A Two-Key-Lock-Pair Access Control Method Using Prime Factorization and Time Stamp

Min-Shiang HWANG*† Wen-Guey TZENG*
Wei-Pang YANG*

December 31, 2002

SUMMARY Many methods, based on the concept of key-lock-pair have been proposed for access control in computer protection systems. However, the proposed methods still either lack of dynamic ability or need quite a lot of computation in performing requests of deleting users/files, inserting users/files, or updating access rights of users to files. In this paper we propose a two-key-lock-pair access control method that is based on the unique factorization theorem and a time stamp mechanism. Our method is dynamic and needs a minimum amount of computation in the sense that it only updates at most one key/lock for each access request, which has not been achieved before.

Key Words: *Access control, Key-lock-pair, Prime factorization theorem, Time stamp.*

1 Introduction

Access control is an important topic in the computer systems [5, 6]. In general, an access control system consists of three components: subject, object and access

*Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 300, Republic of China.
†Directorate General of Telecommunication Laboratories, Ministry of Transportation and Communications, Chung-Li, Taiwan 320, R.O.C.

right. The subject, referred as a user hereafter, is the one which performs access activities. The object, referred as a file hereafter, is the resource that can be used by subjects. The access right of a user to a file is the privilege of the user over the file, which could be "execute", "own", "read", "write", and etc. The access control system in a computer system shall prevent files in the system from being altered, destroyed. or disclosed by unauthorized users.

The most straight forward implementation of access control for a computer is to use a matrix [6], 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 [1, 3, 4, 5, 7, 8, 9, 11, 13] are proposed to resolve this problem. The capability-list and access-list systems are among the earliest [7]. 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.

Using key-lock-pair (KLP) is another approach, in which each user is associated with a key and each file with a lock. When a user make a request of accessing a file, the KLP access control system retrieves the key of the user and the lock of the file and carries out some arithmetic operations on them to get the access right of the user to the file. To our best knowledge, all the proposed KLP access control methods have some drawbacks, such as the KLP access control method in [4] need recompute all the keys in the system when inserting a user to the system.

The third approach is to use two key-lock-pair (2-KLP) method, in which each user is associated with a key and a lock and so is each file [9]. Operations in the 2-KLP access control systems are similar to those in KLP access control systems except that in the 2-KLP access control systems the users and files are treated symmetrically. Some 2-KLP access control systems indeed have an advantage over the KLP access control systems in terms of the number of updated keys/locks for an access request. But, to our best knowledge all the proposed 2-KLP access control systems cannot achieve the goal of updating at most one key/lock for each request of updating the access right of a user to a file, inserting a user/file, or deleting a user/file.

In this paper we propose a new 2-KLP access control method that is based on the unique factorization theorem of integer numbers and the concept of time stamp. Our method achieves optimality in the sense of the number of updated keys/locks for each access request, that is, our 2-KLP access control method updates at most one key/lock for each request. 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 [2].

This paper is organized as follows. Our new 2-KLP access control method is presented in Section 2, in which the correctness of the method is also shown. In Section 3, we analyze the time and space complexities of our method. In Section 4, we conclude and make a remark on dealing with the large numbers induced in our method.

2 New 2-KLP Access Control Method

In this section, we propose a 2-KLP access control method based on the unique factorization theorem of integer numbers and the concept of time stamp. In a 2-KLP 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 data structure in our method consists of a user key-lock table, a file key-lock table, a user prime number stack (SU) and a file prime number stack (SF). Both stacks SU and SF hold primes with smaller ones on the top, which shall be used as lock values of users and files respectively. The use (file) key-lock table has three columns the key value column, the lock value column and the time stamp column. When a user i (file j) is added to the system, we assign a distinct time stamp number TS_i (TS'_j) to the user (file) and pop a prime number from SU (SF) as its lock value L_i (L'_j), i.e., we create an entry in the user (file) key-lock table for the user (file), assign the time stamp to its time stamp column and the lock value to its lock value column. The earliest a user/file is added to the system, the smaller its time stamp is. The key value K_i (K'_j) of user i (file j) is computed as follows.

