

# A New Binary Access Control Method of User-File Key with Time Stamp and Data Security

Md. Rafiqul Islam and Mir Shahidul Islam

Computer Science and Engineering Discipline,  
Khulna University,  
Bangladesh

## Abstract

*A new binary access control method of user-file key with time stamp and data security is proposed. The proposed method is based upon binary form of access rights. In this method each user is assigned one key and each file is assigned another key. The key of a user or a file can be used to reveal the access rights to the files depending on the value of time stamp. The method achieves full dynamism. That means changing of an access right, inserting a user or a file and deleting a user or a file; each of these operations needs only modification of a particular key. The proposed method achieves security of access data itself due to two secret parameters.*

**Keywords:** Access right, Binary form, User-file key, Time stamp, Data security.

## 1 Introduction

Information 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 method is one of the important issues to protect information in a computer system. The access control method can be used to prevent the information stored in a computer

from being destroyed, altered, disclosed or copied by unauthorized users. The access control method can be enforced using access control matrix. The access matrix is a conceptual model [1, 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 each 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 read the file  $F_1$  and execute the file  $F_2$  and  $U_3$  can delete the file  $F_2$ .

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

0-No access, 1-Execute, 2-Read, 3-Write, 4-Delete

Figure 1.1: An access control Matrix

Based on the concept of access control matrix in 1987 Jan proposed a single key access control scheme [3]. The scheme is simple and easy to implement. However, the scheme did not achieve full dynamism. In 1991 Jan *et al.* proposed two-key-lock access control system to achieve full dynamism [5]. That means, when a user or a file is added to the system, construction of one key-lock is sufficient. On the other hand when a user or a 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 systems using time stamp concept [6]. Jan *et al.*'s scheme suffers problem to maintain full dynamism that is shown in Hwang *et al.*'s paper [6]. Here by exploiting the single key access control scheme of Jan [3], Tseng *et al.*'s a new access control scheme with high data security [4], Hwang *et al.*'s two-key-lock system and time stamp concept [6], Chang *et al.*'s access control with binary keys [8] and M. R. Islam *et al.*'s a binary access control scheme with high data security [9], a binary control method with user-file key and time stamp is proposed. The proposed method is simple and achieves full dynamism in the sense that performing one inserting, deleting or updating operation need only modification of one key (user key or file key). So the time complexity of each operation is very dynamic. The proposed scheme uses only  $O(m + n)$  memory space for  $m$  users and  $n$  files.

In this paper we proposed access control scheme with time stamp and 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. The algorithm for verification of access right and changing access rights are devised here. The method achieves full dynamism due to time stamp number. Furthermore the scheme reduced overflow problem by breaking the key into its elements. In the proposed

scheme attention has been given to security of access information itself as Tseng *et al.*'s scheme.

## 2 The concept of Access Control Scheme with the Time Stamp

When a user  $U_i$  is added to the system, a time stamp number,  $TU_i$  will be assigned for the user. Similarly, when a file  $F_j$  is added to the system, a time stamp number,  $TF_j$  will be assigned for the file. To verify the access the right of user  $U_i$  to the file  $F_j$ , at first the time stamp numbers  $TU_i$  and  $TF_j$  of the user and file is compared. If the time stamp number of the user is less than that of the file, then we can easily say that user  $U_i$  was added to the system before  $F_j$ . On the other hand, If the time stamp number of the user is greater than that of the file, then we can easily say that user  $U_i$  was added to the system after the file  $F_j$ . We compute the keys for the users and files and also choose time stamp value for the users and files in the next section.

## 3 Binary Access Scheme with User-File key, Time stamp and 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 sub section basic concept of the method is described that includes key construction process and verification of access right.

