A New Dynamic Access Control Scheme Based on Subject-Object List *

Min-Shiang Hwang [†‡]

Email: m24@twnmoctl.bitnet

Wei-Pang Yang [†0]

Email: wpyang@twnctu01.bitnet

Department of Computer and Information Science [†]

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

September 18, 2003

1 Introduction

It is a major issue to prevent important information from being destroyed, altered, disclosed or copied by unauthorized users in computer protection systems. For this reason, an access control model [5, 6] was introduced for its easy implementation and was widely used in computer protection systems.

In general, an information protection system has three elementary components: subjects, objects, and access rights. Subjects are the entities, such as users, processors, and utility programs, that perform access activities. Moreover, objects are the resources to be protected, such as disks, magnetic tapes, files, and memory segments. The access rights for a particular subject to a particular object include "execute", "read", "delete", "write", and "own". An access control matrix is a matrix that describes the access rights of subjects to objects in the system. It can be defined as follows.

Definition 1.1 *Let* $S = \{S_i | i = 1, 2, \ldots, m\}$ *be the set of subjects,* $O = \{O_j | j = 1, 2, \ldots, n\}$ *be the set of objects, and* $R = \{R_l | l = 1, 2, \ldots, r\}$ *be the set of access rights, where* m *is the total number of subjects;* n *is the total number of objects, and* r *is the total number of access rights in the system. Let the access function* $\mathcal{F}$ *be a function from* $S \times O$ *to* $\mathcal{P}(R)$*, where* $\mathcal{P}(R)$ *denotes all possible sets of elements of* R*.*

*This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no. NSC82-0408-E-009-161, and by the Telecommunication Laboratories Taiwan, R.O.C., under contract no. TL-NSC-82-5206.

†

‡

†

†

[0]Responsible for correspondence.

Then matrix $M = (S, O, \mathcal{F})$ is called the access control matrix.

Usually, the access control matrix is a sparse matrix. Storing it directly in the system requires a large amount of storage space and thus increases the cost of the system [4].

Many methods for reducing the storage space have been proposed. They can be divided into two classes: list-based and key-lock-pair-based access control schemes. Two types of the list-based access control scheme were proposed in the seventies: capability-list (C-list for short) scheme and accessor-list (A-list for short) scheme [5, 6]. In a C-list scheme, each subject has a list of objects to which it has various rights to access. In an A-list, each object has a list of subjects which have the right to access it. List-based methods have two main drawbacks: verifying access rights is extremely time-consuming, and the number of lists increases each time when new subjects or objects are added to the system. In the key-lock-pair-based access control schemes, the access right of a subject to an object is verified through an operation performed on the key and the lock. In the past decades, many key-lock-pair based schemes [1, 2, 3, 7, 8, 10, 11] were proposed. In these schemes, each subject possesses a key and each object has a lock. When a subject wishes to access an object, the scheme verifies the right of the subject to access the object by performing an operation on the subject's key and the object's lock.

None of the above schemes provide a convenient way of updating a few keys for each operation (e.g., adding a subject/object, removing a subject/object, or updating the access right of a subject to an object) while using a reasonable amount of storage space. In this paper, we propose a new access control scheme, the subject-object-list (SO-list for short), for a system which adds, removes, and updates subjects and objects frequently. The main advantage of the SO-list is that the number of lists remains constant despite of the changes in the number of subjects/objects. Furthermore, by applying the key-lock-pair (KLP for short) mechanism to the list, we allow a new subject or object to be added to the system without recomputing a large number of key/lock values. Thus, our method reduces both the number of keys to be updated for each operation and the amount of space needed to store keys and locks.

This paper is organized as follows. In the next section, we introduce the new concept of SO-list. In Section 3, we apply a new KLP method to SO-lists to create a new dynamic access control scheme. A simulation conducted to determine the storage space needed for key values is described in Section 4. Section 5 is the conclusion of this paper.

