A BINARY ACCESS CONTROL SCHEME WITH HIGH DATA SECURITY

Md. Rafiqul Islam and Mohd. Noor Md. Sap[+]

Computer Science and Engineering Discipline
Khulna University, Khulna-9208, Bangladesh
E-mail: cseku@khulna.bangla.net

[+]Faculty of Computer Science and Information System
University Teknologi Malaysia, Johar Bahru, Malaysia.
E-mail: mohdnoor@fsksm.utm.my

ABSTRACT

An access control scheme for implementing the access matrix is presented. The proposed scheme is based upon binary form of access rights. In this scheme each user is assigned one key. The keys are possessed by the users and can be used to derive the access rights to the files. In this scheme attention has been given to the access information (keys of the users) itself. This scheme is more effective and efficient for the systems where files are accessible to only limited number of users.

Keywords: Access right, binary form and data security.

1.0 Introduction

Information protection is 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 is one of the important issues to protect infromation in a computer system. 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 [2, 9] 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 Figure 1.1. We assume that all access rights are expressed by numerical and 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, the user U_1 can execute the file F_1 and read the fileF_2 and U_3 can delete the file F_2.

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

0-No access, 1-Execute, 2-Read, 3-Write, 4-Delete
Figure 1.1: An access control matrix

Wu and Hwang [3] proposed an alternative scheme storing just one key for each user and one lock for each file. In their system lock is a function of key and the access right of the user is computed by operations on lock and key. Several relevant systems [4-8] appeared in the literature after Wu and Hwang's work. Jan in 1987 proposed a single key access control scheme [10] and Tseng *et al.* in 1990 proposed another single key access control scheme with high data security [11]. These two schemes are simple, but they need complex computation for generating keys as well as to find the access rights. The schemes have a common and serious problem that is overflow problem. In this paper we proposed a single key access control scheme with high data security that is based on binary form of the access rights. The proposed scheme is very simple. The key construction and access right verification processes are easy. Furthermore the scheme reduced overflow problem by breaking the key of each user into vector elements. In the proposed scheme attention has been given to security of access information itself as Tseng *et al.'s* scheme. That is why we will review Tseng *et al.'s* scheme in next section.

2.0 ACCESS CONTROL SCHEME WITH HIGH DATA SECURITY

In 1990 Tseng *et al.* proposed an access control scheme with high data security that is based on generalized Markel-Helman cryptosystem [11]. They paid attention to security of the access control information itself. In the system each user possesses a key and the access rights of the user to the files can be derived from the key. The key construction process of the system is as follows:

$$K_i = \sum_{j=1}^{n} a_{ij} A_j^{'} \quad \text{for } i = 1, 2, \ldots, m \tag{2.1}$$

Where $= A_j$. $w \bmod d$, $A_j = T^{j-1}$, $d > T^n - 1$, $w < d$, $gcd\,(w, d) = 1$, gcd – greatest common divisor, m is the total number of users, n is the total number of files and T is the total number of access right used in the system. The access right of user U_i to file F_j is computed as below:

$$a_{ij} = \left\lfloor \frac{K_i^{'}}{T^{j-1}} \right\rfloor \bmod T \tag{2.2}$$

Where $K_i^{'} = K_i\,\theta \bmod d$, and θ is an integer such that

$$w.\,\theta \bmod d = 1 \tag{2.3}$$

The solution of the equation (2.3) can be written as θ = Inv $[w, d]$, where Inv denotes inverse (multiplicative inverse) of w and d that can be computed by extended Euclid's algorithm [1]. The scheme achieves high data security due to two parameters d and w. If these two parameters are kept secret, it is very difficult to get access information. However, the serious problems of the scheme are overflow problem and complexity (overhead) of computation due to T. The value of T easily be 7 and T^n is a very large number if n is not very small number.

In 1997, Islam *et al.* poposed a single key method [12]. The scheme is simple one and based on binary value of access rights. In the scheme overflow problem is reduced due to using of vector elements of key value. However in the scheme no attention has been given to data security. That means, if anyone gets key value of a user, S/he can easily know access information of the user to the file. By exploiting the data security feature of Jan *et al.'s* scheme and the simple and flexible scheme of Islam *et al.* we propose a binary access control scheme with high data security in the next section.

