Malaysian Journal of Computer Science, Vol. 14 No. 2, December 2001, pp. 38-46

A BINARY ACCESS CONTROL SCHEME WITH LOGICAL AND PHYSICAL KEYS

Md. Rafiqul Islam
Computer Science and Engineering Discipline
Khulna University, Khulna, Bangladesh
email: cseku@khulnanet.net

Harihodin Selamat and Mohd. Noor Md. Sap
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
Jalan Semarak
54100 Kuala Lumpur, Malaysia
email: harihodn@itp.utm.my

ABSTRACT

An access control scheme for implementing the access control matrix is presented. The proposed scheme is based upon binary form of access rights and different from the schemes that are based on the concept of key-lock pairs. In this scheme, each user is assigned two keys. The first key is the logical key and is in binary form, and the second key is the physical key that holds access rights. The keys are possessed by the user, and can be used to derive the access rights to the files. The scheme has a special feature, such as a file or user can be added to or removed from the system without much effort. This scheme is more effective and efficient for systems where files are accessible to only a limited number of users.

Keywords: ***Access right, Dynamic access, Logical and Physical keys***

1.0 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 system can be used to prevent the information stored in a computer from being destroyed, altered, disclosed or copied by unauthorised users. Graham and Denning [2] in 1972 developed an abstract protection model for computer systems. The model is based upon access matrix - the state of a protection system. The state of a protection system is defined by a triple *(S, O, A)*, where:

1) *S* is a set of subjects (or processes),
2) *O* is a set of objects (or resources),
3) *A* is an access matrix, with rows and columns corresponding to subjects and objects respectively.

The subject could be users, processors, or utility programs. The objects could be files, tables of a database, storage segments, disks or magnetic tapes. Each element in the access control matrix represents the access right for a subject with respect to its corresponding object. The information protection for a computer system is achieved by employing an access control matrix, as depicted in Fig. 1. In this paper, we will use the terms user and file in place of subject and object respectively. The (*i, j*)th entry of the access control matrix is denoted as a_{ij}, which stands for the access right of the user U_i (*i*th user) to the file F_j (*j*th file). We assume that all access rights are expressed by numerals. Linear hierarchy of access privileges may be applied here. This means, the right to read implies the right to execute, the right to write implies the right to read and execute, and so on. In the access matrix shown in Fig. 1, 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	F_5
U_1	*2*	*1*	*0*	*3*	*0*
U_2	*1*	*0*	*3*	*0*	*4*
U_3	*0*	*4*	*5*	*0*	*3*
U_4	*3*	*0*	*0*	*4*	*0*

0 - no access
1 - execute
2 - read
3 - write
4 - delete
5 - own

Fig. 1: An access control matrix

Based on Graham and Denning's abstract protection model, Wu and Hwang [3] proposed an alternative scheme of storing just one key for each user and one lock for each file. To figure out access rights a_{ij}'s of users to files, a function f of key K_i and lock L_j is used. Mathematically,

$$a_{ij} = f(K_i, L_j) \qquad (1.1)$$

Several relevant access control methods [4, 5, 6, 7, 8, 9] appeared in the literature after Wu and Hwang's work. The previously proposed methods work well. However, on the dynamic access control problem such as adding or deleting files or users, the keys or locks should be re-established. Besides, the complexity of operations on generating keys and locks is sophisticated [13]. Hwang *et al.* in 1992 proposed a protection method using prime factorization [9]. In Hwang *et al.*'s method, since a lock is the product of some prime powers, it may easily exceed the largest integer allowed in a computer system. Chang *et al.* in [13] have improved the results of Hwang *et al.*'s scheme. Recently, Chang *et al.* [10] introduced a simple method with binary keys. The scheme requires reconstructing the whole system in cases of addition or deletion of a file. This means that the dynamism of addition or deletion of a file is not achieved by Chang *et al.*'s method.

In this paper, a new and simple access control called *logical and physical keys method* using binary form of access rights is proposed. The proposed method is for the implementation of sparse access control matrix. Verification of access right is simple. Dynamic access control such as changing access right, addition or deletion of a file or user, can easily be achieved.

