Malaysian Journal of Computer Science, Vol. 13 No. 2, December 2000, pp. 33-38

A TWO-KEY ACCESS CONTROL SCHEME BASED ON BINARY ACCESS MODE

Md. Rafiqul Islam
Assistant Professor, Computer Science and Engineering Discipline
Khulna University, Khulna – 9208
Bangladesh

Harihodin Selamat and Mohd. Noor Md. Sap
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia, Jalan Semarak
54100 Kuala Lumpur, Malaysia.
Tel: 6-03-2904957
Fax: 6-03-2930933
email: mmcc0004@utmkl.utm.my

ABSTRACT

A two-key access control scheme is proposed for implementing the access control matrix. The proposed scheme is based on binary form of access rights and time stamp concept. In this scheme each user is assigned one key and each file is also assigned one key. The key of a user or file can be used to derive the access rights to the files depending on the value of time stamp number. The scheme achieves full dynamism. That means, it can easily handle the dynamic access control problem, such as changing access right, adding a user or file and deleting a user or file.

Keywords: ***Access right, Dynamic access and Two-key***

1.0 INTRODUCTION

Data protection is a very important issue in a computer system, because of the increasing complexity of various sorts of information, the large number of users, and the widely used communication networks. The access control system can be used to prevent the information stored in a computer from being destroyed, altered, disclosed or copied by unauthorized users. The access matrix is a conceptual model [3, 7] that specifies the rights that each user possesses for each file. There is a row in this matrix for each user, and a column for each file. Each cell of the matrix specifies the access authorized for the user in the row to the file in the column. The task of access control is to ensure that only those operations authorized by the access matrix actually get executed. An example of an access matrix is shown in Fig. 1.1. We assume that all access rights are expressed by numerals. Linear hierarchy of access privileges may be applied here. That means, the right to read implies the right to execute, the right to write implies the rights to read and execute and so on. In the access matrix shown below user U_1 can delete file F_1 and execute file F_2 and U_3 can read file F_3.

Based on the concept of access control matrix, in 1991 Jan *et al.* proposed two-key-lock access control system to achieve full dynamism [6]. That means when a user or file is added to system, construction of one key-lock is sufficient. On the other hand when a user or file is deleted from the system, deletion of the key-lock is enough for necessary update. After that Hwang *et al.* proposed another two-key-lock system using time stamp concept [9]. Jan *et al.*'s scheme suffers problem to maintain full dynamism that is shown in Hwang *et al.*'s paper. In this paper we proposed a two-key system based on binary access mode and time stamp. The proposed scheme is simple and achieves full dynamism in the sense that performing one addition, deletion or updating needs only modify one key. Since we have got time stamping concept from Hwang *et al.*" paper, we review their method in the next section.

2.0 ACCESS CONTROL SCHEME BASED ON CHINESE REMAINDER THEOREM AND TIME STAMP CONCEPT

In this section we briefly review Hwang *et al.*'s two-key-lock access control scheme based on Chinese remainder theorem [9]. The scheme consists of two tables, one user key-lock table and one file key-lock table. The user (file) key-lock table has three columns: key value column, lock value column and time stamp column. When a user is added to the system, the system assigns the distinct time

Files / *Users*	F_1	F_2	F_3	F_4
U_1	*4*	*4*	*0*	*1*
U_2	*2*	*1*	*3*	*0*
U_3	*1*	*1*	*2*	*1*
U_4	*2*	*1*	*0*	*4*

0: No access
1: Execute
2: Read
3: Write
4: Delete

Fig. 1.1: An access control matrix

stamp number to the user and select a prime number as lock of the user. The key value of the user U_i (ith user) is computed as follows:

$$K_i = \sum_{j=1}^{n} r_{ij} G_j b_j \bmod P \qquad (2.1)$$

Where, $P = \prod_{j=1}^{n} P_j$ (product of all file lock values),

$G_j = P / P_j$ and n is the total number of files in the system.

