An access control scheme based on Chinese remainder theorem and time stamp concept *

Min-Shiang Hwang †‡
Email: m24%tl9000@tlrouter.motctl.gov.tw

Wen-Guey Tzeng†
Email: tzeng@weber.cis.nctu.edu.tw

Wei-Pang Yang †
Fax: 886-35-721490

Department of Computer and Information Science †
National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.

Directorate General of Telecommunication Laboratories ‡
Ministry of Transportation and Communications
Chung-Li, Taiwan 320, R.O.C.

October 14, 2004

Abstract

In this paper, we propose a new dynamic access control method for the computer system with frequently inserted, deleted and updated users/files. Our method, based on the concepts of the access control matrix, key-lock-pair, time stamp and Chinese remainder theorem, associates each user with a user key and a user lock and each file with a file key and a file lock. Our method can achieve the following four goals. First, by a simple modulo operation on the keys and locks of the user and the file, we can reveal the access right of a user to a file. Second, when a user/file is added to the computer system, we only assign a key and a lock to the user/file without affecting the keys and locks of the other users/files in the system. Third, when a user/file is deleted from the computer system, we simply erase the entry of the user/file in the computer system. Fourth, when the access right of a user to a file is updated, we merely modify the key and lock of the user or the file without affecting the keys and locks of the other users/files in the system. The main contribution of our method is that the action of inserting, deleting a user/file, or updating the access right of a user to a file can be done by modifying only one key and one lock, which can't be achieved simultaneously before.

KeyWords: Access control matrix, Chinese remainder theorem, Dynamic, Key-lock-pair, Time stamp.

*This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no. NSC83-0408-E-009-039.

†

‡

†

†

†

‡

1 Introduction

In today's computer systems, how to prevent an unauthorized user (process, utility program and etc) from altering, destroying or disclosing a file (disk, magnetic tape, printer, or memory segment) is an important issue in the field of computer security. The general concept of such prevention is called access control which in a computer system, is to verify the access right of the user to the file when a user requests the access of a file.

The most straight forward implementation of access control for a computer is to use a matrix [1], called the access control matrix, in which the entry of the ith row and jth column records the access right of the ith user over the jth file. When a user makes a request of accessing a file, the access control system grants the request only if the corresponding entry in the access control matrix records such a privilege. A drawback of the access control matrix method is that when the access control matrix is sparse and stored directly, the efficiency of space usage is quite low. Therefore quite a few methods [2, 3, 4, 5, 6, 7, 8, 9] are proposed to resolve this problem.

The capability-list and access-list systems are among the earliest [3]. However, they usually trade time for space, for example, to check the access right of a user to a file in the access-list system need go through the entire list in the worst case, which is quite time consuming. Therefore, many approaches have been proposed for access control of computer systems in the literatures [9, 10, 11]. Among them, the "key-lock" concept is plausible.

In a key-lock access control system, each user possesses a key and each file has a lock. When a user requests the access of a file, the key-lock access control system verifies the access right of the user to the file by making some operation on the user's key and the file's lock. Thus the important issues in key-lock access control methods include the effectiveness of maintaining keys and locks.

In the following we review some previous key-lock access control methods. The single-key-lock method of Wu and Hwang [9] uses an $m \times m$ key matrix and n m-tuple lock vectors. The method is time-consuming not only in the retrieval of access right but also in maintaining keys and locks while inserting and deleting a user/file and updating the access right of a user to a file. Furthermore, the storage space required by the method is $O(m^2 + mn)$ for m users and n files, which is more than that of the access control matrix method. The key-lock-pair method of Chang [4] is based on Chinese remainder theorem. The method cannot delete a user/file without overhauling all keys and locks. The key-lock-pair method of Chang and Jiang [5] is based on binary operations. Again, the method also need recompute all the keys in the system when inserting a user/file to the system. The two-key-lock-pair method of Chang and Jan [12] is also based on Chinese remainder theorem. While deleting a user/file, the method has to re-compute almost all keys. The single-key-lock method of Laih and etc [8] is based on the idea of Newton's interpolating polynomial. The method is basically the same as the access control matrix except that the access right vectors of files are replaced by lock vectors. Strictly speaking, when inserting a user/file all keys and lock vectors have to be updated in the method. The two-key-lock-pair method of Jan and etc [7] is based on the concept of p-adic numbers. The main purpose of the method is for dynamic access control, which means users/files are inserted into and deleted from a computer system frequently. However, the method need re-compute all keys and locks while deleting a user/file or updating the access right of a user to file. The storage space required by the method is the same as that of the access control matrix method. The single-key-lock method of Hwang and etc [6] is based on prime factorization. Again, the method also need re-compute all locks while inserting and deleting a user and updating the access right of a user to a file.

