_1^{z-s_2} f_2^{s_1}}(b)$, where $t_1 \le t_2 \le s_1 \le s_2$. The Lucas function has been studied extensively and the Lucas-function-based cryptosystems are considered as secure as the RSA cryptosystem. But, no RSA-like common modulus attack has been found in the context of the Lucas function. Collusion of two users described in the previous subsection does not seem applicable in computing $w_{t'}$. Therefore, $w_{t'}$ cannot be computed efficiently by collusion of two (or more) users.

4.3 Security of $K_{i,t}$

The third one is easiest to discuss. By general belief (definition) on the security of one-way hash functions, even if some $K_{i,t}$ leaks out, the value K_i and w_t cannot be computed within a reasonable amount of time. Therefore, other class keys are safe.

5 Applications

In this section, we propose two novel applications of our time-bound key assignment scheme. The first application is to broadcast data to authorized users so that an authorized user can only get the information designated to him. The broadcasting uses the optimal bandwidth. It is useful in designing a secure and efficient electronic subscription system. The second application is to construct a cryptographic key backup system. Each user who uses the system can encrypt his files. However, he cannot hold the encrypted files as hostages and in case of emergency, a higher class officer can authorize another employee to open the encrypted files no matter whether the owner of the files is present or not.

5.1 Secure Broadcasting

We consider an application of securely transmitting information to multiple users through a broadcast channel, which could be a local area network, Internet, a wireless cellular Personal Communication System, etc. Broadcasting data can save quite a lot of bandwidth over the point-to-point transmission. However, broadcast data are not secure since anyone on the broadcast channel can eavesdrop to get the data. To guard against unauthorized eavesdroppers, data can be encrypted and then broadcast so that only authorized users with proper keys can decrypt the data to obtain useful information [1], [4]. It can be easily conceived that in some applications, a more *flexible* secure broadcasting system is required. We illustrate this by an example of an electronic newspaper subscription system.

A newspaper company provides newspapers in electronic form (called newspaper hereafter) to its subscribers through a broadcast channel. We consider a situation that some subscribers are only interested in some part of the newspaper and want to subscribe to that part so the subscription fee is less. Furthermore, from the view of the company, a subscriber usually subscribes to the newspaper for only a period of time. After the expiration date, the subscriber should not be able to decrypt the broadcast data with his old key. Assume that there are m possible partial subscriptions. Let S_i be the data for the ith partial subscription and U_i be the set of users who subscribe S_i. Note that $S_i \cap S_j$ may not be empty and, in most cases, it is not. A straightforward solution is to encrypt each S_i with a key K_i and distributes the key K_i to all subscribers in U_i. If the subscription of a subscriber in U_i expires, the newspaper company changes the encryption key of S_i to K_i' at the expiration date and distributes the new key K_i' to the remained subscribers in U_i. It can be seen that the company not only uses more bandwidth in broadcasting, but also has a complicated key distribution problem.

By our time-bound key assignment scheme, we propose a subscription system for the newspaper company as follows. Let the newspaper of each day be partitioned into m parts, P_i, $1 \le i \le m$, which form a partial-order hierarchy. The ith partial subscription data S_i consists of the data assigned to P_i and those assigned to its lower classes P_j, $P_j \preceq P_i$. A subscriber who subscribes to P_i can get all the data of the classes in the subhierarchy rooted at P_i. The system works as follows:

1. For each date t, the company encrypts data in P_i with key $K_{i,t}$, $1 \le i \le m$, and then broadcasts them. Note that only the data in P_i (not all the data in S_i) are encrypted with $K_{i,t}$.
2. When a subscriber subscribes the ith part S_i for a period of dates from t_1 to t_2, he is given the information $I(i, t_1, t_2)$.
3. When the subscriber receives the broadcast data $\langle j, t, D \rangle$ and $P_j \preceq P_i, t_1 \le t \le t_2$, he uses $I(i, t_1, t_2)$ to

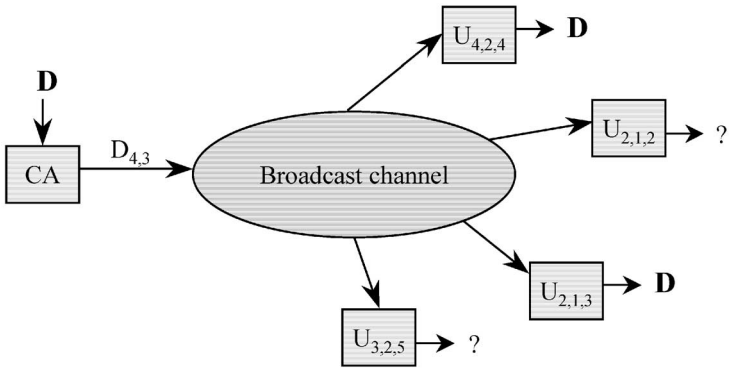

Fig. 2. A secure electronic subscription system.

derive the class key $K_{j,t}$ at time t to decrypt D, which he is authorized to get.