That means, there will be n such G_j's. Here b_j satisfies $G_j b_j \bmod P_j = 1$. So, $b_j = [\text{inv}(G_j, P_j)] \bmod P_j$. To find out inv($Q_j$, P_j) the extended Euclid's algorithm is required [1, 2]. Access right is computed as

$$r_{ij} = K_i \bmod P_j \qquad (2.2)$$

When a file is added to the system its key value is similarly computed using user lock values. Let us see the construction process of the user and file key-lock tables. For this we consider the access matrix of Fig. 1.1. Suppose users and files are added to the system in the sequence U_1, F_1, F_2, U_2, F_3, U_3, F_4. Let TU_i is the time stamp of user U_i and TF_j is the time stamp of file F_j. In Table 2.1 K_i is the key, and L_i is the lock of the user U_i respectively. In Table 2.2 Q_j is the key and P_j is the lock of file F_j respectively. The lock values are relatively pairwise prime numbers.

Table 2.1: The user key-lock table

User	K_i	L_i	TU_i
U_1	Null	5	0
U_2	7	6	3
U_3	1	7	4
U_4	7	11	6

Table 2.2: The file key-lock table

File	Q_j	P_j	TF_j
F_1	4	5	1
F_2	4	6	2
F_3	135	7	5
F_4	246	11	7

2.2 Checking Access Right

To check the access right of user U_i to file F_j, first the time stamp values TU_i and TF_j of the user and the file is compared. If the time stamp value of the user is smaller than that of the file, *i.e.* user U_i is added to the system before file F_j, the system uses 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 U_i is added to the system after file F_j, the system uses the key of the user and the lock of the file to verify the access right of the user to the file.

Example 2.1: Verification of access right

If U_3 wants to execute the file F_4, the system fetches the time stamp TU_3 and TF_4 from the user and file key-lock tables, since $TU_3 = 4 < TF_4 = 7$

$r_{34} = Q_4 \bmod L_3 = 246 \bmod 7 = 1$.

Since $r_{34} = 1$ is equal to the requested access right *1* (execute), the access request is accepted. On the other hand if U_4 wishes to write in file F_1, the system compare TU_4 and TF_1, since $TU_4 = 6 > TF_1 = 1$, so $r_{41} = K_4 \bmod P_1 = 7 \bmod 5 = 2$

Since $r_{41} = 2$ is smaller than the requested access right *3* (write), the access request is denied.

In this scheme the key construction process is time consuming due to G_j and b_j. Since when the system contains large number of files and users, the computations of the above terms are time consuming. We can see the result of time consummation of such computations in [12]. The size of K_i is proportional to n (number of files in the system) and Q_j is proportional to m (number of users in the system). That means, the K_i and Q_j are very large numbers. On the other hand P_j and L_i are relatively small numbers with respect to K_i and Q_j. Hence verification of access right will be not fast enough, when the system contains large number of users and files. Since K_i and Q_j are very large numbers, the system suffers overflow problem. Using the concept of time stamping we proposed a simple two-key access control scheme based on the binary coding of the access modes (rights). The key construction process of the scheme is simple and verification of access right is easy. On the other hand the system achieves full dynamism. We introduce the proposed scheme in the next section

3.0 TWO-KEY METHOD BASED ON BINARY ACCESS MODE

In this section we will describe the proposed method with respect to the key construction process, checking access right and dynamic access control, such as changing access right, adding a user or file and deleting a user or file.

3.1 Basic Concept

Let each access right r_{ij} in access matrix be represented in its binary form $r_{ij} = (r_{ij}^{c} r_{ij}^{c-1} \ldots r_{ij}^{1})$ where, $c = 1 + \lfloor \log(r_{max}) \rfloor$ and r_{max} is the maximum of access rights ($r_{max} = 4$ according to Fig. 1.1). Suppose there are m users and n files in the system. The system consists of two tables, user key table and one file key table. The user key table contains two columns: key value column and time stamp column. Similarly the file key table has two columns: key value column (key of the file) and time stamp column. The key value of a user is computed from access rights of the user to the files and the key value of a file is computed from the access rights

of the users to the file (the file for which the key value is computed). Suppose K_i denotes the key of user U_i and the key is represented as $K_i = \left(K_i^c, K_i^{c-1}, \ldots, K_i^1\right)$, *i.e.*, each key is broken into c elements. When user U_i is added to the system each element of the key K_i is computed as follows:

$$K_i^z = \sum_{j=1}^{n} r_{ij}^z . 2^j \quad \text{for z } = 1, 2, \ldots, c. \tag{3.1}$$

where r_{ij}^z denotes the zth bit of r_{ij} and $r_{ij}^z \in \{0, 1\}$.