2.0 THE LOGICAL AND PHYSICAL KEYS METHOD

Let $A[m, n]$ be an access control matrix, where m is the number of users, n is the number of files and a_{ij} is the (i, j)th entry in the access control matrix, A. This means, i represents the user number and j represents the file number in the system. Suppose each access right a_{ij} in the access matrix is represented in its binary form $a_{ij} = (a_{ij}^c a_{ij}^{c-1} \ldots a_{ij}^1)$ where, $c = 1 + \lfloor \log(a_{max}) \rfloor$, a_{ij}^1 is the first bit of access right a_{ij} and a_{max} is the maximum of access rights ($a_{max} = 5$ as for the access control matrix in Fig. 1). Thus, the access right a_{ij} can be expressed in the following form:

$$a_{ij} = (a_{ij}^c a_{ij}^{c-1} \ldots a_{ij}^1) = \sum_{z=1}^{c} a_{ij}^z \cdot 2^{z-1} \qquad (2.1)$$

where $a_{ij}^z \in \{0, 1\}$. For example from Fig. 1, we have $a_{23} = 3$ and $c = 1 + \lfloor \log(5) \rfloor = 3$. So, we can represent the access right a_{23} as 011 in binary form.

In the proposed method, each user is assigned two keys. The first one is the logical key and the second is the physical key that holds access rights of the user to the files. These keys are derived from access rights with respect to the files. The keys are possessed by the user and can be used to derive access rights to the files. From the first key, we can determine whether a user has an access to a file. The logical key is in binary form. Using the bits of the logical key from the physical key, one can find the access rights of the user to the files. Each user U_i is assigned two vectors, K_{iL} and K_{iP}. The logical key K_{iL} is defined as follows:

$$K_{iL} = K_{iL}^1 K_{iL}^2 \ldots K_{iL}^n \qquad (2.2)$$

for $i = 1, 2, \ldots, m$

where, $K_{iL}^x \in \{0,1\}$ and K_{iL}^x denotes the xth bit of the logical key K_{iL}.

Taking the value of K_{iL}^x as follows:

$$K_{iL}^x = \begin{cases} 0 \text{ if } a_{ij} \text{ is a zero bit-string}, \\ 1 \text{ if } a_{ij} \text{ is a non-zero bit-string} \end{cases} \qquad (2.3)$$

If the bit-string a_{ij} (binary form of access right a_{ij}) contains all zero bits such as $a_{ij} = 000$ (if $c = 3$), then a_{ij} is a zero bit-string. Otherwise, a_{ij} is a non-zero bit-string, *i.e.*, $a_{ij} \in \{001, 010, \ldots, 111\}$. From (2.3), it is clear that K_{iL} contains 1 in jth position ($K_{iL}^j = 1$) if user U_i has access (whatever it is, *i.e.*, $a_{ij} > 0$) to jth file. On the other hand, K_{iL} contains

0 ($K_{iL}^{j}=0$) if user U_i does not have access ($a_{ij} = 0$) to the jth file. So, from the logical key, one can easily determine whether a user has any access to a file.

Let a_{ij}^{z} be the zth bit of binary form of access right a_{ij}. The physical key of user U_i, is represented as $K_{iP} = \left(K_{iP}^{c}, K_{iP}^{c-1}, \quad \cdot, K_{iP}^{1}\right)$, *i.e.*, K_{iP} is a vector of c elements. Each element of K_{ip} is computed as follows:

$$K_{iP}^{z} = \sum_{e=1}^{p} a_{ij}^{z} \cdot 2^{e} \tag{2.4}$$

where, $e=\sum_{u=1}^{j} K_{iL}^{u}$, $p=\sum_{u=1}^{n} K_{iL}^{u}$ (K_{iL}^{u} denotes the value of uth bit of K_{iL})
and 1 £ z £ c, 1 £ j £ n, 1 £ i £ m.