We illustrate our proposed electronic subscription (secure broadcasting) system in Fig. 2, in which the hierarchy in Fig. 1 is used. In the example, U_{i,t_1,t_2} denotes that the user who is in class C_i from time t_1 to t_2. CA encrypts data D with key $K_{4,3}$ as $D_{4,3}$. Therefore, $U_{4,2,4}$ and $U_{2,1,3}$ can decrypt $D_{4,3}$ to obtain the original data D, while $U_{2,1,2}$ and $U_{3,2,5}$ cannot.

Our proposed electronic subscription system has the following merits:

1. The company broadcasts data with optimal bandwidth. Since each P_i is broadcast once only, the total broadcast data are about the size of the newspaper.
2. Each subscriber holds only one key information and the size of the key information is independent of the number of total partial subscriptions and the subscription time period.
3. Subscription is time-bound and the company is exempted from the complicated key distribution problem in case of expiration of some subscriptions.
4. The company can sell its backlog of newspaper easily. The company stores the old broadcast data in encrypted form in the database. When a user wants to use the backlog newspapers of ith part S_i from dates t_1 to t_2, he is simply given $I(i, t_1, t_2)$ and allowed to access the database. With $I(i, t_1, t_2)$, the user can use the information in S_i from dates t_1 to t_2.

5.2 Cryptographic Key Backup

More and more people use cryptographic algorithms to encrypt their files in a computer system. Consider that in the computer system of a company, its employees use cryptographic algorithms to encrypt their files. Although it provides privacy (confidentiality) for the data, it also raises some concerns [10]:

1. An employee may lose his key. Therefore, the encrypted data of the employee can not be decrypted any more, even by the employee himself, and the important information of the data is lost.
2. It is possible that when an employee is not available at some time, but his encrypted file is urgently needed. A higher manager should be able to authorize another employee to decrypt the file of the employee in case of an emergency.
3. It is possible that an employee holds the encrypted file as hostages in case of discontent or layoff.

There have been some cryptographic key backup systems proposed to solve the problem [10], [16]. We use our scheme to provide another solution as follows:

1. All employees in a company are arranged into classes by their positions and duties. These classes form a partial-order hierarchy. The manager in a higher class has the right to authorize another one to access the data held by an employee in a lower class.
2. When an employee joins (or renews his position, for example, once per year) in the company, he is given information $I(i, t_1, t_2)$, where C_i is the class for his position, t_1 is the time of the earliest encrypted files which he is authorized to decrypt and t_2 is the next renewal of his position or duty.
3. Any employee, in class C_i, encrypting his file with key K at time t must attach a tag (escrowed field), such as $E(K_{j,t}, K)$, to the file, where E is the encryption function of a secret-key cryptosystem, such as DES. This can be enforced by the computer system.
4. When a higher manager authorizes another employee to access the file (encrypted at time t) of an employee in class C_j, the manager uses his $I(i, t_1, t_2)$ to derive the key $K_{j,t}$ and uses $K_{j,t}$ to decrypt the file tag $E(K_{j,t}, K)$ to get the encryption key K of the file. The manager gives K to the authorized employee to decrypt the encrypted file. Therefore, no employee can hold the encrypted data as hostages since some higher managers can decrypt the data.

6 Discussion

The time-bound key assignment problem can be solved by previously proposed schemes. The method is as follows: Let $\mathcal{C}$ be the set of classes and $\mathcal{Z}$ be the set of all nonempty intervals $[t_1...t_2]$ in $[1...z]$. There are $(z^2 + z)/2$ nonempty intervals in $[1...z]$. $\mathcal{Z}$ is also partially ordered by the subset relation. The Cartesian product ordering, $\mathcal{C} \times \mathcal{Z}$, is also a partially ordered set of levels. Therefore, we can apply the previously proposed schemes for partial-order hierarchies to $\mathcal{C} \times \mathcal{Z}$. When a user in class C_i from time t_1 to t_2, he is

given the key for the class $\langle C_i \times [t_1...t_2] \rangle$ of $\mathcal{C} \times \mathcal{Z}$. Thus, he can compute the cryptographic keys $K_{\langle C_j \times [t...t] \rangle}$ for $C_j \preceq C_i$ and $t_1 \leq t \leq t_2$. Note that the data in class C_j at time t is encrypted with the key $K_{\langle C_j \times [t...t] \rangle}$.