2 The Subject-Object-List (SO-list)

In this section, we describe the concept of the SO-list. This concept is analogous to that of a C-list and A-list, except that the number of SO-lists remains constant despite of the changes in the number of subjects/objects. A C-list identifies objects that can be accessed by a subject with an access right. Conversely, an A-list identifies subjects that can access a certain object, to which they have an access right. An SO-list identifies access rights in which subjects can access a certain object with rights.

For convenience, we give an example to illustrate the concept of the SO-list.

Example 2.1 *The access control matrix of a simple system with four subjects and five objects is shown in Figure 1. The SO-lists for* 5 *access rights* $(0, 1, 2, 3, 4)$ *are depicted in Figure 2. Since the access right of subject* S_2 *to object* O_3 *is 3, the list of access right* R_3 *has a node which contains* (S_2, O_3).

The node structure used in the above SO-list contains one subject-ID, one object-ID, and one next point, as shown in Figure 3.

In Figure 2, the list R_i denotes the access right of type i. For example, 0 for "no access", 1 for "execute", 2 for "read", 3 for "write", and 4 for "own". Since R_0

Subject	Object O_1	O_2	O_3	O_4	O_5
S_1	4	4	0	1	0
S_2	2	1	3	0	2
S_3	1	3	4	1	4
S_4	2	1	0	4	3

0 : no access

1 : execute

2 : read

3 : write

4 : own

Figure 1: The access control matrix for a simple system.

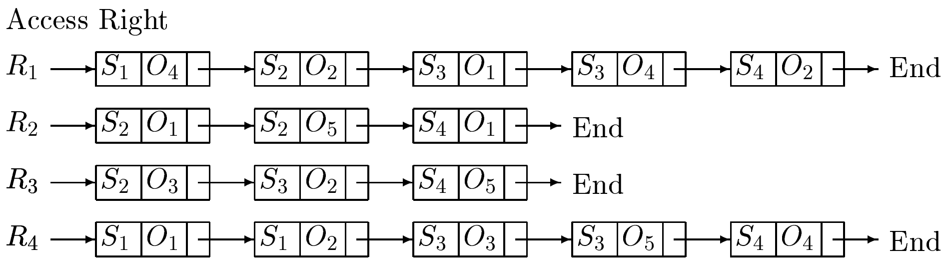

Figure 2: The SO-list scheme for the system in Figure 1.

Subject-ID	*Object-ID*	*Next Point*

Figure 3: The node structure of SO-lists.

denotes "no access" right of a subject to an object, we need not store R_0 list in the system.

We note that the number of SO-lists remains constant. Only four lists are needed, even when the number of subjects/objects in the system changes. However, the C-list and A-list do not have this property.

3 The KLP Method on SO-list

In this section, a KLP scheme on SO-lists is developed for a new dynamic access control system. We shall first introduce two lemmas and one theorem, which play important roles in the development of our scheme.

Lemma 3.1 *Let S_i be the ith ordered subject in list v, t_v be the length of the list v, and L_s be the lock value of subject, and $L_s > \max\{S_i \mid i = 1, 2, \cdots, t_v\}$. Then, there exists an integer K_{v_1}, called the first-key of access right v, such that S_i is equal to $\lfloor K_{v_1}/L_s^{i-1} \rfloor \bmod L_s$, where $\lfloor \cdot \rfloor$ indicates a floor function.*

The proof of Lemma 3.1 is presented in Appendix A. If S_i, L_s, and K_{v1} in Lemma 3.1 are replaced by O_j, L_o, and K_{v2}, respectively, we have the following lemma.

Lemma 3.2 *Let O_j be the jth ordered object in list v, t_v be the length of the list v, and L_o be the lock value of object, and $L_o > \max\{O_j \mid j = 1, 2, \ldots, t_v\}$. Then, there exists K_{v_2}, called the second-key of access right v, such that O_j is equal to $\lfloor K_{v_2}/L_o^{j-1} \rfloor \bmod L_o$.*

From Lemma 3.1 and 3.2, the following theorem is derived directly.