Let $c = 3$, then the elements of the K_{iP} (the physical key of the ith user) can be computed as follows:

$$\begin{aligned} K_{iP}^{1} &= \sum_{e=1}^{p} a_{ij}^{1} \cdot 2^{e} \\ K_{iP}^{2} &= \sum_{e=1}^{p} a_{ij}^{2} \cdot 2^{e} \\ K_{iP}^{3} &= \sum_{e=1}^{p} a_{ij}^{3} \cdot 2^{e} \end{aligned} \tag{2.5}$$

If the user U_i (the ith user) wants to access the file F_j (the jth file), with the access mode r_{ij}, at first the system fetches the logical key K_{iL}. If the jth bit of K_{iL} is zero ($K_{iL}^{j}=0$), then the user does not have any access to the file F_j and the access request is denied. On the other hand, if the jth bit of K_{iL} is equal to one ($K_{iL}^{j}=1$), then the user has access to the file and the system fetches the second key K_{iP} (the physical key) of the user to verify the access right. We can check access right of a user according to the *algorithm 1*. If the requested access mode is equal to or less than the output result of *algorithm 1* (r_{ij}£ a_{ij}), then the access request is accepted. Otherwise, the access request is denied.

Algorithm 1 : ***Access right checking***

```
/* $K_{iL}$ denotes the logical key and $K_{iP}$ denotes the physical key of the $i$th user, $a_{ij}$ is the access right of the $i$th user to
the $j$th file, $c$ is the number of bits in $a_{ij}$ */
Input: $K_{iL}$ , $K_{iP} = \left(K_{iP}^{c}, K_{iP}^{c-1}, . . ., K_{iP}^{1}\right)$.
Output: $a_{ij} = ( a_{ij}^{c}\, a_{ij}^{c-1} \ldots a_{ij}^{1} )$.
Step 1: If $K_{iL}^{j}=0$ , then $a_{ij} = 0$;

Step 2: Else    /* i.e., If $K_{iL}^{j}=1$ */
      Begin
      compute $e=\sum_{u=1}^{j} K_{iL}^{u}$;
      For $1$ £ $z$ £ $c$ do
          Compute $a_{ij}^{z} = \left\lfloor \frac{K_{iP}^{z}}{2^{e}} \right\rfloor \bmod 2;$
          end_for;
   End;

Step 3: Output $a_{ij} = ( a_{ij}^{c}\, a_{ij}^{c-1} \ldots a_{ij}^{1} )$ or $a_{ij} = 0$.
```

Example 1: *Initialization of keys*
By considering the access control matrix in Fig. 1 and using binary form of access rights, we can find a binary access control matrix as shown in Fig. 2. Here, $a_{max} = 5$, then $c = 1 + \lfloor log(5) \rfloor$, *i.e.* $c = 3$.

Files / *Users*	F_1	F_2	F_3	F_4	F_5
U_1	*010*	*001*	*000*	*011*	*000*
U_2	*001*	*000*	*011*	*000*	*100*
U_3	*000*	*100*	*101*	*000*	*011*
U_4	*011*	*000*	*000*	*100*	*000*

Fig. 2: The binary access control matrix for Fig. 1

By considering access control matrix in Fig. 2, using (2.2), (2.3) and (2.4), we compute the logical and physical keys for users U_1, U_2, U_3 and U_4 as follows:

$K_{1L} = 11010$; $K^1_{1P} = 2^2 + 2^3 = 12$, $K^2_{1P} = 2 + 2^3 = 10$, $K^3_{1P} = 0$,

$K_{1P} = (K^3_{1P}, K^2_{1P}, K^1_{1P}) = (0, 10, 12)$;

$K_{2L} = 10101$; $K^1_{2P} = 2 + 2^2 = 6$, $K^2_{2P} = 2^2 = 4$, $K^3_{2P} = 2^3 = 8$,

$K_{2P} = (K^3_{2P}, K^2_{2P}, K^1_{2P}) = (8, 4, 6)$;

$K_{3L} = 01101$; $K^1_{3P} = 2^2 + 2^3 = 12$, $K^2_{3P} = 2^3 = 8$, $K^3_{3P} = 2 + 2^2 = 6$,

$K_{3P} = (K^3_{3P}, K^2_{3P}, K^1_{3P}) = (6, 8, 12)$;

$K_{4L} = 10010$; $K^1_{4P} = 2$, $K^2_{4P} = 2$, $K^3_{1P} = 2^2 = 4$,

$K_{4P} = (K^3_{4P}, K^2_{4P}, K^1_{4P}) = (4, 2, 2)$.