$$K_i = \prod_{j=j_1,\cdots,j_n} L_j'^{r_{ij}} \tag{1}$$

$$(K'_j = \prod_{i=i_1,\cdots,i_m} L_i^{r_{ij}}) \tag{2}$$

where files $j_1, \cdots, j_n$ (users $i_1, \cdots, i_m$) are in the system when user i (file j) is added to the system and r_{ij} is the access right of user i to file j. Note that we use a number to denote an access right, for example, 0 for "no access", 1 for "read", 2 for "write", 3 for "execute", and etc.

We use the following example to illustrate the above construction.

Example 2.1 *Consider a file system as depicted in Table 1. Let U_1, U_2, …, U_5 and F_1, F_2, …, F_5 be the five users and the five 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, F_5. By the above method we maintain two key-lock tables for users and files as in Table 2.*

Table 1: An access control matrix of the illustrated examples.

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

0 : no access
1 : read
2 : write
3 : execute
4 : own

In the following subsections we give algorithms for inserting a user/file to the system, checking the access right of a user to a file, changing the access right of a user to a file, and deleting a user/file from the system. The correctness of these algorithms are shown in the last subsection.

2.1 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 SU as the lock

Table 2: (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).

	K_i	L_i	TS_i
U_1	Null	2	0
U_2	12	3	3
U_3	162	5	4
U_4	12	7	6
U_5	4725	11	8

(a)

	K'_j	L'_j	TS'_j
F_1	16	2	1
F_2	8	3	2
F_3	2025	5	5
F_4	24500	7	7
F_5	31752	11	9

(b)

13
17
19
23
$\vdots$

(c)

13
17
19
23
$\vdots$

(d)

Table 3: Algorithm for inserting a new user.

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.	$TS \longleftarrow (TS+1)$; /* Set time stamp */
2.	$TS_i \longleftarrow TS$;
3.	$L_i \longleftarrow$ POP SU;
4.	Compute the key value $K_i \longleftarrow \prod_{j=1}^{n} L_j^{\prime r_{ij}}$;

value of the user. Then we compute the key value of the user by Equation (1) the access rights that are currently in the system, which are inputs, of the user to the files and the lock values of the files that are currently in the system. To add a new file to the system, the procedure is similar to that of adding a new user except that we use the lock values of users. The algorithm for inserting a new user to the system is given in Table 3.

2.2 Checking Access Right

When a user requires to access a file, 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 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 access right r_{ij} with respect to user i and file j is

$$r_{ij} = \begin{cases} \max\{l : K_i \bmod (L'_j)^l = 0\} & if \quad TS_i > TS'_j, \\ \max\{l : K'_j \bmod (L_i)^l = 0\} & otherwise. \end{cases} \tag{3}$$

The following example shows the verification of two access requests.

Example 2.2 *Let us consider the example in Example 2.1 again. If an access request* (U_i, F_j, R_{ij}) = *(3,4,3) occurs, the system fetches the time stamp* TS_3 *and* TS'_4 *from the user and file key-lock tables. Since* $TS_3 = 4 < TS'_4 = 7$, *thus*

$$r_{34} \quad = \quad \max\{l : K'_4 \bmod (L_3)^l = 0\} = 3.$$

Since $R_{34} = r_{34} = 3$, *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* $TS_5 = 8 > TS'_4 = 7$, r_{54} *is then computed to be 1. Therefore the access request is denied.*

2.3 Updating the Access Right of a User to a File

When the access right of user i to file j 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. Then we compute the new K_i or K'_j from the old K_i or K'_j according to the equation

$$\begin{cases} K_i & \longleftarrow & K_i L_j'^{(r'_{ij}-r_{ij})} & if \quad TS_i > TS'_j, \\ K'_j & \longleftarrow & K'_j L_i^{(r'_{ij}-r_{ij})} & otherwise. \end{cases} \tag{4}$$

The access right r_{ij} of user i to file j is computed by Equation (3).