### 3.1 Key construction Process

In the proposed method each access right  $r_{ij}$  is represented as  $(r_{ij}^1 r_{ij}^2 \dots r_{ij}^c)$  which is a binary form where  $c = I + \lfloor \log(r_{max}) \rfloor$  and  $r_{max}$  is the maximum value of access rights. Suppose that there are  $m$  users and  $n$  files in the system. Here each user and file are

assigned keys that are computed from the binary form of access rights and some parameters. Thus the system consists of two tables, one user key table and one file key table. When a user is added to the system, a time stamp number is assigned for the user. Similarly, when a file is added to the system, a time stamp number is assigned for the file. Thus, the user key table consists key value column (key of the user) and time stamp column. Similarly, the file key table contains key value column (key of the file) and column for the time stamp.

The key of the user  $U_i$  is broken into  $c$  elements, such as  $UK_i = (UK_i^1, UK_i^2, \dots, UK_i^c)$ . The elements of  $UK_i$  are computed as follows:

$$UK_i^z = \sum_{j=1}^n r_{ij}^z B_j \quad \text{for } i = 1, 2, 3, \dots, m$$

and  $z = 1, 2, 3, \dots, c \dots (3.1)$

The key of the files  $F_j$  is broken into  $c$  elements such as  $FK_j = (FK_j^1, FK_j^2, \dots, FK_j^c)$ . The elements of  $FK_j$  are computed as follows:

$$FK_j^z = \sum_{i=1}^m r_{ij}^z D_i \quad \text{for } j = 1, 2, 3, \dots, n$$

and  $z = 1, 2, 3, \dots, c \dots (3.2)$

Where  $r_{ij}^z$  denotes the  $z^{th}$  bit of  $r_{ij}$  and the parameter  $B_j$  is computed as  $B_j = 2^{j-1} \cdot w \text{ mod } d$  for  $1 \leq j \leq n \dots (3.3)$

Where  $w$  and  $d$  two positive integers, and  $d > 2^n - 1$

Suppose that the number of files in the system will be greater than or equal to that of users *i.e.*,  $m \leq n$  and we take  $D_i = B_j$  for  $1 \leq i \leq m$  and  $i = j$ .

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

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

$$UK_i^2 = \sum_{j=1}^n r_{ij}^2 B_j$$

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

There will be also 3 elements of  $FK_j$  and they are as follows,

$$FK_j^1 = \sum_{i=1}^m r_{ij}^1 D_i$$

$$FK_j^2 = \sum_{i=1}^m r_{ij}^2 D_i$$

$$FK_j^3 = \sum_{i=1}^m r_{ij}^3 D_i$$

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

### Example 3.1 key construction process

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

FILES USERS	$F_1$	$F_2$	$F_3$	$F_4$
$U_1$	2	1	0	2
$U_2$	3	0	2	1
$U_3$	4	1	0	2

Figure: 3.1 An access control matrix

FILES USERS	$F_1$	$F_2$	$F_3$	$F_4$
$U_1$	010	001	000	010
$U_2$	011	000	010	001
$U_3$	100	001	000	010

Figure: 3.2 A binary access control matrix

Consider the access control matrix in Figure 3.2 and using equations (3.1) and (3.3) keys of the users can be computed. At first we compute the parameter  $B_j$  for  $1 \leq j \leq$

$n$ . For this we choose  $w = 5$  and from the system we get  $m = 3, n = 4$ . Thus we get

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

Then

$$\begin{aligned} B_1 &= 2^{1-1} \cdot 5 \bmod 17 = 5 \\ B_2 &= 2^{2-1} \cdot 5 \bmod 17 = 10 \\ B_3 &= 2^{3-1} \cdot 5 \bmod 17 = 3 \\ B_4 &= 2^{4-1} \cdot 5 \bmod 17 = 6 \end{aligned}$$

So, the keys for users are computed as follows:

For user  $U_1$ :

$$\begin{aligned} UK_1^1 &= \sum_{j=1}^n r_{1j}^1 B_j = 0 \times 5 + 0 \times 10 + 0 \times \\ &3 + 0 \times 6 = 0 \end{aligned}$$

$$\begin{aligned} UK_1^2 &= \sum_{j=1}^n r_{1j}^2 B_j = 1 \times 5 + 0 \times 10 + 0 \times \\ &3 + 1 \times 6 = 11 \end{aligned}$$

$$\begin{aligned} UK_1^3 &= \sum_{j=1}^n r_{1j}^3 B_j = 0 \times 5 + 1 \times 10 + 0 \times \\ &3 + 0 \times 6 = 10 \end{aligned}$$

$$\text{So, } UK_1 = (UK_1^1, UK_1^2, UK_1^3) = (0, 11, 10)$$

For user  $U_2$  :

$$\begin{aligned} UK_2^1 &= \sum_{j=1}^n r_{2j}^1 B_j = 0 \times 5 + 0 \times 10 + 0 \times \\ &3 + 0 \times 6 = 0 \end{aligned}$$

$$\begin{aligned} UK_2^2 &= \sum_{j=1}^n r_{2j}^2 B_j = 1 \times 5 + 0 \times 10 + 1 \times \\ &3 + 0 \times 6 = 8 \end{aligned}$$

$$\begin{aligned} UK_2^3 &= \sum_{j=1}^n r_{2j}^3 B_j = 1 \times 5 + 0 \times 10 + 0 \times \\ &3 + 1 \times 6 = 11 \end{aligned}$$

$$\text{So, } UK_2 = (UK_2^1, UK_2^2, UK_2^3) = (0, 8, 11)$$

For user  $U_3$ :

$$\begin{aligned} UK_3^1 &= \sum_{j=1}^n r_{3j}^1 B_j = 1 \times 5 + 0 \times 10 + 0 \times \\ &3 + 0 \times 6 = 5 \end{aligned}$$

$$\begin{aligned} UK_3^2 &= \sum_{j=1}^n r_{3j}^2 B_j = 0 \times 5 + 0 \times 10 + 0 \times 3 \\ &+ 1 \times 6 = 6 \end{aligned}$$

$$\begin{aligned} UK_3^3 &= \sum_{j=1}^n r_{3j}^3 B_j = 0 \times 5 + 1 \times 10 + 0 \times 3 \\ &+ 1 \times 6 = 10 \end{aligned}$$

$$\text{So, } UK_3 = (UK_3^1, UK_3^2, UK_3^3) = (5, 6, 10)$$

So we got the keys of users

$$UK_1 = (0, 11, 10); UK_2 = (0, 8, 11); UK_3 = (5, 6, 10).$$

Consider the access control matrix in Figure 3.2 and using equations (3.2) and (3.3) keys of the files can be computed as follows,

For file  $F_j$ :

$$FK_1^1 = \sum_{i=1}^m r_{i1}^1 D_i = 0 \times 5 + 0 \times 10 + 1 \times 3 = 3$$

$$FK_1^2 = \sum_{i=1}^m r_{i1}^2 D_i = 1 \times 5 + 1 \times 10 + 0 \times 3 = 15$$

$$FK_1^3 = \sum_{i=1}^m r_{i1}^3 D_i = 0 \times 5 + 1 \times 10 + 0 \times 3 = 10$$

$$\text{So, } FK_1 = (FK_1^1, FK_1^2, FK_1^3) = (3, 15, 10)$$

For file  $F_2$ :

$$FK_2^1 = \sum_{i=1}^m r_{i2}^1 D_i = 0 \times 5 + 0 \times 10 + 0 \times 3 = 0$$

$$FK_2^2 = \sum_{i=1}^m r_{i2}^2 D_i = 0 \times 5 + 0 \times 10 + 0 \times 3 = 0$$

$$FK_2^3 = \sum_{i=1}^m r_{i2}^3 D_i = 1 \times 5 + 0 \times 10 + 1 \times 3 = 8$$

$$\text{So, } FK_2 = (FK_2^1, FK_2^2, FK_2^3) = (0, 0, 8)$$

For file  $F_3$ :