3.0 BINARY ACCESS CONTROL WITH HIGH DATA SECURITY

We will describe the proposed method with respect to key construction process, verification of access right, updating access right and adding/deleting a user or file. In the first subsection the basic concept of the method is described that includes key constrction process and verification of access right.

3.1 Basic Concept

In the proposed method each access right r_{ij} is represented as ($r_{ij}^1\ r_{ij}^2\ \ldots\ r_{ij}^c$) which is a binary form, where $c = 1 + \lfloor \log (r_{max}) \rfloor$ and r_{max} is the maximum value of access right. Suppose that there are m users and n files in the system. Here each user is assigned a key that is computed from the binary form of access rights and some parameters. The access rights of a user to files can be derived from user key. Here $K_i = (K_i^1, K_i^2, \ldots, K_i^c)$, *i.e.*, each key is broken into c elements. The elements of K_i are computed as follows

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

Where r_{ij}^z denotes the zth bit of r_{ij} and $r_{ij}^z \in \{1, 2\}$.

The Parameter B_j is computed as

$$B_j = 2^{j-1}.w \bmod d \quad \text{for } 1 \le j \le n \tag{3.2}$$

Suppose $c = 3$, *i.e.*, the system may include $r_{max} = 7$ and that is enough for a system. Then there will be 3 elements of K_i and they are as follows

$$K_i^1 = \sum_{j=1}^{n} r_{ij}^1 . B_j$$

$$K_i^2 = \sum_{j=1}^{n} r_{ij}^2 . B_j \tag{3.3}$$

$$K_i^3 = \sum_{j=1}^{n} r_{ij}^3 . B_j$$

Where r_{ij}^1 denotes first bit of r_{ij} and so on.

The access right r_{ij} of user U_i to file F_j can be computed as below

$$r_{ij}^z = \left\lfloor \frac{Q_i^z}{2^{j-1}} \right\rfloor \bmod 2 \quad \text{for } z = 1, 2, ..., c \tag{3.4}$$

Where $Q_i^z = K_i^z x \bmod d$ for $z = 1, 2, ..., c$ and x is a positive integer such that $x.w \equiv 1 (\text{mod } d)$ or $x.w \bmod d = 1$.

a) Example 3.1 key construction process

Consider access matrix of Figure 3.1 and using binary form of access rights, we found a binary access control matrix shown in Figure 3.2

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

Figure 3.1: An access control matrix

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

Figure 3.2: A binary access control matrix
(according to Figure 3.1)

Considering the access control matrix in Figure 3.2 and using equations (3.1) and (3.2) keys of the users can be computed. At first we compute the parameter B_j for $1 \le j \le n$. For this we choose $w = 5$ and from the system we get $m = 3$, $n = 4$. Thus we get

$2^4 - 1 = 15 \therefore$ we take $d = 17 > 15$. Then

$B_1 = 2^{1-1}.5 \bmod 17 = 5$

$B_2 = 2^{2-1}.5 \bmod 17 = 10$

$B_3 = 2^{3-1}.5 \bmod 17 = 3$

$B_4 = 2^{4-1}.5 \bmod 17 = 6$

So, the keys for users are computed as follows:

For user U_1:

$$K_1^1 = \sum_{j=1}^{n} r_{1j}^1 B_j = 1 \times 5 + 0 \times 10 + 0 \times 3 + 0 \times 6 = 5$$

$$K_1^2 = \sum_{j=1}^{n} r_{1j}^2 B_j = 0 \times 5 + 0 \times 10 + 1 \times 3 + 0 \times 6 = 3$$

$$K_1^3 = \sum_{j=1}^{n} r_{1j}^3 B_j = 0 \times 5 + 0 \times 10 + 0 \times 3 + 1 \times 6 = 6$$

Thus, $K_1 = (K_1^1, K_1^2, K_1^3) = (5, 3, 6)$

For User U_2:

$$K_2^1 = \sum_{j=1}^{n} r_{2j}^1 B_j = 0 \times 5 + 0 \times 10 + 0 \times 3 + 0 \times 6 = 0$$

$$K_2^2 = \sum_{j=1}^{n} r_{2j}^2 B_j = 1 \times 5 + 0 \times 10 + 1 \times 3 + 0 \times 6 = 8$$

$$K_2^3 = \sum_{j=1}^{n} r_{2j}^3 B_j = 1 \times 5 + 1 \times 10 + 0 \times 3 + 0 \times 6 = 15$$