The following is an example of updating access right of a user to a file.

Example 2.3 *Assume that the access right* r_{23} *is changed into 2. The system fetches the time stamps* TS_2 *and* TS'_3 *from the user and file key-lock tables. Since* $TS'_3 = 2 > TS_2 = 6$, *thus the access right* r_{23} *of user 2 to file 3 is first computed by Equation (3), and* K'_3 *is then recomputed as* $K'_3 L_2^{(2-4)} = 225$.

Table 4: Algorithm for deleting a user.

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

2.4 Deleting a User/File

When a user is deleted from the system, we simply remove the entry of the user from the user key-lock table. The information of access rights of the deleted user to files is still embedded in the keys of the files that are added to the system after the user (but before the user is deleted). A user does not use the key values of the files that are added to the system before it to compute the access rights of it to them. Even later a new user reuses the lock value of a deleted user. No confusion should occurs. This is because the time stamp mechanism keeps the order of users and files by the time that they are added into the system. Also, in order 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 SU. The prime number is then reserved for new users in the future. To delete a file from the system, the procedure is similar to that of deleting a user.

The algorithm for deleting a user from the system is given in Table 4.

Example 2.4 *Let us consider Example 2.1. Assume that user* 4 *is deleted from the system. We only delete the entry of user 4 from the user key-lock table and push the lock value* L_4 *of the user back to SU.*

2.5 Correctness of the Algorithms

The correctness of our algorithms is based on the user of time stamps, which keep the order of users and files according to the time they are added to the system, and the unique factorization theorem, which guarantees that the access right can be uniquely derived from the key and lock values.

Since the system is dynamic, we consider all possibilities of relationships between files and users. Initially we assume that the system works correctly. When a user i is added to the system, its time stamp TS_i is the largest among all users and files. Its lock value is distinct from all other users'. Its key value is computed by Equation (1). We consider a request of checking the access right of a user j to a file l. In the case that $j = i$, $TS_j > TS'_l$ and the access right r_{jl} is computed by the upper part of Equation (3), which shall be correct by the way that K_i is computed and by the unique factorization theorem. In the case that $j < i$, by the assumption r_{jl} is computed correctly. The correctness of inserting a file can be argued similarly.

To update the access right of a user to a file, the key value K_j or K'_l is recomputed by Equation (4). Since only one key value is updated to accommodate the change of the access right, all other values, including time stamps and other key and other lock values, are not affected. The system is not affected for other operations except the case of checking access right of user j to file l, which is recomputed by Equation (4).

For deleting a user/file from the system, only the entry of the user is deleted from the user-key-lock/file-key-lock table. All others are not affected. Although the information of access rights of the files to the user remains in the system, the system still works correctly, as explained in Subsection 2.4.

Therefore all algorithms used in our 2-KLP access control system are correct.

3 Complexity Analysis

We assume that the system has n users and m files. If we ignore the overflow problem of key values, the space complexity of our method is simply $O(m+n)$ since each user/file possesses a key and a lock. Under the model of bits, the maximum lock value for n users is $O(n \log n)$ since the number of primes (used as lock values) smaller than a value $q/\ln q$, where ln is the natural logarithm. Thus the size of each lock value is $O(\log n)$ bits. Therefore the total number of bits of all lock values is $O(n \log n + m \log m)$. To analyze the size of all key values, we assume that the maximum value of the access rights is a constant c. The maximum value of a key is $O((n \log n)^{cn})$ by Equation (1). Thus its size is $O(n \log n)$ bits. Totally the size of all keys is $O(n^2 \log n + m^2 \log m)$. So, the space complexity of our method is $O(n^2 \log n + m^2 \log m)$ bits.