Theorem 3.1 *Let the definitions of $S_i, O_j, K_{v_1}, K_{v_2}, L_s, L_o$, and t_v be the same as those in Lemma 3.1 and Lemma 3.2. Then, there exists a triple of integers $K_v =$*

(K_{v1}, K_{v2}, t_v) *called the key value of access right* v *and some integers* p *such that* $S_i = \lfloor K_{v_1}/L_s^p \rfloor \bmod L_s$ *and* $O_j = \lfloor K_{v_2}/L_o^p \rfloor \bmod L_o$.

According to Theorem 3.1, we can construct a key value K_v of access right v as follows.

$$K_{v1} = \sum_{h=1}^{t_v} S_h L_s^{h-1}, \quad (1)$$

$$K_{v2} = \sum_{h=1}^{t_v} O_h L_o^{h-1}, \quad (2)$$

$$K_v = (K_{v1}, K_{v2}, t_v). \quad (3)$$

The following example illustrates the method of constructing keys.

Example 3.1 *Consider the access control matrix given in Figure 1 and Example 2.1. Instead of being a list shown as in Figure 2, each* R_i *is presented as set of triplets in here. That is* $R_i = \{(S_i, O_j, t)\}$*, where* S_i *denotes* i*th subject,* O_j *denotes* j*th object of the list, and* t *denotes the sequent order of triplets in the set. We obtain the following set of triplets:*

$$R_1 = \{(1,4,1),(2,2,2),(3,1,3),(3,4,4),(4,2,5)\},$$

$$R_2 = \{(2,1,1),(2,5,2),(4,1,3)\},$$

$$R_3 = \{(2,3,1),(3,2,2),(4,5,3)\},$$

$$R_4 = \{(1,1,1),(1,2,2),(3,3,3),(3,5,4),(4,4,5)\}.$$

Since the number of maximum subjects and objects in the above lists is 5. We assume, according to Lemma 3.1 and 3.2, the lock values of L_s *and* L_o *to be 7. Applying Equations (1)– (3), we construct the following key values for the four groups* R_1*,* R_2*,* R_3*, and* R_4*, respectively.*

$$K_1 = (10795, 6241, 5),$$

$$
\begin{aligned}
K_2 &= (212, 85, 3), \\
K_3 &= (219, 262, 3), \quad \textit{and} \\
K_4 &= (10788, 11481, 5).
\end{aligned}
$$

In the following subsections, we shall present algorithms based on our access control method for checking a subject's access right to an object, adding a subject/object to the system, removing a subject/object from the system, and changing a subject's access right to an object.

3.1 *Checking an access right*

To check the access right of subject i to object j, we first check the first-key K_{v1} in K_v. If there exist some integers p such that $S_i = \lfloor K_{v_1}/L_s^p \rfloor \bmod L_s$ is satisfied, we check the second-key K_{v2} in K_v with p. The algorithm for checking the access right of a subject to an object is given in Table 1.

Example 3.2 *Consider Example 3.1 again. If an access request* $(S_i, O_j, v) = (3, 2, 3)$ *occurs, the system fetches the key* $K_3 = (219, 262, 3)$ *and checks the request in the following steps.*

Step 1: *Since* $S_i = 3, K_{31} = 219,$ *and* $L_s = 7,$ *thus*

$$3 = \lfloor \frac{219}{7^p} \rfloor \bmod 7, 1 \leq p \leq t-1.$$

We have

$$p = 1.$$

Step 2: *Since* $O_j = 2, K_{32} = 262,$ *and* $L_o = 7,$ *thus*

$$\lfloor \frac{262}{7^1} \rfloor \bmod 7 = 2 = O_j.$$

Therefore, the access request $(S_3, O_2, 3)$ *is accepted.*

Table 1: Algorithm for checking an access right.

Input:	S_i, O_j; /* Subject i and object j. */
	v; /* The requested access right of subject i to object j. */
Output:	b; /* The Boolean operand which denotes acceptance or rejection. */
1.	Fetch $K_v = (K_{v1}, K_{v2}, t_v)$
	/* Checking first-key */
2.	Find p such that $S_i = \lfloor K_{v_1}/L_s^p \rfloor \bmod L_s$;
3.	if "found"
	/* Checking second-key for all p. */
4.	then if exists O_j such that
	$O_j = \lfloor K_{v_2}/L_o^p \rfloor \bmod L_o$
	then return (b=accepted)
5.	else return(b=rejected);

Table 2: Algorithm for adding a new subject to the system.