So, $K_2 = (K_2^1, K_2^2, K_2^3) = (0, 8, 15)$

For User U_3:

$$K_3^1 = \sum_{j=1}^{n} r_{3j}^1 B_j = 0 \times 5 + 1 \times 10 + 0 \times 3 + 0 \times 6 = 10$$

$$K_3^2 = \sum_{j=1}^{n} r_{3j}^2 B_j = 1 \times 5 + 0 \times 10 + 0 \times 3 + 0 \times 6 = 5$$

$$K_3^3 = \sum_{j=1}^{n} r_{3j}^3 B_j = 0 \times 5 + 0 \times 10 + 0 \times 3 + 1 \times 6 = 6$$

So, $K_3 = (K_3^1, K_3^2, K_3^3) = (10, 5, 6)$

So we got the keys of users

$K_1 = (5, 3, 6)$; $K_2 = (0, 8, 15)$; $K_3 = (10, 5, 6)$

a) Example 3.2 Verification of access right

Suppose U_1 wants to read file F_3. That means, we have to compute a_{13} = ? Here $K_1 = (5, 3, 6)$.

According to equation (3.4), we get $w.x \bmod = 1$, $5x \bmod 17 = 1$, so $x = 7$.

Thus, $Q_1^1 = K_1^1 x \bmod d = 35 \bmod 17 = 1$

$$Q_1^2 = K_1^2 \, x \bmod d = 21 \bmod 17 = 4$$

$$Q_1^3 = K_1^3 \, x \bmod d = 42 \bmod 17 = 8$$

So, $r_{13}^1 = \left\lfloor \frac{1}{4} \right\rfloor \bmod 2 = 0$

$$r_{13}^2 = \left\lfloor \frac{4}{4} \right\rfloor \bmod 2 = 1$$

$$r_{13}^3 = \left\lfloor \frac{8}{4} \right\rfloor \bmod 2 = 0$$

$$r_{13} = (r_{13}^1 \, r_{13}^2 \, r_{13}^3) = (010) = 2$$

Since the system found $r_{13} = 2$ (read) is equal to the requested access right read or 2, so the access request is allowed. If the value of requested access right is less than or equal to the value of access right that is found from the system, then the access request is allowed. Otherwise the access request is denied.

3.2 **Changing Access Right**

Let access right $r_{ij} = (r_{ij}^1 \, r_{ij}^2 \, \ldots r_{ij}^c)$ is changed to $s_{ij} = (s_{ij}^1 \, s_{ij}^2 \ldots s_{ij}^c)$. Here we have update key K_i and this can be done by using the following algorithm 3.2. It can be noticed that there will be no change in B_j.

Algorithm 3.2: Changing access right

// B_j is computed as section 3.1//

Input: K_i, r_{ij} and s_{ij}

Output: Update K_i

Steps: 1. Input r_{ij} and s_{ij}

2. For $1 \le z \le c$ do

begin

set $b_1 = s_{ij}^z$ and $b_2 = r_{ij}^z$

Compute $t = b_1 - b_2$

If $(t \neq 0)$ then

$K_i^z = K_i^z + t.\, B_j$

Else

K_i^z is remained unchanged

3. Output updated K_i^z

a) Example 3.3 Changing access right

Suppose that access right r_{23} = 010 is changed to S_{23} = 011. From example 3.1 we have K_2 = (0, 8, 15) and B_2 = 3. By the algorithm 3.2 we get new value of keys as below

$K_2^1 = 0$ (since $s_{23}^1 = 0$, $r_{23}^1 = 0$ and $t = 0$),

$K_2^2 = 0$ (since $s_{23}^2 = 1$, $r_{23}^2 = 1$ and $t = 0$),

$K_2^3 = 0$ (since $s_{23}^3 = 0$, $r_{23}^3 = 0$ and $t = 1$),

So, new value of K_2 = (0, 8, 18).

Let r_{11} = 100 changed to S_{11} = 011. We have K_1 = (5, 3, 6) and B_1 = 5. By algorithm 3.2 we get

$K_1^1 = 5 - 5 = 0$ (since t = -1)

$K_1^2 = 3 + 5 = 8$ (since t = -1) and

$K_1^3 = 6 + 5 = 11$

So, new value of K_1 = (0, 8, 11).