All the above methods cannot achieve optimality of updating one key for each operation (e.q. inserting a user/file, deleting a user/file or updating the access right of a user to a file) while using a reasonable amount of storage space. In this paper, we propose an access control method which is based on Chinese remainder theorem and the concept of time stamp. Our method achieves real dynamicism in the sense that performing one inserting, deleting or updating need only modify one key. Therefore the time complexity of each operation is dramatically reduced. Also, like those methods of Chang [4], Jan and etc [7] and Hwang and etc [6], our method uses only $O(m+n)$ storage space for m users and n files.[1] Thus, we can say that our method achieves optimality in the sense of the number of key updating for each operation and the amount of storage space usage for keys and locks. However, like most KLP and 2-KLP access control methods, our method suffers the overflow problem of keys and locks. For practical applications this problem can be dealt by some conventional solutions, such as by grouping relevant users and files such that the number of items in each level is smaller enough to make this size of keys and locks reasonable [12].

This paper is organized as follows. In the next section, we review Jan's two-key-lock-pair mechainsm [7], where we have our idea from. In Section 3, we present our new dynamic access control method. In Section4, we discuss practical issues. In the last section, we conclude the paper.

2 Review of a two-key-lock-pair method

In this section we review the two-key-lock-pair access control method of Jan and etc [7]. In a two-key-lock-pair access control system, users and files are treated symmetrically, i.e., each user possesses a key and a lock and so does each file. The two-key-lock-pair scheme of Jan and etc [7] is shown in Figure 1. There are two

[1] We ignore the overflow issue here

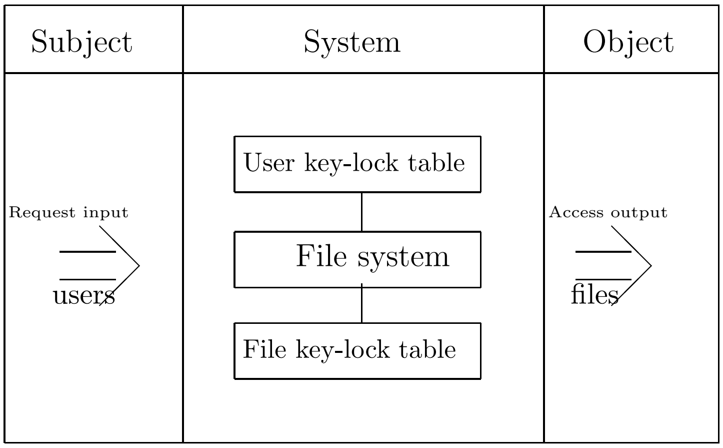

Figure 1: The Two-Key-Lock-Pair Scheme.

tables in the scheme. One is the user key-lock table and the other is the file key-lock table. The user key-lock table has three columns: one is the key column K, another is the lock column L and the other is the number column NF which denotes the number of files in the computer system when the user is inserted into the system. The file key-lock table also has three columns: one is the key column K', another is the lock column L' and the other is the number column NU which denotes the number of users in the computer system when the file is inserted into the system. Let t be the total number of distinct access rights. For example, t is equal to 5 if the access right could be "none" which is denoted by number 0, "read" denoted by 1, "write" denoted by 2, "execute" denoted by 3, "own" denoted by 4. Let r_{ij} be the number which denotes the access right of user i to file j. The key values of user i and file j are defined as

$$K_i = r_{i1} + r_{i2}t + \cdots + r_{i,NF_i}t^{NF_i-1} \text{ and} \tag{1}$$

$$K'_j = r_{1j} + r_{2j}t + \cdots + r_{NU_j,j}t^{NU_j-1}, \tag{2}$$

Table 1: An Access Control Matrix

	F_1	F_2	F_3	F_4	F_5	F_6
U_1	4	4	0	1	4	2
U_2	2	1	3	0	4	3
U_3	1	1	2	1	0	3
U_4	2	1	0	4	3	2
U_5	0	3	3	2	4	2
U_6	2	3	3	0	2	3

Table 2: (a) The User Key-Lock Table (b) The File Key-Lock Table

	K_i	L_i	NF_i
U_1	Null	5^0	0
U_2	7	5^1	2
U_3	6	5^2	2
U_4	7	5^3	3
U_5	340	5^4	4
U_6	92	5^5	4