Input:	S_i; /* A new subject. */,
	$r_{ij}, 1 \leq j \leq n$, where n is the number of objects in the system.
	/* The access right of the new subject i to object j. */
Output:	K_v; /* The key value for access right v. */
	For $j = 1$ to n do;
	begin
	case $r_{ij} = v$:
	$K_v = (K_{v1}, K_{v2}, t_v)$;
	$K'_{v1} = K_{v1} + S_i L_s^{t_v}$;
	$K'_{v2} = K_{v2} + O_j L_o^{t_v}$;
	$t'_v = t_v + 1$;
	$K_v \longleftarrow (K'_{v1}, K'_{v2}, t'_v)$;
	endcase;
	end;

3.2 *Adding a New Subject/Object*

When adding a new subject to the system, we recompute the key values by computing $\sum_i S_i L_s^{t_v}$ and $\sum_j O_j L_o^{t_v}$, and then adding them to the old key values. The procedure for adding a new object to the system is similar to that for adding a new subject to the system. The algorithm for adding a new subject to the system is given in Table 2.

Example 3.3 *Consider Example 3.1. Let the subject to be added to the system be S_5 and the access rights to $O_1, \ldots, O_5$ be 0, 3, 3, 2, and 0, respectively. Thus, we need* *to recompute only K_2 and K_3. In Example 3.1, we have*

$$\begin{aligned} K_2 &= (212, 85, 3), \\ K_3 &= (219, 262, 3). \end{aligned}$$

According to the algorithm in Table 2,

$$\begin{aligned} K'_{21} &= K_{21} + 5 \times 7^3 = 1927, \\ K'_{22} &= K_{22} + 4 \times 7^3 = 1457, \\ t'_2 &= t_2 + 1 = 4. \end{aligned}$$

And,

$$\begin{aligned} K'_{31} &= K_{31} + 5 \times 7^3 + 5 \times 7^4 = 13939, \\ K'_{32} &= K_{32} + 2 \times 7^3 + 3 \times 7^4 = 8151, \\ t'_3 &= t_3 + 2 = 5. \end{aligned}$$

The new key values are as follows:

$$\begin{aligned} K_2 &= (1927, 1457, 4), \\ K_3 &= (13939, 8151, 5). \end{aligned}$$

Note that the order of the keys to be recomputed is only $O(r)$, where r is the number of types of access rights, which remains constant. In comparison, other methods in the literature [1, 2, 3, 7, 8, 10, 11] require $O(n)$, where n is the number of objects. Since $O(r) \ll O(n)$, our scheme is more effective and efficient.

3.3 *Removing a Subject/Object*

When a subject $S_i, 1 \leq i \leq m$, is removed from the system, we first locate which key values of the access rights are involved. Then, we locate all pairs (subject, object)

Table 3: Algorithm for removing a subject from system.

Input:	S_i; /* Subject i. */
Output:	K'_v; /* The new key values */
1.	Fetch $K_v = (K_{v1}, K_{v2}, t_v)$;
2.	Find p such that $S_i = \lfloor K_{v1}/L_s^p \rfloor \bmod L_s$;
3.	Compute O_j by $O_j = \lfloor K_{v2}/L_o^p \rfloor \bmod L_o$;
	/* Let (S_f, O_f) be the last pair of (subject,object) in R_v list */
4.	Find (S_f, O_f) such that $S_f = \lfloor K_{v1}/L_s^{t_v-1} \rfloor \bmod L_s$ and
	$O_f = \lfloor K_{v2}/L_o^{t_v-1} \rfloor \bmod L_o$;
5.	$K'_{v1} \longleftarrow K_{v1} - (S_i - S_f)L_s^p - S_f L_s^{t_v-1}$;
6.	$K'_{v2} \longleftarrow K_{v2} - (O_j - O_f)L_o^p - O_f L_o^{t_v-1}$;
7.	$t'_v \longleftarrow t_v - 1$;

with the same access right to be removed. The algorithm for removing a subject from the system is given in Table 3. The procedure for removing an object from the system is similar to that for removing a subject from the system. We emphasize again that the keys involved in removing a subject/object are the same as those in the case of adding a subject/object to the system. Therefore, only $O(r)$ keys need to be recomputed.