Suppose $c = 3$, then we can compute the elements of the key K_i (the key of the user U_i) as follows:

$$\begin{aligned} K_i^1 &= \sum_{j=1}^{n} r_{ij}^1 . 2^j \\ K_i^2 &= \sum_{j=1}^{n} r_{ij}^2 . 2^j \\ K_i^3 &= \sum_{j=1}^{n} r_{ij}^3 . 2^j \end{aligned} \tag{3.2}$$

where, r_{ij}^1 denotes first bit of the access right r_{ij}.

Similarly when a file is added to the system, we compute an element Q_j, the key of the file F_j as follows:

$$Q_j^z = \sum_{i=1}^{m} r_{ij}^z . 2^i \quad \text{for z } = 1, 2, \ldots, c. \tag{3.3}$$

3.2 Construction Process of the Key Tables

Let us consider the following access control matrix of Fig. 3.1.

Files / Users	F_1	F_2	F_3	F_4
U_1	1	2	0	4
U_2	2	3	3	1
U_3	0	4	1	3

Fig. 3.1: An access control matrix

Files / Users	F_1	F_2	F_3	F_4
U_1	001	010	000	100
U_2	010	011	011	001
U_3	000	100	001	011

Fig. 3.2: A binary access control matrix

By considering the above access control matrix and using binary form of the access rights, we get a binary access control matrix as shown in Fig. 3.2. Let users and files be added to the system in the sequence U_1, F_1, F_2, U_2, U_3, F_3, F_4. Suppose TU_i is the time stamp of user U_i and TF_j is the time stamp of file F_j. Using corresponding access rights depicted in Fig. 3.2 we can compute the keys of the users (files) and their time stamps as follows:

$TU_1 = 0$; $K_1^1 = 0$, $K_1^2 = 0$, $K_1^3 = 0$; $K_1 = (K_1^3, K_1^2, K_1^1) = (0, 0, 0)$.

$TF_1 = 1$; $Q_1^1 = 2^1 = 2$, $Q_1^2 = 0$, $Q_1^3 = 0$; $Q_1 = (Q_1^3, Q_1^2, Q_1^1) = (0, 0, 2)$.

$TF_2 = 2$; $Q_2^1 = 0$, $Q_2^2 = 2^1 = 2$, $Q_2^3 = 0$; $Q_2 = (Q_2^3, Q_2^2, Q_2^1) = (0, 2, 0)$.

$TU_2 = 3$; $K_2^1 = 2^2 = 4$, $K_2^2 = 2^1 + 2^2 = 6$, $K_2^3 = 0$; $K_2 = (K_2^3, K_2^2, K_2^1) = (0, 6, 4)$.

$TU_3 = 4$; $K_3^1 = 0$, $K_3^2 = 0$, $K_3^3 = 2^2 = 4$; $K_3 = (K_3^3, K_3^2, K_3^1) = (4, 0, 0)$.

$TF_3 = 5$; $Q_3^1 = 2^2 + 2^3 = 12$, $Q_3^2 = 2^2 = 4$, $Q_3^3 = 0$; $Q_3 = (Q_3^3, Q_3^2, Q_3^1) = (0, 4, 12)$.

$TF_4 = 6$; $Q_4^1 = 2^2 + 2^3 = 12$, $Q_4^2 = 2^3 = 8$, $Q_4^3 = 2^1 = 2$; $Q_4 = (Q_4^3, Q_4^2, Q_4^1) = (2, 8, 12)$.