(a)

	K'_j	L'_j	NU_j
F_1	4	5^0	1
F_2	4	5^1	1
F_3	65	5^2	3
F_4	526	5^3	4
F_5	9149	5^4	6
F_6	10967	5^5	6

(b)

respectively. The lock values of user i and file j are t^{i-1} and t^{j-1}, respectively. Then the access right r_{ij} of user i to file j can be derived by the keys and locks of user i and file j as

$$r_{ij} = \begin{cases} \lfloor K_i/L'_j \rfloor \bmod t & \text{if } i > NU_j, \\ \lfloor K'_j/L_i \rfloor \bmod t & \text{otherwise.} \end{cases} \tag{3}$$

We illustrate the above description by the following example. Let U_1, U_2, …, U_6 and F_1, F_2, …, F_6 be the six users and the six files which are added to the system in the sequence U_1, F_1, F_2, U_2, U_3, F_3, U_4, F_4, U_5, U_6, F_5, F_6. The corresponding access rights of users to files are shown in Table 1. The corresponding user and file key-lock tables are shown in Table 2. It can be seen that when user 1 is deleted from the system, all key and lock values of files have to be re-computed.

3 New dynamic access control method

In this section we propose a new dynamic two-key-lock-pair method for information protection of computer systems. Since our two-key-lock-pair access control method is based on Chinese remainder theorem, we first introduce the theorem.

Theorem 3.1 (Chinese Remainder Theorem) *[1] Let $P_1, P_2, \ldots, P_t$ be pairwise relatively prime integers and let $n = P_1 \times P_2 \times \cdots \times P_t$. Then the system of equations*

$$X = X_i \bmod P_i \textit{ for } i = 1, 2, \ldots, t$$

has a common solution X in the range $[0, n-1]$.

By Chinese remainder theorem, we can use two sets $\{L_1, L_2, \ldots, L_m\}$ and $\{L'_1, L'_2, \ldots, L'_n\}$ of primes as the lock values of users and files, respectively. Let r_{ij} be the number that denotes the access right of user i to file j. Then we can calculate two sets $\{K_1, K_2, \ldots, K_m\}$ and $\{K'_1, K'_2, \ldots, K'_n\}$ of key values for users and files respectively by the following system of equations.

$$\begin{aligned} K_i &= r_{ij} \bmod L'_j \text{ for } j = 1, 2, \ldots, NF_i, 1 \le i \le m; \quad (4) \\ K'_j &= r_{ij} \bmod L_i \text{ for } i = 1, 2, \ldots, NU_j, 1 \le j \le n, \quad (5) \end{aligned}$$

where NF_i is the number of files in the system when user i is added to the system and NU_j is the number of users in the system when file j is added to the system. After this, we can see that the access right of user i to file j can be computed by the following equation.

$$r_{ij} = \begin{cases} K_i \bmod L'_j & \text{if } NU_j < i, \\ K'_j \bmod L_i & \text{otherwise.} \end{cases} \quad (6)$$

We use the following example to explicate the above method.

Example 3.1 *Consider a file system as depicted in Table 1. The sequence of users and files being added into the system is the same as that in Section 2. By the above*

Table 3: (a) The user key-lock table. (b) The file key-lock table.

	K_i	L_i	NF_i
U_1	Null	5	0
U_2	7	6	2
U_3	1	7	2
U_4	7	11	3
U_5	255	13	4
U_6	297	17	4

(a)

	K'_j	L'_j	NU_j
F_1	4	5	1
F_2	4	6	1
F_3	135	7	3
F_4	246	11	4
F_5	784	13	6
F_6	717	17	6

(b)

method we maintain two key-lock tables for users and files as in Table 3. If we want to compute the access right r_{54} of user 5 to file 4, then we find that NU_4 is 4 in the file key-lock table. Since $NU_4 < 5$, by Equation 6 the access right of user 5 to file 4 is

$$\begin{aligned} r_{54} &= K_5 \bmod L'_4 \\ &= 255 \bmod 11 \\ &= 2. \end{aligned}$$

It is easy to see that if we want to delete a user/file, we have to update many key values in the user and file key-lock tables. In order to solve this problem, we introduce the concept of time stamp to replace NU and NF in the user and file key-lock tables.