Therefore, $K_{1L} = 11010$, $K_{1P} = (0, 10, 12)$;
$K_{2L} = 10101$, $K_{2P} = (8, 4, 6)$;
$K_{3L} = 01101$, $K_{3P} = (6, 8, 12)$;
$K_{4L} = 10010$, $K_{4P} = (4, 2, 2)$.

Example 2: *Verification of access request*
Suppose it is required to verify the access right of the user U_2 to the file F_3, *i.e.* a_{23} = ?. First, the system fetches the logical key K_{2L} and then the physical key K_{2P} of the user U_2. Since $K^3_{2L} = 1$, by executing *algorithm 1*, we get $e = 2$, $a^1_{23} = \left\lfloor \frac{6}{4} \right\rfloor \bmod 2 = 1$, $a^2_{23} = \left\lfloor \frac{6}{4} \right\rfloor \bmod 2 = 1$, $a^3_{23} = \left\lfloor \frac{8}{4} \right\rfloor \bmod 2 = 0$. So, $a_{23} = \left(a^3_{23}\, a^2_{23}\, a^1_{23}\right) = 011 = 3$, which is correct. Again, suppose that we want to verify access right a_{42}. Since $K^2_{4L} = 0$, then $a_{42} = 0$, that is, the user U_4 does not have access to the file F_2. If the user U_1 sends a request to write (*2* in numeral) in the file F_2, (since $K^2_{1L} = 1$) by executing *algorithm 1*, we get $a_{12} = 001 = 1$. So the request is denied, since $a_{12}\ {}^{1}\ 2$.

2.1 Changing Access Right

Let us consider the access right a_{ij} (the access right of the *i*th user to the *j*th file) is changed to $b_{ij} = \left(b^c_{ij}\, b^{c-1}_{ij} \ldots b^1_{ij}\right)$. Here, we can consider the following three cases:

i) a_{ij} is zero bit-string ($K^j_{iL} = 0$) and *bij* is non-zero bit string
ii) a_{ij} is non-zero bit-string ($K^j_{iL} = 1$) and *bij* is zero bit-string
iii) a_{ij} is non-zero bit-string ($K^j_{iL} = 1$) and *bij* is also non-zero bit-string

For the first two cases, both keys (K_{iL} and K_{iP}) should be updated. However, any update in K_{iL} is very easy and can be performed just by resetting the respective bit of the K_{iL}. For the third case, the logical ke y K_{iL} remains unchanged and K_{iP} should be updated. In practice, the third case is more frequent than the other two. Using *algorithm 2*, we can perform the necessary update.

```
Algorithm 2 : Changing an access right
Input: $K_{iL}$, $K_{iP}$, $a_{ij}$ and $b_{ij}$.
Output: $K_{iL}$ (updated) and $K'_{iP}$.
Step 1: Compute $e = \sum_{u=1}^{j} K^{u}_{iL}$ ;

Step 2: If $( K^{j}_{iL} = 0$ and $b_{ij} \neq 0 )$ then
          begin
          set $K^{j}_{iL} = 1$ and update $K_{iL}$;
          $e = e + 1$ ;
            end_if;
         If $( K^{j}_{iL} = 1$ and $b_{ij} = 0 )$ then
            begin
               set $K^{j}_{iL} = 0$ and update $K_{iL}$;
                    $e = e - 1$;
            end_if;
         If $( K^{j}_{iL} = 1$ and $b_{ij} \neq 0 )$ then
               $K_{iL}$ remains unchanged;

Step 3: For $1 \leq z \leq c$ do
           begin
           set $t_1 = a^{z}_{ij}$ and $t_2 = b^{z}_{ij}$ ;
           compute $t = t_2 - t_1$;
                      If ($t \neq 0$) then
                         $K'^{z}_{iP} = K^{z}_{iP} + t.2^{e}$;
                      Else
                         $K'^{z}_{iP} = K^{z}_{iP}$;
        end_for;

Step 4: output $K_{iL}$ and $K'_{iP}$.
```