Example 3.4 *Consider Example 3.1 again. Suppose that subject 3 is the subject to be removed from the system. According to the algorithm in Table 3, only K_1, K_3, K_4 need to be recomputed, and the new key value K_1 is*

$$\begin{aligned} K'_{11} &= K_{11} - (3-4) \times 7^2 - 3 \times 7^3 - 4 \times 7^{(5-1)} = 211, \\ K'_{12} &= K_{12} - (1-2) \times 7^2 - 4 \times 7^3 - 2 \times 7^{(5-1)} = 116, \\ t'_1 &= t_1 - 2 = 3. \end{aligned}$$

The new key values $K_3 = (30, 38, 2)$ *and* $K_4 = (204, 211, 3)$ *are obtained in a similar way.*

3.4 *Changing an Access Right*

When an access right is changed from v to v', we have to carry out two tasks. The first is to recompute the key value of K_v. The second is to recompute the key value of $K_{v'}$. The procedures for these tasks are the same as those in the algorithm for removing a subject and adding a subject to the system, respectively.

Example 3.5 *Consider Example 3.1 again. Suppose that the access right of subject 3 to object 4 is changed from 1 to 3. Since the access right involved is* $\{1,3\}$, *only* K_1 *and* K_3 *need to be recomputed. As given in Example 3.1, their key values are*

$$\begin{aligned} K_1 &= (10795, 6241, 5), \\ K_3 &= (219, 262, 3). \end{aligned}$$

By applying the algorithm in Table 2 and 3, we have the following new key values:

$$\begin{aligned} K'_{11} &= K_{11} - (3-4) \times 7^3 - 4 \times 7^{5-1} = 1264, \\ K'_{12} &= K_{12} - (4-2) \times 7^3 - 2 \times 7^{5-1} = 753, \\ t'_v &= t_v - 1. \end{aligned}$$

And,

$$\begin{aligned} K'_{31} &= K_{31} + 3 \times 7^3 = 1248, \\ K'_{32} &= K_{32} + 4 \times 7^3 = 1634, \\ t'_3 &= t_3 + 1 = 4. \end{aligned}$$

Thus, the following new key values are obtained:

$$K_1 = (1264, 753, 4),$$

$$K_3 = (1248, 1634, 4).$$

4 Performance Simulation

Four algorithms have been introduced in Section 3 so as to verify a subject's access right to an object, adding a subject/object to the system, removing a subject/object from the system, and changing a subject's access right to an object. Additionally, the order of the number of keys to be recomputed has been demonstrated here to be only $O(r)$, where r is the number of types of access rights in the system which remains constant. Next, the time complexity of the key values K_i is investigated in the above four algorithms. Let $T(a^b)$ denote the number of multiplication needed to compute a^b. Knuth [9] indicated that $\lfloor \log_2 b \rfloor \le T(a^b) \le 2\lfloor \log_2 b \rfloor$ by using addition chain methods. Since the key value is defined in Equation (1) as $K_{v1} = \sum_{h=1}^{t_v} S_h L_s^{h-1}$, we have $T(K_{v1}) = \sum_{h=1}^{t_v} 1 + T(L_s^{h-1}) \le t_v + 2t_v \lfloor \log_2(t_v - 1) \rfloor = O(t_v \log_2 t_v)$, where t_v denotes the number of pairs $(subject, object)$ with the access right v. The time complexity of computing K_{v2} in Equation (2) is similar to that of K_{v1}. Therefore, the time complexity for computing the key values in Section 3 is upper bounded by $(t_v \log_2 t_v)$.

Next, the storage space required for the key values and lock values of the proposed scheme is considered. The lock value L_s in Equation (1) is equal to the number of subjects plus one. Similarly, the lock value L_o in Equation (2) is equal to the number of objects plus one. If the length of an integer is 32 bits, our scheme can accommodate $2^{32} \approx 10^9$ subjects and objects. Therefore, the storage of lock values is not a limitation in the new scheme.