Table 3.1: The user key table

User	K_i	TU_i
U_1	(0, 0, 0)	0
U_2	(0, 6, 4)	3
U_3	(4, 0, 0)	4

Table 3.2: The file key table

User	Q_j	TF_j
F_1	(0, 0, 2)	1
F_2	(0, 2, 0)	2
F_3	(0, 4, 12)	5
F_4	(2, 8, 12)	6

3.3 Checking Access Right

To check the access right of user U_i to file F_j, we first compare the time stamp values TU_i and TF_j of the user and the file. If the time stamp value of the user is larger than that of the file, *i.e.* user U_i is added to the system after file F_j, we use the key of the user to verify the access right of the user to the file. If the time stamp value of the user is smaller than that of the file, *i.e.* user U_i is added to the system before file F_j, we use the key of the file to verify the access right of the user to the file. The *algorithm 3.1* for checking access right of a user to a file is given below:

Algorithm 3.1: Checking access right

Steps:

1. Input U_i, F_j and a_{ij} (the request access mode);
2. If $TU_i > TF_j$ then
   Begin
   For $1 \le z \le c$ do
   Compute $r_{ij}^z = \left\lfloor \frac{K_i^z}{2^j} \right\rfloor \bmod 2$;
   End;
   Else
   Begin
   For $1 \le z \le c$ do
   Compute $r_{ij}^z = \left\lfloor \frac{Q_i^z}{2^j} \right\rfloor \bmod 2$;
   End;
3. If $a_{ij} \le r_{ij}$ then
   Access is allowed;
   Else access is denied.

Example 3.1: Verification of access right

Suppose user U_2 wants to write in file F_3. That means, $a_{23} = 2$. The system fetches time stamps TU_2 and TF_3 from the user and file key tables. Since $TU_2 = 3 < TF_3 = 5$, so we use

$$r_{23}^z = \left\lfloor \frac{Q_3^z}{2^i} \right\rfloor \bmod 2 \quad \text{for } z = 1,2,3. \text{(since } c = 3\text{)}.$$

Hence,

$$r_{23}^1 = \left\lfloor \frac{12}{2^2} \right\rfloor \bmod 2 = 1; \quad r_{23}^2 = \left\lfloor \frac{4}{2^2} \right\rfloor \bmod 2 = 1;$$

$$r_{23}^3 = 0;$$

$r_{23} = \left(r_{23}^3 r_{23}^2 r_{23}^1\right) = (011) = 3$ is greater than the requested access right *2* (write), the access is allowed. However, if U_2 wants to delete file F_3, the access will be denied. Because in that case $a_{23} = 4$ and we found $r_{23} = 3 < 4$.

Suppose user U_3 wishes to delete file F_2. That means, $a_{32} = 4$. The system fetches time stamps TU_3 and TF_2 from user and file key tables. Since $TU_3 = 4 > TF_2 = 2$, we use

$$r_{23}^z = \left\lfloor \frac{K_3^z}{2^j} \right\rfloor \bmod 2 \quad \text{for z = 1,2, 3.(since } c = 3\text{). Hence,}$$

$$r_{23}^1 = 0; \; r_{23}^2 = 0; \; r_{23}^3 = \left\lfloor \frac{4}{2^2} \right\rfloor \bmod 2 = 1;$$

$r_{32} = \left(r_{32}^3 r_{32}^2 r_{32}^1\right) = (100) = 4$ is equal to the requested access right *4* (delete), the access is allowed.

3.4 Changing Access Right

Let us consider access right r_{ij} is changed to $p_{ij} = \left(p_{ij}^c p_{ij}^{c-1} \ldots p_{ij}^1\right)$. To update the key we first compare the time stamps TU_i and TF_j of the user and the file. If $TU_i > TF_j$, we recompute the user key K_i. Otherwise we recompute the file key Q_j. By executing *algorithm 3.2*, we can update the key.