$$FK_3^1 = \sum_{i=1}^m r_{i3}^1 D_i = 0 \times 5 + 0 \times 10 + 0 \times 3 = 0$$

$$FK_3^2 = \sum_{i=1}^m r_{i3}^2 D_i = 0 \times 5 + 1 \times 10 + 0 \times 3 = 10$$

$$FK_3^3 = \sum_{i=1}^m r_{i3}^3 D_i = 0 \times 5 + 0 \times 10 + 0 \times 3 = 0$$

$$\text{So, } FK_3 = (FK_3^1, FK_3^2, FK_3^3) = (0, 10, 0).$$

For file  $F_4$ :

$$FK_4^1 = \sum_{i=1}^m r_{i4}^1 D_i = 0 \times 5 + 0 \times 10 + 0 \times 3 = 0$$

$$FK_4^2 = \sum_{i=1}^m r_{i4}^2 D_i = 1 \times 5 + 0 \times 10 + 1 \times 3 = 8$$

$$FK_4^3 = \sum_{i=1}^m r_{i4}^3 D_i = 0 \times 5 + 1 \times 10 + 0 \times 3 = 10$$

$$\text{So, } FK_4 = (FK_4^1, FK_4^2, FK_4^3) = (0, 8, 10)$$

So we got the keys of files

$$FK_1 = (3, 15, 10); FK_2 = (0, 0, 8); FK_3 = (0, 10, 0); FK_4 = (0, 8, 10).$$

So, the user key table:

User	$UK_i$	$TU_i$
$U_1$	(0, 11, 10)	0
$U_2$	(0, 8, 11)	3
$U_3$	(5, 6, 10)	5

Figure: 3.3 User's key table with Time stamp

And File key table:

File	$FK_i$	$Tf_i$
$F_1$	(3, 15, 10)	1
$F_2$	(0, 0, 8)	2
$F_3$	(0, 10, 0)	4
$F_4$	(0, 8, 10)	6

Figure: 3.4 File key with Time stamp

### 3.2 Verification of access right

Suppose  $x$  is a positive integer such that  $x.w \bmod d = l$ . That means to compute  $x$  we have to use extended Euclid's algorithm [1]. To verify access right, the access right of the user  $U_i$  to the file  $F_j$  is computed by following algorithm.

Algorithm 3.2: Verification of access right

Input:  $UK_i$  or  $FK_j$ .

Output:  $r_{ij}$

Steps 1. Input  $UK_i$  or  $FK_j$ .

2. If  $TU_i > TF_j$  then

begin

For  $1 \leq z \leq c$  do

begin

compute  $UQ_i^z = UK_i^z \cdot x \bmod d$

$r_{ij}^z = \lfloor UQ_i^z / 2^{z-1} \rfloor \bmod 2,$

end for

end if

else

begin

For  $1 \leq z \leq c$  do

begin

compute  $FQ_i^z = FK_j^z \cdot x \bmod d$

$r_{ij}^z = \lfloor FQ_i^z / 2^{z-1} \rfloor \bmod 2,$

end for

end else

3. Output  $r_{ij}$ .

#### Example 3.2: Verification of access right

Suppose that the user  $U_2$  wants to write in the file  $F_3$ . Now we have to verify whether the user  $U_2$  is permitted to write in the file  $F_3$  or not. Since we have taken  $w = 5$  and  $d = 17$ , so we get  $x = 7$ . Here  $TU_2 = 3$  and  $TF_3 = 4$ . Since  $TU_2 < TF_3$ , we use equation

$$r_{ij}^z = \lfloor FQ_i^z / 2^{z-1} \rfloor \bmod 2, \text{ for } z = 1, 2, 3, \dots, c.$$