For the time complexity of our method, we ignore the overflow problem and assume that each arithmetic operation needs $O(1)$ time only. By Equation (1), to insert a new user need access all lock values and access rights of all files. Thus its time complexity is $O(m)$. By Equation (3), to check the access right of a user to a file need $O(1)$ time. By Equation (4), to update the access right of a user to a file need access the lock values of the files with time stamps smaller than the user. Therefore its time complexity is $O(m)$ also. To delete a user/file from the system, because only its corresponding entry is removed from the key-lock tables, the time complexity is $O(1)$.

Finally we consider the number of modified key/lock values for each operation. It is easy to see that our method updates one key for the operation of inserting a user/file and updating the access right of a user to a file, and zero key/lock for the operation of deleting a user/file. Compared with the other methods, which has been analyzed in [8], our method is optimal in the sense of the number of modified key/lock values. This optimality was not achieved before.

4 Conclusion Remark

We have proposed a new dynamic 2-KLP access control method based on the concepts of time stamps and the unique factorization theorem. The merits of our method include the following:

1. By a simple operation on the keys and locks of the user and the file, it reveals the access right of a user to a file.

2. When a user/file is inserted into the system, our method only assigns a key and a lock to the user/file without affecting the other keys and locks.

3. To delete a user/file from the system, it simply erases the entry of the user/file from the user/file-key-lock-table.

4. When the access right of a user to a file is changed, it merely modifies the key and lock of the user or the file without affecting the keys and locks.

5. Lock values are recycled by pushing the freed lock valued back to the stacks SU and SF. Using this mechanism, we can avoid using large prime numbers as lock values, which will reduce the size of key values when users and files are inserted into and deleted from the system frequently.

Indeed, the most serious problems of our and other 2-KLP access control methods is the overflow problem of key values and the complexity of arithmetic operations on overflowed numbers. In order to overcome these problems such that the system can be practical, we 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 [1].

Acknowledgements

The authors wish to thank many anonymous referees for their suggestions to improve this paper. Part of this research was supported by the National Science Council, Taiwan, R.O.C., under contract no. NSC83-0408-E-009-039.

References

[1] Chang, C. C., "On the design of a key-lock-pair mechanism in information protection system," *BIT,* vol. 26, pp. 410-417, 1986

[2] Chang, C. C. and Chen, C. P., "A key-lock-pair mechanism based on generalized Chinese remainder theorem," *Journal of the Chinese Institute of Engineers,* vol. 9, no. 4, pp. 383-390, 1986

[3] Chang, C. C., "An information protection scheme based upon number theory," *The Computer Journal,* vol. 30, no. 3, pp. 249-253, 1987

[4] Chang, C. K. and Jiang, T. M., "A binary single-key-lock system for access control," *IEEE Transactions on Computers,* vol. 38, no. 10, pp. 1462-1466, 1989

[5] Conway, R. W., Maxwell, W. L. and Morgan, W. C., "On the implementation of security measures in information system," *Commun. of ACM,* vol. 15, no. 4, pp. 211-210, Apr. 1972

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

[7] Graham, G. S. and Denning, P. J., "Protection-principles and practice," in *Proc. Spring Jt. Computer Conf.,* vol. 40, AFIPS (Montrale, NJ), pp.417-429, 1972

[8] Hwang, J. J., Shao, B. M. and Wang, P. C., "A new access control method using prime factorisation," *The Computer Journal,* vol. 35, no. 1, pp. 16-20, 1992

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

[10] Knuth, D. E., *The Art of Computer Programming, Vol 2,* Addison-Wesley, Mass., 1981

[11] Laih, C. S., Harn, L. and Lee, J. Y., "On the design of a single-key-lock mechanism based on Newton's interpolating polynomial," *IEEE Trans. Software Eng.,* vol. 15, no. 9, pp. 1135-1137, Sep. 1989

[12] Niven, I. and Zuckerman, H. S., *A Introduction to the Theory of Numbers,* Wiley, New York, 1980

[13] Wu, M. L. and Hwang, T. Y., "Access control with single-key-lock," *IEEE Trans. Software Eng.,* vol. 10, no. 2, pp. 185-191, 1984