Let see whether we can verify access right with updated values of the keys. Suppose that we wish to compute r_{21} = ? We have K_2 = (0, 8, 18).

$$Q_2^1 = K_2^1.x \bmod d = 0 \bmod 17 = 0$$

$$Q_2^2 = K_2^2.x \bmod d = 8 \times 7 \bmod 17 = 5$$

$$Q_2^3 = K_2^3.x \bmod d = 18 \times 7 \bmod 17 = 5$$

$$r_{21}^1 = 0,\ r_{21}^2 = 5 \bmod 2 = 1,\ r_{21}^3 = 7 \bmod 2 = 1$$

r_{21} = 011 which is correct (see Figure 3.2). If we want to verify any access right with updated value of the key(s), the result will be correct.

3.3 **Adding and deleting a user or File**

In this system the process of deletion of user is very simple. When a user is added, the system will construct the key (elements of key vector) for the new user. If a user is deleted from the system, the key for the user is just deleted from the system. Here we can see Figure 3.2 of section 3.1, if we want to add U_4, we have to construct $K_4 = (K_4^1, K_4^2, K_4^3)$. On the other hand if we want to delete U_2 from the system, we will delete the key $K_2 = (K_2^1, K_2^2, K_2^3)$. When a file is added to the system, we have to update the keys

of users who will possess non-zero access right to the file. Suppose that we want to add F_5 to the system, where $r_{15} = 000$, $r_{25} = 010$, $r_{35} = 100$, $r_{45} = 000$. That means, U_1 and U_4 has no access to the file F_5 and U_2 can read F_5 and U_3 can delete F_5. In this case if we see key construction process of section 3.1, we notice that we have to update key element K_2^2 of key K_2 (since $r_{25}^2 \neq 0$) and key element K_3^1 of K_3 (since $r_{35}^1 \neq 0$). Similarly in case of file deletion we need update the keys of users who can access the file to be deleted.

4.0 **Discussion**

The key construction process of the proposed binary access control 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 non-zero bits of the access right (*i.e.*, for $a_{ij}^z = 1$). As we know the access matrix is usually a sparse [7, 9] and we do not need to consider the zero access rights as well as the zero bits of non-zero access rights, the key construction process is easy. Here the access right verification process is also easy. By executing a simple algorithm we can update the key(s) in case of access right changing. The storage required to implement the proposed scheme is $O(m)$ *i.e.*, one key (key vector) for one user. Here in the system file does not possess any key or lock. Only the user posses key and the storage complexity is $O(m)$. So, we can say that the storage requirement for the system is small. The special feature of the scheme is access information security due to two parameters x and d. If these two parameters are kept secret, it is very difficult to get access information. On the other hand this scheme reduced overflow by breaking the key into its elements (since each key is a vector of c elements). However, Tseng *et al.'s* scheme suffers overflow problem due to T.

5.0 **Conclusion**

In this paper we proposed a very simple and efficient scheme based on binary access modes. For the proposed method we devised formulas for constructing keys and verification of access rights. Here an algorithm for updating access right is also devised. One good feature of our system is that attenction has been given to the security of access information itself. The storage required to implement our scheme is small. The proposed method is very suitable for implementing sparse access control matrix.

Reference

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

[2] G. S. Graham and P. J. Denning, *Protection-Principle and Practice*; Proc. Spring Joint Comuter Conf., Vol.40, AFIPS PRESS, Montavle, NJ, 1972, pp. 417-429.

[3] M.L. Wu and T. Y. Hwang, *Access Control with Single-Key-Lock*; IEEE Transaction on Software Engg., Vol.SE-10, No.2, 1984, pp. 185-191.

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

[5] C. C. Chang, *An Information Protection Scheme Based upon Number Theory*; The Computer Journal, Vol.30, No.3, 1987, pp. 249-253.

[6] C. K. Chang and T. M. Jiang, *A Binary Single-key-lock system for Access Control*; IEEE Transaction on Computers, Vol.38, No.10, 1989, pp. 1462-1466.

[7] J. J. Hwang, B. M. Shao and P. C. Wang, *A New Access Control Method Using Prime Factorization*; The Computer Journal, Vol.35, No.1, 1992, pp. 16-20.

[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] R. S. Sandhu and P. Samarati, *Access Control: Principle and practice*, IEEE communication magazine, 1994, pp. 40-48.

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

[11] 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.

[12] 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