Here  $FK_2 = (0, 0, 8)$  thus we get

$$\begin{aligned} Q_2^1 &= FK_2^1 \cdot x \text{ mod } d = 0 \text{ mod } 17 = 0 \\ Q_2^2 &= FK_2^2 \cdot x \text{ mod } d = 0 \text{ mod } 17 = 0 \\ Q_2^3 &= FK_2^3 \cdot x \text{ mod } d = 56 \text{ mod } 17 = 5 \end{aligned}$$

and so,

$$\begin{aligned} r_{23}^1 &= \lfloor 0/4 \rfloor \text{ mod } 2 = 0 \\ r_{23}^2 &= \lfloor 0/4 \rfloor \text{ mod } 2 = 0 \\ r_{23}^3 &= \lfloor 5/4 \rfloor \text{ mod } 2 = 1 \\ \text{so } r_{23} &= (r_{23}^1 r_{23}^2 r_{23}^3) = (001) = 1 \end{aligned}$$

Since the system found  $r_{23} = 1$  (execute) and the write access is 3. So, the access request is denied. 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.

#### 4. Changing access right

Let access right  $r_{ij} = (r_{ij}^1 r_{ij}^2 \dots r_{ij}^c)$  is changed to  $s_{ij} = (s_{ij}^1 s_{ij}^2 \dots s_{ij}^c)$ . Here we have to update key for user  $UK_i$  or for files  $FK_j$ . The updated key can be obtained by using the following algorithm 5.1. It can be noticed that there will be no change in  $D_i$  or  $B_j$ .

Algorithm 4.1: Changing access right  
//  $D_i$  or  $B_j$  are computed as section 3.1

Input:  $UK_i$  or  $FK_j$ ,  $r_{ij}$  and  $s_{ij}$

Output: Updated  $UK_i$  or  $FK_j$

Steps: 1. Input  $UK_i$  or  $FK_j$ ,  $r_{ij}$  and  $s_{ij}$

2. If  $TU_i > TF_j$  then

begin

For  $1 \leq z \leq c$  do

begin

set  $ub_1 = s_{ij}^z$  and  $ub_2 = r_{ij}^z$

compute  $ut = ub_1 - ub_2$

If ( $ut \neq 0$ ) then

$UK_i^z = UK_i^z + ut \cdot B_j$

else  $UK_i^z$  is remained unchanged.

end for

end if

Else

begin

For  $1 \leq z \leq c$  do

begin

set  $fb_1 = s_{ij}^z$  and  $fb_2 = r_{ij}^z$

compute  $ft = fb_1 - fb_2$

If ( $ft \neq 0$ ) then

$FK_j^z = FK_j^z + ft \cdot D_i$

else  $FK_j^z$  is remained unchanged.

end for

end else

3. Output updated  $UK_i$  or  $FK_j$

#### Example 4.1: Changing access right

Suppose the access right  $r_{21} = 011$  changes to  $s_{21} = 010$ . From example 3.1 we have  $UK_2 = (0, 8, 11)$ . Here  $TU_i > TF_j$ , and  $B_1 = 5$ .

$$UK_2^1 = 0 \text{ (since } s_{21}^1 = 0, r_{21}^1 = 0, \text{ and } ut = 0)$$

$$UK_2^2 = 8 \text{ (since } s_{21}^2 = 1, r_{21}^2 = 1, \text{ and } ut = 0)$$

$$UK_2^3 = 11 - 5 = 6 \text{ (since } s_{21}^3 = 0, r_{21}^3 = 1, \text{ and } ut = -1)$$

So, new value of  $UK_2 = (0, 8, 6)$

Let see whether we can verify access right with updated values of keys. Suppose that we wish to compute  $r_{21}$ . We have key value  $UK_2 = (0, 8, 6)$ .

$$Q_2^1 = UK_2^1 \cdot x \text{ mod } d = 56 \text{ mod } 17 = 5$$

$$Q_2^2 = UK_2^2 \cdot x \text{ mod } d = 56 \text{ mod } 17 = 5$$

$$Q_2^3 = UK_2^3 \cdot x \text{ mod } d = 42 \text{ mod } 17 = 8$$

$$\begin{aligned}r_{21}^1 &= \lfloor 0/4 \rfloor \bmod 2 = 0 \\r_{21}^2 &= \lfloor 5/4 \rfloor \bmod 2 = 1 \\r_{21}^3 &= \lfloor 8/4 \rfloor \bmod 2 = 0\end{aligned}$$

$r_{21} = 010$  which is correct. If we want to verify access right with updated value of the key(s), the result will be correct.