Time stamp has a unique property of identifing the sequence of users and files added to system. We can assign a distinct time stamp number to the user/file when a user/file is added to the system. Therefore, when deleting a user/file, we simply remove the user/file entry from the user/file key-lock table and the order of users and files in the system is not changed. On the other hand, with the number (NU and NF) of users and files if deleting a user or file from the system directly, it will result in confusion of which formula to be taken in Equation 6. For example, when we delete U_3 and U_4 from Table 3 (thus 4 users remains in the system) and add a

Table 4: (a) The user key-lock table. (b) The file key-lock table. (c) The user prime number stack (SU). (d) The file prime number stack (SF).

(a)

	K_i	L_i	TS_i
U_1	Null	5	0
U_2	7	6	3
U_3	1	7	4
U_4	7	11	6
U_5	255	13	8
U_6	297	17	9

(b)

	K'_j	L'_j	TS'_j
F_1	4	5	1
F_2	4	6	2
F_3	135	7	5
F_4	246	11	7
F_5	784	13	10
F_6	717	17	11

(c)

19
23
29
31
37
⋮
⋮

(d)

19
23
29
31
37
⋮
⋮

new file (F_7) to the system with $NU_7 = 4$, by Equation 6, r_{76} will be computed with K_7 and L'_6. Obviously, it's not correct.

The data structure used in our scheme consists of one user key-lock table, one file key-lock table, one user prime number stack (SU) and one file prime number stack (SF), which are shown in Figure 2. The user (file) key-lock table has three column: key value column, lock value column and time stamp column. When a user i is added to the system, the system assigns the current time stamp number to the user and pop a prime number from the user prime number stack as the lock of the user. The key value of the user i is computed by the system of equations

$$K_i = r_{ij} \bmod L'_j, \text{ for all existent files } j,$$

where r_{ij} is the access right of the user i to the file j. When a file is added to the system, its key value, lock value and time stamp are computed similarly. We use Example 3.1 in this section to show our method. The resulting user/file key-lock tables are shown in Table 4.

In the following subsections we give algorithms, based on our two-key-lock-pair access control method, for inserting a user/file, checking access right of a user to a file, changing the access right of a user to a file, and deleting a user/file.

Key Values K_i	*Lock Values* L_i	*Time Stamp* TS_i
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

(a)

Key Values K'_j	*Lock Values* L'_j	*Time Stamp* TS'_j
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

(b)

(c) (d)

Figure 2: Data structure of (a)the user key-lock table, (b)the file key-lock table, (c)the user prime number stack (SU) and (d) the file prime number stack (SF).

Table 5: Algorithm for inserting a new user to the system.

Input:	i; /* User i */
	$r_{ij}, 1 \leq j \leq n$, where n is the number of files in the system currently;
	/* The access right of new user i to file j */
Output:	K_i; /* The user key value for user i. */
1.	$TS \longleftarrow TS + 1$;
2.	$TS_i \longleftarrow TS$;
	/* SU: the user prime number stack */
3.	$L_i \longleftarrow$ POP SU;
4.	for $j = 1, \ldots, n$ do
5.	Let $P_j = L'_j$;
	/* Computing the key values by Chinese remainder theorem */
6.	Compute $P = P_1 \times P_2 \times \cdots \times P_n$;
7.	for $j = 1, \ldots, n$ do
8.	begin
9.	Compute $G_j = P/P_j$;
10.	Find G'_j such that $G_j G'_j \bmod P_j = 1$;
11.	end;
12.	Compute key value $K_i \longleftarrow (\sum_{j=1}^{n} r_{ij} G_j G'_j) \bmod P$;

3.1 Inserting a new user/file

When inserting a new user to the system, we assign the value of the current time stamp as the time stamp of the user and pop a prime number from the user prime number stack as the lock value of the user. Then we compute the key value of the user by Equation 4 using the access rights, which are inputs, of the user to the files in the system and using the lock values of the files in the system. To add a new file to the system, the procedure is similar to adding a new user to the system except that we use the lock values of users. The algorithm for inserting a new user to the system is given in Table 5.

3.2 Checking access right

To check the access right of user i to file j, we first compare the time stamp values TS_i and TS'_j of the user and the file. If the time stamp value of the user is smaller than that of the file, i.e., user i is added to the system before file j, we use the lock

Table 6: Algorithm for checking access right.