Example 3: *Changing access right*

Let the access right $a_{14} = 011$ is changed to $b_{14} = 101$. Here, $K_{1L} = 11010$, $K_{1P} = (0, 10, 12)$. By executing *algorithm 2*, we get, $e = 3$ and $K'^{1}_{1P} = K^{1}_{1P} = 12$, $K'^{2}_{1P} = K^{2}_{1P} - 2^{3} = 10 - 8 = 2$, $K'^{3}_{1P} = K^{3}_{1P} + 2^{3} = 0 + 8 = 8$, $K'_{1P} = (8, 2, 12)$.
Suppose $a_{43} = 000$ is changed to $b_{43} = 011$. Here, $K_{4L} = 10010$ and $K_{4P} = (4, 2, 2)$. So, by executing *algorithm 2*, we get $e = 2$, and $K_{4L} = 10110$ (by setting $K^{3}_{4L} = 1$); $K'^{1}_{4P} = K^{1}_{4P} + 2^{2} = 2 + 4 = 6$, $K'^{2}_{4P} = K^{2}_{4P} + 2^{2} = 2 + 4 = 6$, $K'^{3}_{4P} = K^{3}_{4P} = 4$, $K'_{4P} = (4, 6, 6)$.

If we verify the access rights of the users with the changing values of keys, we will get the correct results.

2.3 Adding File

Let the file F_q be added to the system. When a file is added to the system, the keys of the users should be updated. Suppose a_{iq} denotes the access right of the ith user to the file F_q (qth file). We can update K_{iL} by adding one bit to it. If the access right $a_{iq} = 0$, then the qth bit of K_{iL} is zero. Otherwise, the qth bit of K_{iL} will be 1. However, the elements of the physical key vector should be updated according to the value of the respective bit of a_{iq}. If the zth bit of a_{iq} is zero, then the respective element of the physical key remains unchanged. Otherwise, the element must be updated. *Algorithm 3* shows how we can update the keys of the users in case of a file addition.

Algorithm 3 : ***Adding file***
Input: Fq, access rights of the users to the file F_q.
Output: Updated keys of the users.

```
        Begin
        /* begin of the algorithm */
         For $1 £ i £ m$ do
         begin
           If ( $a_{iq} = 0$ ) then
             set $K_{iL}^{q} = 0;$
           Else
             set $K_{iL}^{q} = 1;$
          update $K'_{IL} = K_{iL}^{1} K_{iL}^{2} \ . \ . \ . \ K_{iL}^{n} K_{iL}^{q}$ ;
           If ( $K_{iL}^{q} = 1$ ) then
           begin
           compute $p = \sum_{u=1}^{n+1} K_{iL}^{u}$ ;
           For $1 £ z £ c$ do
              begin
                $t = a_{iq}^{z};$
                If ( $t ¹ 0$) then
                    $K'^{z}_{iP} = K_{iP}^{z} + t . 2^{p};$
                Else
                    $K'^{z}_{iP} = K_{iP}^{z}$ ;
             end_for;
          end_if;
        end_for;
    end_of_algorithm.
```

2.4 Deleting File

Let the file F_j be deleted from the system. When a file is deleted from the system, the logical key can be updated by deleting the respective bit of the key. The physical key should be updated according to the value of the respective bit of the access right of the user to the file. If the zth bit of the access right a_{ij} is zero, then the respective element of the physical key remains unchanged. Otherwise, the element should be updated. By performing the following *algorithm 4*, the system can update the keys of the users in case of a file deletion.

It is easily noticed that in case of an addition or a deletion of a file, the system updates the elements of the key vectors for $a_{ij}^{z} = 1$ of those users who can access the file. Any update in K_{iL} is very fast because it can be performed by binary shift operations.