Two key values using Equation (1) and Equation (2) are for $\sum_{h=1}^{t_v} S_h L_s^{h-1} \le \sum_{h=1}^{t_v} (L_s - 1) L_s^{h-1} \le \sum_{h=1}^{mn} (L_s - 1) L_s^{h-1} \le L_s^{mn} - 1$. Similarly, $\sum_{h=1}^{t_v} O_h L_o^{h-1} \le L_o^{mn} - 1$, where L_s and L_o are constants. Therefore, two key values in our scheme demand storage space of $O(mn)$. Properties of the key values in a practical implementation are further explored by conducting a simulation study on SunSPARC/SunOS R.4.1.1. Various numbers of subjects and objects were tested in the simulation. Results obtained from the case of $m = 1000$ subjects and $n = 1000$ objects are summarized in Figure 4. This figure illustrates the relationship between the ratio of nonzero entries in the access matrix and the ratio of storage space in our scheme with the original access matrix. The x-axis represents the percentage of nonzero entries and the y-axis represents the percentage of storage capacity. The various r_n (solid lines) represent the number of access rights in the computer protection system. The dashed line represents a constant, i.e., one hundred percent of the storage capacity. Using a smaller proportion of the storage capacity is preferable. The proportion of the storage capacity is smaller than one in Figure 4 in the following two cases. One is when the nonzero entry rate is less than 7%. The other is when $r_n \geq 5$ and the nonzero entry rate is less than 10%. Since r_n ranges from 5 to 8 and the nonzero entry rate is less than 10%, the space required to store key values in our scheme is less than that needed for the original access matrix. The following conclusions may be drawn from Figure 4. First, the sparser the access matrix implies that the more storage space is saved. Second, the larger the number of access rights used indicates that the less storage space is required in our scheme.

The nonzero entry rate of access control matrix is generally small in most computer systems [4]. Seven common types of access rights in computer systems are no access, read, write, modify, execute, delete, and own. Other types of access right may be extended in other application systems. For example, the following types of access right are extended in database systems, i.e., insert, remove, update, select, grant, and revoke, etc. The number of access rights is obviously increased gradually in the

Figure 4: Simulation results of storage space needed in the case of $m = 1000$ and $n = 1000$.

computer systems. Therefore, our scheme requires less storage space.

5 Discussions and Conclusions

We have introduced a new structure called SO-list in this paper. A SO-list is analogous to a C-list and A-list in presenting an access control matrix. The main difference between these structures is that the number of SO-lists remains constant despite of the changes in the number of subjects/objects. Furthermore, combining the SO-list with the KLP mechanism yields the following advantages:

1. The number of key values is equal to the number of access rights, which remains constant in computer systems. Therefore, our method need not to construct a new key value when adding a subject/object to the system.

2. Only a few key values need to be recomputed when a new subject/object is

added to or removed from the system or when the access right of a subject to an object is changed.

3. The sparser the access matrix is, the more storage space is saved. The larger the number of access rights used, the less storage space is needed in our scheme.

4. Because the KLP on SO-list stores only a few key values, it is more difficult to discover access rights of a subject to an object than those in a C-list and A-list. Therefore, this scheme is more secure than a C-list and A-list scheme.

5. The complexity of the storage space needed for keys and locks of r access rights is only $O(r)$ when the problem of overflow of key and lock values is ignored. Here r is a constant, the number of access rights in the system.

6. Since the number of SO-lists is constant (i.e., 4 in Figure 2), the scheme is easy to apply parallel processing for speeding up the computation of controlling access request and recomputing the key values when changing an access right, adding, and removing a new subject or object.

This scheme can also be applied to solve the user hierarchy problem. The hierarchy defined by [5, 6] is an ownership tree hierarchy of users. In general, a user hierarchy is presented by an H-matrix [11]. The lower triangle of the matrix alone is sufficient to present the hierarchy, and the upper triangle is regarded as undefined. In other words, the H-matrix denotes the relationships between users. Assume that we take the H-matrix to be an access control matrix by filling the upper triangle with zeros, our scheme can then be applied to present the user hierarchy.