Input:	(i, j, R_{ij}); /* User i, file j and the request access right of user i to file j */
Output:	Acceptance or rejection;
1.	if $TS'_j < TS_i$
2.	then $r_{ij} \longleftarrow K_i \bmod L'_j$
3.	else $r_{ij} \longleftarrow K'_j \bmod L_i$;
4.	if $R_{ij} \leq r_{ij}$
5.	then "accepted"
6.	else "rejected";

of the user and the key of the file to verify the access right of the user to the file. If the time stamp value of the user is greater than that of the file, i.e., user i is added to the system after file j, we use the key of the user and the lock of the file to verify the access right of the user to the file. The algorithm for checking access right of a user to a file is given in Table 6. The following example shows verification of access requests.

Example 3.2 *Let us consider Example 3.1 again. If an access request* (U_i, F_j, R_{ij}) *= (3,4,1) occurs. The system fetches the time stamps* TS_3 *and* TS'_4 *from the user and file key-lock tables. Since* $TS_3 = 4 < TS'_4 = 7$, *thus*

$$\begin{aligned} r_{34} &= K'_4 \bmod L_3 \\ &= 246 \bmod 7 \\ &= 1. \end{aligned}$$

Because $r_{34} = 1$ *is equal to the request access right 1, the access request is accepted. On the other hand, assume that another access request* $(U_i, F_j, R_{ij}) = (5, 4, 3)$ *occurs. The system fetches time stamps* TS_5 *and* TS'_4 *from user and file key-lock tables. Since* $TS_5 = 8 > TS'_4 = 7$, *thus*

$$r_{54} = K_5 \bmod L'_4$$

$$= 255 \bmod 11$$
$$= 2.$$

Because $r_{54} = 2$ is smaller than the request access right 3, the access request is rejected.

3.3 Changing access right

When the access right is changed from r_{ij} into r'_{ij}, we first compare the time stamp values TS_i and TS'_j of the user and the file. If $TS_i > TS'_j$, we re-compute the user key K_i by using all lock values L'_j of files with time stamp less than TS_i. By Chinese remainder theorem, we can compute the new K_i from the old K_i according to the following equation.

$$K_i \longleftarrow [K_i + (r'_{ij} - r_{ij})G_i G'_i] \bmod P, \tag{7}$$

where $L_1, L_2, \ldots, L_t$ are the lock values of t files with time stamp less than TS_i, $P = L_1 \times L_2 \times \cdots \times L_t$, $G_i = P/L_i$ and G'_i is the value such that $G_i \times G'_i \bmod L_i = 1$. If $TS_i < TS'_j$, the equation becomes

$$K'_j \longleftarrow [K'_j + (r'_{ij} - r_{ij})G_i G'_i] \bmod P'. \tag{8}$$

The following example shows how to change the access right of a user to a file.

Example 3.3 *Assume that the access right r_{42} is changed from 1 to 2, the system fetches the time stamps TS_4 and TS'_2 from the user and file key-lock tables. Since $TS'_2 = 2 < TS_4 = 6$, thus*

$$\begin{aligned} K_4 &= [K_4 + (r'_{42} - r_{42})G_4 G'_4] \bmod P_4 \\ &= [7 + (2-1) \times (5 \times 7) \times 5] \bmod (5 \times 6 \times 7) \\ &= (7 + 175) \bmod 210 \\ &= 182. \end{aligned}$$

Table 7: Algorithm for deleting a user from system.

Input:	i; /* User i */
	/* Delete the entry for the user from user key-lock table */
1.	PUSH L_i to the user prime number stack (SU);
2.	Delete U_i from the user key-lock table;

3.4 Deleting a user/file

By the property of Chinese remainder theorem, a user can be arbitrarily deleted from the user key-lock table. The deletion will not affect the previously discussed actions. To avoid using large prime numbers as lock values, which would result in large key values, we can recycle the released prime numbers by pushing it back to the user prime number stack. The prime number is then reserved for future users. The algorithm for deleting a user from the system is given in Table 7. To delete a file from the system, the procedure is similar to deleting a user from the system.

The following example shows how to delete a user from the computer system.

Example 3.4 *Let us consider Example 3.1 again. Assume that user* 3 *is deleted from the system. We only delete the entry of user 3 from the user key-lock table. And the prime* $L_3 = 7$ *is then reserved for future users by pushing it back to the user prime number stack.*

4 Practical issues

The most serious problems of our and other KLP access control methods is the overflow problem of key values and the complexity of arithmetic operations on overflowed numbers [10, 11].