Algorithm 4 : ***Deleting file***
Input: the keys of the users.
Output: updated keys of the users.

```
        Begin
        /* begin of the algorithm */
           For $1 £ i £ m$ do
              begin
              compute $e = \sum_{u=1}^{j} K_{iL}^{u}$ ;
           update $K'_{IL} = K_{iL}^{1} K_{iL}^{2} \ . \ . \ . \ K_{iL}^{j-1} K_{iL}^{j+1} \ . \ . \ . \ K_{iL}^{n}$ ;
           /* shift one bit left for $j$th bit to $n$th bit */
         If ( $K_{iL}^{q} = 1$ ) then
              begin
```

```
            $e = e - 1$;
              For $1 \le z \le c$ do
                  begin
                    $t = a_{iq}^{z}$;
                    If ( $t \neq 0$ ) then
                            $K_{iP}^{'z} = K_{iP}^{z} - t.2^{e}$;
                  Else
                            $K_{iP}^{'z} = K_{iP}^{z}$ ;
                end_for;
              end_if;
          end_for;
      end_of_algorithm.
```

2.5 Adding and Deleting Users

In the proposed scheme, the process of addition or deletion of a user is very simple. When a user is added, the system will construct the keys for the new user. If a user is deleted from the system, the system deletes the keys of the user.

3.0 COMPARISONS AND DISCUSSIONS

The effectiveness and efficiency of an access control scheme can be evaluated by considering the six criteria [11] which are as follows:

1) Effort for initialising keys and locks.
2) Effort for computing an access right from a lock and key.
3) Effort for revising keys and locks when an access right is modified.
4) Effort for appending and updating keys and locks when a new user or file is added.
5) Effort for removing and updating keys and locks when a user or file is deleted.
6) Space for storing keys and locks.

For the sake of comparison, we refer to Chang *et al.*'s access control scheme with binary keys [10] as Chang's 94 and Chang *et al.*'s binary access control method using prime factorisation [13] as Chang's 97. Since Chang's 94 and Chang's 97 and our scheme use the binary form of access mode, the efficiency of the schemes with respect to the above six criteria will be discussed.

In case of initialisation, Chang's 94 requires separating each bit of every access mode and construct key vectors for the users. There are *c* vectors for each key, *i.e.* there are *cm* ($cm = c \times m$) key vectors for *m* users [10]. Chang's 97 requires to select *m* keys and here *mc* multiplication are required to compute each lock element. There are *n* locks for *n* files [13]. In our method, we have to compute *m* logical keys and *cm* elements of physical keys. There are *cm* key elements for *m* users. Here the construction of the logical key can be performed by binary shift operations that are very fast. The construction process of the elements of physical key is simple. We need to consider the non-zero bits of the access right (*i.e.*, for $a_{ij}^{z} = 1$) and according to the formula devised in Section 2, we can construct the elements of the physical key. So, the computation of physical key is simple (for the system in which most files are only accessible to a few users, *i.e.* for sparse access matrix whose entries are mostly zeros). More important, such cases are common in the general time-sharing and multi-user database systems.

To find out an access right, Chang's 94 requires finding out one bit from each key vector and after combining *c* bits taken from *c* key vectors, we get the access right [10]. In Chang's 97, it requires *c* divisions [13]. Here, lock values are very large numbers, whereas keys are relatively small. Such division process takes time. Our scheme requires *2c* divisions and several binary shift operations to find an access right. In our method, the determination of whether a user can access a file or not is very fast. If a user does not possess any access right, we can find that very fast. Whereas, Chang's 94 and Chang's 97 require the computing of each of *c* bits even when there is no access right.

For changing an access right, Chang's 94 needs to update *c* key vectors of the user [10]. Chang's 97 needs to update c lock elements of the file [13]. Our method requires changing the elements of the physical key if $b_{ij}^{z} - a_{ij}^{z} \neq 0$ (when the access right a_{ij} is changed into b_{ij}).