#### **4.1 Addition of a user or a file**

When a user is inserted into the system, we assign a time stamp number as the time stamp of the user. Then the key value of the user is computed by equation (3.1). To insert a new file, the system assigns a time stamp number to the file and the key value of the file is computed by equation (3.2).

#### **4.2 Deletion of a user or a file**

The deletion process is very easy. When a user or a file is being deleted from the system, the key value and the time stamp of the user or file is deleted from the user (file) key table.

### **5 Discussion**

We assume that the system has  $m$  users and  $n$  files. By ignoring the overflow problem of key values, the space complexity of the proposed method is  $O(m + n)$  since each user or file possesses a key. For the time complexity of the method, we assume that each arithmetic operation needs  $O(1)$  time only. By equation (3.1) to construct a key of a user need to access all access rights of the files. Thus its time complexity is  $O(n)$ . Similarly time complexity to construct a key of a file is  $O(m)$ . By equation (3.2) to check access right of a user to a file need  $O(1)$  time. To delete a user or a file from the system only its corresponding entry is removed from the key tables, the time complexity is also  $O(1)$ . Here it may consider the number of modified key values

for each operation. It is easy to see that the proposed method updates key for the operations such as changing accesses right of a user to a file and inserting a user or a file to the system. On the other hand deletion process is very simple. So, the system achieves full dynamism. Here the following remarks can be highlighted:

1. To reveal the access right of a user to a file a simple operation on one key of a user or a file is enough.
2. To change the access right of a user to a file, it modifies only the key of the user or file.
3. To insert a new user/file into the system, the proposed method only constructs (assigns) a key to the user /file without modification of the other keys.
4. To delete a user or a file from the system, it simply removed the corresponding entry of the user or file from the user/file key table.
5. The method has special security feature due to two parameters  $x$  and  $d$ .
6. The method reduces overflow problem by breaking the key into its elements.

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). Therefore, we proposed a new access control method that achieves full dynamism as well as security of access data itself.

### **6 Conclusion**

In this paper a very simple and efficient user-file key access control scheme based on

binary access mode using time stamp is proposed. For the proposed method we devised formulas key construction process. Here algorithms for verification of access right and updating access right are also devised. The proposed scheme achieved full dynamism that means, changing access right, insertion and deletion of user/file can be implemented by performing operations on only a key. Space for the scheme is not large. The method achieves security of access data itself due secret parameters.

## References

- [1] D.E.R Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1983.
- [2] G.S Graham and P. J. Denning, "Protection Principle and Practice"; *Proc. Spring Joint Computer Conf., Vol.40*, AFIPS PRESS, Montavle, NJ, 1972, pp. 417-429.
- [3] J.K. Jan, "A Single Key Access Control Scheme in Information Protection System"; *Proceedings of National Computer Symposium*, Taiwan, 1987, pp. 299-303.
- [4] 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.
- [5] J.K. Jan, C. C. Chang and S. T. Wang, "A Dynamic Key-Lock-Pair Access Control Scheme", *Computers and Security*, Vol. 10, 1991, pp. 129-139.
- [6] M. S. Hwang, W.G. Tzeng and W. P. Yang, "A two-key-lock-pair access Control Method using prime Factorization and time stamp", *IEICE Trans. Information and System*, Vol. E77-D, No. 99, 1994, pp. 1042-1046.
- [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. R. Islam and M. N. M. Sap, "A Binary access control scheme with high data security", *Journal of Information Technology*, UTM, Vol. 14, No.2, Dec. 2002, pp. 8-16.