However, there is still one weakness in our scheme: the lock values of subject (L_s) and object (L_o) are limited by the number of subjects and objects, respectively. Therefore, it is necessary to predict increases in the number of subjects and objects.

This situation is obviously unreasonable. To overcome this weakness, we can incorporate an extension of the concept of grouped keys [1] into our scheme.

Appendix A: Proof of Lemma 3.1 in Section 3

Proof. Let $K_{v_1} = \sum_{h=1}^{t_v} S_h L_s^{h-1}$. We need to prove

$$\lfloor K_{v_1}/L_s^{i-1} \rfloor \bmod L_s = S_i, 1 \le i \le t_v.$$

We have

$$\begin{aligned} & \lfloor K_{v_1}/L_s^{i-1} \rfloor \bmod L_s \\ = \; & \lfloor (S_{t_v} L_s^{t_v - 1} + S_{t_v - 1} L_s^{t_v - 2} + \cdots + S_{i+1} L_s^{i} + S_i L_s^{i-1} + \\ & S_{i-1} L_s^{i-2} + \cdots + S_1)/L_s^{i-1} \rfloor \bmod L_s \\ = \; & \lfloor (L_s Q + S_i + R) \rfloor \bmod L_s \end{aligned}$$

where $Q = S_{i+1} + S_{i+2} L_s + \cdots + S_{t_v} L_s^{t_v - i - 1}$, and $R = (S_{i-1}/L_s) + (S_{i-2}/L_s^2) + \cdots + (S_1/L_s^{i-1})$.

Since

$$\begin{aligned} & S_1 + S_2 L_s + \cdots + S_{i-1} L_s^{i-2} \\ < \; & L_s^{i-1}, \end{aligned}$$

this implies that

$$\lfloor R \rfloor = 0.$$

Since $L_s Q$ and S_i are integers, therefore

$$\begin{aligned} & \lfloor (L_s Q + S_i + R) \rfloor \bmod L_s \\ = \; & (L_s Q + S_i + \lfloor R \rfloor) \bmod L_s \\ = \; & S_i. \end{aligned}$$

□

Acknowledgments

We thank Dr. Wen-Guey Tzeng for their contributions to the paper. We wish to thank many anonymous referees for their suggestions to improve this paper. This research was funding grant NSC82-0408-E-009-161 and TL-NSC-82-5206 from the National Science Council and Telecommunication Laboratories, Taiwan, R.O.C., respectively.

References

[1] C. C. Chang and C. P. Chen, A key-lock-pair mechanism based on generalized Chinese remainder theorem, *Journal of the Chinese Institute of Engineers* 9(4) (1986) 383-390.

[2] C. C. Chang, An information protection scheme based upon number theory, *The Computer Journal* 30(3) (1987) 249-253.

[3] C. K. Chang and T. M. Jiang, "A binary single-key-lock system for access control," *IEEE Transactions on Computers* 38(10) (1989) 1462-1466.

[4] R. W. Conway, W. L. Maxwell and W. C. Morgan, On the implementation of security measures in information system, *Commun. ACM* 15(4) (1972) 211-220.

[5] D. E. R. Denning, *Cryptography and Data Security,* (MA:Addison- Wesley, 1982).

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

[7] J. J. Hwang, B. M. Shao and P. C. Wang, A new access control method using prime factorisation, *The Computer Journal* 35(1) (1992) 16-20.

[8] J. K. Jan, C. C. Chang and S. J. Wang, A Dynamic key-lock-pair access control Scheme, *Computers & Security* 10 (1991) 129-139.

[9] D. E. Knuth, *"The Art of Computer Programming Vol.2: Seminumberical Algorithm,* Addison-Wesley, Reading, MA. 1969.

[10] 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 Trans. Software Eng.* 15(9) (1989) 1135-1137.

[11] M. L. Wu and T. Y. Hwang, Access control with single-key-lock, *IEEE Trans. Software Eng.* 10(2) (1984) 185-191.