In order to overcome these overflow problems such that the system can be practical, two approaches are proposed. One can group relevant users and files into classes so that each class consists of less files and users and the overflow problem can be controlled within some tolerable limit [12]. The other is decompose key values in B-based form. Key values K_i (for $i = 1, 2, \cdots, m$) are decomposed as follows.

$$K_i = C_{ij}B^{j-1} + C_{i(j-1)}B^{j-2} + \cdots + C_{i2}B + C_{i1} \tag{9}$$

where all coefficient (C_{ij}) of B satisfy $0 \leq C_{ik} < B$ for $k = 1, 2, \cdots, j$.

To avoid overflow computations, B has to be chosen less than the square root of 2^l and larger than lock value L_i (ie., $B \leq \sqrt{2^l}$ and $L_i < B$ for $i = 1, 2, \cdots, m$). Here we assume that the computer system is in l-bits integer wordlength.

K_i is thus represented by $(C_{ij}, C_{i(j-1)}, \cdots, C_{i2}, C_{i1})$. Using this representation of keys, keys can be stored without overflow problem.

5 Conclusions

We have proposed a new two-key-lock-pair access control method based on Chinese remainder theorem and the concept of time stamp for file protection systems. Its advantageous properties include the following.

1. When a user requests the access of a file, our scheme can reveal the access right of the user to the file easily.

2. To delete a user/file from the system, the scheme need only delete the key value of the user/file.

3. To insert a user/file, the scheme need only compute the key value of the user/file.

4. To change the access right of a user to a file, the scheme need only modify the key value of the user or the file.

5. The lock value (a prime number) is recyclable. This property avoids using large prime numbers as lock values, which will reduce the size of a key value when users and files are inserted into and deleted from the system frequently.

6. The complexity of storage space for keys and locks of m users and n files is only $O(m+n)$ when the overflow problem of key and lock values is ignored.

Consider upgrading a computer system so that a “new” access right, e.g., “read & execute”, is provided. By our scheme, the keys and the locks of the existing users and files don’t have to be changed. The only thing our scheme has to do is to accommodate this new access right for future users and files. Some previous schemes, such as [9, 8, 7], do not have this property. The complexity issues about the access control methods are throughly discussed in the paper [6]. We refer interested readers to the paper.

References

[1] D.E.R. Denning, *Cryptography and Data Security,* Reading, MA:Addison-Wesley, 1982.

[2] R.W. Conway, W.L. Maxwell and W.C. Morgan, On the implementation of security measures in information system, *Commun. ACM* , 15(4) (April 1972), 211-220.

[3] G.S. Graham and P.J. Denning, Protection-principles and practice, *Proc. Spring Jt. Computer Conf.*, 40, AFIPS, Montrale, NJ, 1972, 417-429.

[4] C.C. Chang, On the design of a key-lock-pair mechanism in information protection system, *BIT* , 26 (1986), 410-417.

[5] C.K. Chang and T.M. Jiang, A binary single-key-lock system for access control, *IEEE Transactions on Computer,* 38(10) (1989), 1462-1466.

[6] J.J. Hwang, B.M. Shao and P.C. Wang, A new access control method using prime factorization, *The Computer Journal*, 35(1) (1992), 16-20.

[7] J.K. Jan, C.C. Chang and S.J. Wang, A dynamic key-lock-pair access control scheme, *Computers & Security*, 10 (1991), 129-139.

[8] C.S. Laih, L. Harn and J.Y. Lee, On the design of a single-key-lock mechanism based on Newton's interpolating polynomial, *IEEE Trans. Software Eng.*, 15(9) (September 1989), 1135-1137.

[9] M.L. Wu and T.Y. Hwang, Access control with single-key-lock, *IEEE Trans. Software Eng.*, 10(2) (1984), 185-191.

[10] Min-Shiang Hwang and Wei-Pang Yang, A new dynamic access control scheme based on subject-object-list, *Data & Knowledge Engineering*, 14(1) (1994), 45-56.

[11] Min-Shiang Hwang, Wen-Guey Tzeng, and Wei-Pang Yang, A two-key-lock-pair access control method using prime factorization and time stamp, *IEICE Trans. on Information and Systems*, E77-D(9) (1994), 1042-1046.

[12] C.C. Chang and J.K. Jan, An access control scheme for new users and files, *Int. J. Policy Inf.*, 12(2) (December 1988), 89-98.