When a user is added to the system, Chang's 94 and our system require constructing the key for the user. However, Chang's 97 needs to update the locks of the files that can be accessed by the user. For the deletion of a user, Chang's 94 as well as our method require deleting the key of the user. Whereas Chang's 97 requires updating the lock of the files that are accessible by the user.

When a file is added to the system, Chang's 94 needs updating the whole system [10]. Chang's 97 requires constructing the lock of the file [13]. Our system requires updating the elements of the key vectors for $a_{ij}^{z} = 1$ of those users who can access the file. For file deletion, Chang's 94 requires updating the whole system. Chang's 97 needs to delete the lock of the file. Our system requires updating the elements of the key vectors for $a_{ij}^{z} = 1$ of those users who can access the file.

The storage space required to implement Chang's 94 is $O(m)$. However, the storage for Chang's 97 is $O(mn)$. The storage required to implement our scheme is $mn + mc.\log_2(L_{max})$ bits, where L_{max} is the largest value among all elements of physical keys. This means the order of magnitude for storage is $O(2m) = O(m)$.

There are two other criteria that can be considered as the efficiency evaluating criteria for the access control schemes. They are:

a) Effort for finding the files that can be accessed by a particular user.
b) Effort for finding the users who can access a particular file.

For the first criteria, Chang' 94 and Chang's 97 require checking the access rights of the user to all the files. Whereas our scheme needs computing only *p* (*p* is defined in Section 2). For the second criteria, Chang's 94 and Chang's 97 need checking the access rights of all the users to the file. However, our scheme requires checking the bits of a particular position of the logical keys. If we wish to find out all the users who can access the file F_j (the *j*th file), we check the bits of *j*th positions of the logical keys.

Conclusively, our scheme is superior to Chang's 94 in case of file addition and file deletion. On the other hand, our scheme is superior to Chang's 97 scheme in terms of space requirement. The proposed scheme is superior to Chang's 94 and Chang's 97 for the efforts in finding files that can be accessed by a particular file and the users who can access a particular file.

4.0 CONCLUSION

In this paper, we have proposed a simple and efficient scheme based on binary access modes. In the proposed method, we devise algorithms for access right checking and for implementation of dynamic access control, such as changing access right and updating files. One good feature of our system is the insertion or deletion of any file can be successfully implemented without reconstructing all the key vectors. The storage required to implement our scheme is small. Excellence of the scheme is more pronounced for those systems where files are accessible to only a few users. This means that our new scheme is very suitable for implementation of a sparse access matrix. Based on the six criteria, the proposed scheme is a considerably better method for access control than the other comparable schemes. The merit lies in its simplicity in terms of the basic idea, the algorithms and space requirement.

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", in *Proceedings of the Spring Joint Computer Conf.,* Vol. 40, AFIPS Press, Montvale, 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 Systems", *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] 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 Transaction on Software Engineering*, Vol. 15, No. 9, 1989, pp. 1135-1137.

[8] 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.

[9] 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.

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

[11] J. C. R. Tseng and W.-P. Yang, "A New Access Control Scheme with High Data Security", in *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 Dynamic Access Control with Binary Key-Pair". *Malaysian Journal of Computer Science*, Vol. 10 No. 1, 1997, pp. 36-41.

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

[14] M. R. Islam, H. Selamat and M. N. M. Sap, "An Information Protection Scheme Using Logical and Open Keys", in *Proceedings of IRMA International Conference*, 1998, USA.

BIOGRAPHY

Md. Rafiqul Islam obtained his Master of Science in Engineering (Computers) from Azerbaijan Polytechnic Institute in 1987 and PhD. in computer science from Universiti Teknologi Malaysia in 1999. He is an Assistant Professor and Head (in-charge) of Computer Science and Engineering Discipline of Khulna University. His research areas include design and analysis of algorithms, information security (access control, cryptography, database security).

Harihodin Selamat holds a MSc from Cranfield University, UK and a PhD. 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. He holds a B.Sc. (Hons) from the National University of Malaysia, an MSc from Cranfield University, UK, and a PhD. from the University of Strathclyde, UK. He is currently carrying out research in database security, case-based reasoning and information retrieval.