Algorithm 3.2: Changing access right

Steps:

1. Input r_{ij} and p_{ij};
2. If $TU_i > TF_j$ then
   Begin
   For $1 \le z \le c$ do
   Begin
   set $t_1 = p_{ij}^z$ and $t_2 = r_{ij}^z$;
   compute $t = t_1 - t_2$;
   If ($t \ne 0$) then
   $K_i'^r = K_i^r + t.2^j$;
   Else
   $K_i'^r = K_i^r$;
   End_for;
   Else
   Begin
   For $1 \le z \le c$ do
   Begin
   set $t_1 = p_{ij}^z$ and $t_2 = r_{ij}^z$;
   compute $t = t_1 - t_2$;
   If ($t \ne 0$) then
   $Q_j'^z = Q_j^z + t.2^i$;
   Else
   $Q_j'^z = K_j^z$;
   End_for;
   End;

Step 4: Output K'_i or Q'_i.

Example 3.2: Changing access right

Suppose the access right $r_{21} = 010$ (see binary access control matrix in Fig. 3.2) is changed to $p_{21} = 011$. Since $TU_2 = 3 > TF_1 = 2$, so we update K_2. Here $K_2 = (0, 6, 4)$. By executing *algorithm 3.2*, we get $K'^1_2 = K^1_2 + 2 = 4 + 2 = 6$, $K'^2_2 = K^2_1 = 6$, $K'^3_2 = K^3_2 = 0$, $K'_2 = (0, 6, 6)$. Let $r_{34} = 011$ will be changed to $p_{34} = 100$. Since $TU_3 = 4 < TF_4 = 6$, we update Q_4. Here $Q_4 = (2, 8, 12)$. So,

by executing *algorithm 3.2*, we get $Q'_4 = Q^1_4 - 2^3 = 12 - 2^3 = 4$ (since $t = -1$), $Q'_4 = Q^2_4 - 2^3 = 8 - 8 = 0$, $Q'_4 = Q^3_3 + 2^3 = 2 + 8 = 10$. So, $Q'_4 = (10, 0, 4)$.

If we wish to verify any access right with changing values of the keys, the result will be correct.

3.5 Adding a User or File

When a user is added to the system, we assign the value of the current time stamp as the time stamp of the user. Then we compute the key value of the user by equation (3.2). To add a new file to the system we assign the current time stamp of the file and compute the key value of the file by equation (3.3).

3.6 Deleting a User or File

The deleting process is very simple. When a user (file) is being deleted from the system, we delete the key value and the time stamp for the user (file) from the user (file) key table.

4.0 DISCUSSIONS

The key construction process of the proposed scheme is simple, since the key is a sum of some terms that are in the form of power of 2. Here we need to consider only the non-zero bits of access right (*i.e.*, for $r^r_{ij} = 1$). As we know the access matrix is usually a sparse [3, 7, 11] and we do not need to consider the zero access rights as well as zero bits of non-zero access rights, the key construction process is obviously simple. The user (file) key table contains only key value and time stamp value. That means we use simple user (file) key table. To find out each bit of an access right the system requires *2* divisions. So, to find *c* bits of the access rights it requires *2c* divisions and *c* is usually a very small number, such as $c = 3$ or $c = 4$. Changing of access right is also easy. New user (file) can be easily added to the system by constructing the corresponding key value. Deletion of a user (file) is very simple. The storage required to implement the proposed scheme is $O(c(m + n)) = O(m + n)$, that is one key for one user and one key for one file. If we consider $c = 4$, we can accommodate $2^4 - 1 = 15$ access modes (execute, read and so on) and that will be enough for practical use.

As we know each key of a user consists of *c* elements. If we take one integer for one element, the key can be defined as a structured (record) of *c* integers. However, one integer may not be enough for storing one element of the key. For instance, if we consider a *32*-bit computer, the largest integer allowed by such a computer is 2^{32}. Since each element is a sum of several terms that are in the form of power of *2*, there may be an overflow to hold one element using one integer. In such a case we must take several integers for each element. So, we require an array of integers for holding one element. Thus each element of a key is an array of several integers and each key is a structure of such *c* arrays.