By the above method the time-bound key assignment problem is solved in principle. However, it is much less efficient since the number of the public parameters is $O(m \cdot z^2)$, which may be too large if z is large.

7 Conclusion

We have presented a time-bound cryptographic key assignment scheme for a set of classes that are partially ordered by a binary relation $\preceq$. The most important feature of our scheme is that each class C_i has many time-dependent keys $K_{i,t}$, which each corresponds to a time period t. Key derivation is constrained by both "$\preceq$" and the time t. From the given information $I(i, t_1, t_2)$, one can compute the keys $K_{j,t}$ for $C_j \preceq C_i$ and $t_1 \leq t \leq t_2$ only. By this distinct feature, we have presented two interesting and novel applications of our key assignment scheme. One is to broadcast data to authorized users in a multilevel security way with optimal bandwidth. The other is to construct a cryptographic key backup system that makes an employee able to encrypt his files in a computer system, while not able to hold the encrypted files as hostages.

Like all previously proposed schemes about the key assignment problem for a partial-order hierarchy, the number of public parameters for key derivation is dependent on the total number of classes in the hierarchy. It is interesting to design a key assignment scheme for a partial-order hierarchy such that the number of public parameters is independent of the total number of classes in the hierarchy.

Acknowledgments

The author would like to thank an anonymous referee for suggesting the discussion in Section 6. This research was supported in part by the National Science Council of Taiwan, ROC, under grant NSC-86-2213-E-009-004.

References

[1] S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," *ACM Trans. Computer Systems,* vol. 1, no. 3, pp. 239-248, 1983.
[2] D. Bleichenbacher, W. Bosma, and A.K. Lenstra, "Some Remarks on Lucas-Based Cryptosystems," *Proc. Advances in Cryptology-Crypto 95.* Springer-Verlag, pp. 386-396, 1995.
[3] C.C. Chang, R.J. Hwang, and T.C. Wu, "Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," *Information Systems,* vol. 17, no. 3, pp. 243-247, 1992.
[4] G. Chiou and W. Chen, "Secure Broadcasting Using the Secure Lock," *Trans. Software Eng.,* vol. 15, no. 8, pp. 929-934, 1989.
[5] J.M. DeLaurentis, "A Further Weakness in the Common Modulus Protocol for the RSA Cryptosystem," *Cryptologia,* vol. 8, no. 3, pp. 253-259, 1984.
[6] D.E. Denning, *Cryptography and Data Security.* Reading, Mass.: Addison-Wesley, 1982.
[7] 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.
[8] H. Lehmer, "An Extended Theory of Lucas Functions," *Ann. Math.,* vol. 31, pp. 419-448, 1930.
[9] 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," *Trans. Computers,* vol. 34, no. 9, pp. 797-802, 1985.
[10] D.P. Maher, "Crypto Backup and Key Escrow," *Comm. ACM,* vol. 39, no. 3, pp. 48-53, 1996.
[11] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography.* Boca Raton: CRC Press, 1996.
[12] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM,* vol. 21, no. 2, pp. 120-126 1978.
[13] R.S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," *Information Processing Letters,* no. 27, pp. 95-98, 1988.
[14] B. Schneier, *Applied Cryptography: Protocol, Algorithms, and Source Code in C,* second ed., New York: John Wiley and Sons, 1996.
[15] P.J. Smith and M.J. Lennon, "LUC: A New Public Key System," *Proc. IFIP TC11 Ninth Int'l Conf. Information Security-SEC 93,* pp. 103-117, 1993.
[16] S.T. Walker, S.B. Lipner, C.M. Ellison, and D.M. Balenson, "Commercial Key Recovery," *Comm. ACM,* vol. 39, no. 3, pp. 41-47, 1996.
[17] S.M. Yen and C.S. Laih, "Fast Algorithms for LUC Digital Signature Computation," *Proc. IEEE Computers and Digital Techniques,* vol. 142, no. 2, pp. 165-169, 1995.

Wen-Guey Tzeng received the BS degree in computer science and information engineering from National Taiwan University, in 1985 and the MS and PhD degrees in computer science from the State University of New York at Stony Brook in 1987 and 1991, respectively. He joined the Department of Computer and Information Science, National Chiao Tung University, Taiwan in 1991. Dr. Tzeng's current research interests include information security, cryptology, and network security.