5.0 CONCLUSION

In this paper we proposed a very simple and efficient two-key access control scheme based on binary access modes and time stamp concept. For the proposed method we devised algorithms for verifying and changing access rights. The scheme achieves full dynamism. That is, changing the access right, insertion as well as deletion of any user (file) can be successfully implemented by performing operations on one key. The required storage for implementation of the scheme is not very large. The proposed method gives the flexibility of using access modes and the overflow problem can be easily handled. Furthermore, the proposed method is very suitable for implementing sparse access control matrix.

REFERENCES

[1] D. E. R. Denning, *Cryptography and Data Security*; Addison-Wesley, Reading, MA, 1983.

[2] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd edition, Readind MA: Addison-Wesley, 1981.

[3] G. S. Graham and P. J. Denning, *Protection- Principle and Practice*; Proc. Spring Joint Computer Conf., Vol. 40, AFIPS Press, Montvale, NJ, 1972, pp. 417-429.

[4] J. K. Jan, *A Single Key Access Control Scheme in Information Protection System*, Proceedings of National Computer Symposium, Taiwan, 1987, pp. 299-303.

[5] J. C. R. Tseng and W. P. Yang, *A New Access Control Scheme with High Data Security*, Ninth Annual International Phoenix Conference on Computer and Communications, IEEE Comp. Soc. Press, 1990, pp. 683-688.

[6] J. K. Jan, C. C. Chang and S. J. Wang, *A Dynamic Key-Lock-Pair Access Control Scheme*, Computers and Security, Vol. 10, 1991, pp. 129-139.

[7] R. S. Sandhu and P. Samarati, *Access Control: Principle and Practice*, IEEE Communication Magazine, 1994, pp. 40-48.

[8] C. C. Chang, J. J. Shen and T. C. Wu, *Access Control with Binary Keys*, Computers and Security, Vol. 13, 1994, pp. 681-686.

[9] M. S. Hwang, W. G. Tzeng and W. P. Yang, *An Access Control Scheme Based on Chinese Remainder Theorem and Time Stamp Concept*, Computers and Security, Vol. 15, No. 1, pp. 73-81, 1996.

[10] C. C. Chang, D. C. Lou and T. C. Wu, *A Binary Access Control Method Using Prime Factorization*, Information Sciences, 1997, pp. 15-26.

[11] M. R. Islam, H. Selamat and M. N. M. Sap, *A Binary Access Control Scheme with Single Key*, Journal of Information Technology, UTM, Vol. 9, No. 2, 1997, pp. 1-10.

[12] M. R. Islam, H. Selamat. and M. N. M. Sap, *A Technique to Ease a Common and Major Computational Complexity of the Security Schemes Based on Chinese Remainder Theorem*, Journal of Information Technology, UTM, Vol. 10, No. 1 (To appear).

BIOGRAPHY

Md. Rafiqul Islam obtained his Master of Science in Engineering (Computers) from Azerbaijan Polytechnic Institute in 1987. He is an Assistant Professor of Computer Science and Engineering Discipline of Khulna University, Bangladesh. Currently, he is on study leave and doing Ph.D at Faculty of Computer Science and Information Systems, in Universiti Teknologi Malaysia. His research areas include design and analysis of algorithms, Database security and Cryptography. He has published a number of papers related to these areas. He is an associate member of Bangladesh Computer Society.

Harihodin Selamat holds an MSc from Cranfield University, UK and a Ph.D from the University of Bradford, UK both in computer science. Currently he is an Associate Professor at the Faculty of Computer Science and Information Systems in Universiti Teknologi Malaysia. His research areas include Database security, Database design and Software engineering.

Mohd Noor Md. Sap is an Associate Professor at the Faculty of Computer Science and Information Systems in Universiti Teknologi Malaysia. a B.Sc. (Hons) from the National University of Malaysia, an MSc from Cranfield University, UK, and a Ph.D from the University of Strathclyde, UK. He is currently carrying out research in Database security, Case-based reasoning and Information retrieval.