# 一种新的单钥-锁对访问控制方案

卿 利* 朱清新

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】针对现有单钥-锁对访问控制方案存在严重的溢出问题，基于整数二进制表示的唯一性，提出了一种新的单钥-锁对访问控制方案。新的方案实现简单，除了具有一般单钥-锁对方案的良好动态特性外，还在不需要访问权限递增假设下，实现了用户对文件的多种访问控制权限，并大大减小了溢出问题的发生。

关 键 词 访问控制; 二进制; 单钥-锁对; 溢出问题; 访问权限

中图分类号 TP309.2 文献标识码 A

# New Single-Key-Lock-Pair Access Control Scheme

Qing Li Zhu Qingxin

(School of Computer Science and Engineering, UEST of China Chengdu 610054)

**Abstract** To overcome the shortcoming of single-key-lock-pair access control scheme at present, a new single-key-lock-pair access control scheme is proposed, based on the property that a integer can be denoted into only one binary digital. All the single-key-lock-pair access control scheme are threaten by overflow problem. Besides that, under old scheme, user can own several kinds of right on one file only based on the supposition that the access rights is increase by degrees. With perfect dynamic feature, the new scheme is simply realized. Under new access control scheme, user can own several kinds of access right upon one file under the new scheme without the supposition that the access rights is increase by degrees, and the possibility of overflow problem is significantly reduced by our new method.

**Key words** access control; binary; single-key-lock-pair; overflow problem; access right

随着计算机网络的发展，由分布式环境提供的远程访问和资源共享给人们的学习、生活和工作带来极大的便利，但同时，共享资源也受到非法使用的威胁。为了防止计算机系统中存储的信息受到非授权用户的破坏、篡改、泄漏和复制，通常使用信息保护系统来对用户的访问进行控制，一个访问控制系统一般包括主体、客体和访问权限三个组件[1]。

通过存取控制矩阵可以实现一般的访问控制[2]。矩阵的每一个元素 $r_{ij}$ 表示主体 $U_i$ 对于客体 $F_j$ 所拥有的访问权限。当用户 $U_i$ 以访问权限 $r_{ij}'$ 提出对文件 $F_j$ 进行访问时，系统在存取控制矩阵中找到 $r_{ij}$，若 $r_{ij}' = r_{ij}$，则准许访问，否则拒绝用户要求。这种方法的缺点是需要的存储空间大，当存取控制矩阵是稀疏矩阵时，空间的利用率非常低，而且查找速度慢。

文献[3]提出使用单钥-锁对(Single-Key-Lock-Pair, SKL)表来实现访问控制系统，给每个主体 $U_i$ 分配一个钥向量 $\boldsymbol{K}_i$，每个客体 $F_j$ 分配一个锁向量 $\boldsymbol{L}_j$，当用户请求访问文件时，SKL系统利用对应的主体钥向量和客体锁向量通过运算验证用户对文件的访问权。文献[4,5]提出基于中国剩余定理和欧拉定理的单钥-锁对方案，文献[6]使用牛顿插值多项式来实现访问控制系统，文献[7]应用素因子分解定理获得一种SKL方案，文献[8]在此基础上改进为用二进制实现，文献[9]对文献[3]的方法加以改进。单钥-锁对访问控制系统具有很

2002年10月14日收稿

* 男 30岁 博士生 主要从事计算机网络与信息安全方面的研究

好的动态特性，但所有单钥-锁对访问控制系统都存在溢出问题，这使许多单钥-锁对方案不具有实际可行性，或适用范围十分狭窄。此外，现有的钥-锁对访问控制方案在实现用户对文件的多种访问权限时[3~9]，均假设权限具有递增关系，即高权限包含低权限，使访问控制系统不适合对文件进行细粒度保护。

本文基于整数二进制表示的唯一性提出一种新的单钥-锁对访问控制方案，在一定程度上缓解了溢出问题，发生溢出的可能性小于现有的方案[3~9]，而且实现简单，有很好的动态特性。在不增加计算复杂度的基础上，实现了用户对同一文件的多种访问权限，并且不需要访问权限递增假设，提出了彻底解决溢出问题的方向。

# 1　新的单钥-锁对访问控制方案

## 1.1　基本机制

在新的单钥-锁对访问控制系统中，每个用户有一个钥值$K$，每个文件有一个锁向量$\boldsymbol{L}$。考虑到整数在分解为二进制表示的唯一性，如整数可以唯一表示为

$$a=\sum_{i=1}^{b}a_i 2^{i-1}$$

式中　$b$为$a$的最大二进制位数，$a_i\in\{0,1\}$，$i=1,2,3,\cdots,b$。

设$\boldsymbol{A}_{m\times n}$是一个存取控制矩阵，其中$m$为用户数，$n$为文件数，矩阵$\boldsymbol{A}_{m\times n}$的元素$(i,j)$代表用户$U_i$对文件$F_j$的访问权限$r_{ij}$，$r_{ij}\in\{0,1,2,\cdots,b\}$，$b$为访问权限的种类数，即有$b$种访问权限，每一种访问权限用$0\sim b$的整数表示，当$r_{ij}=0$时表示无访问权。

每一个用户按照加入系统的先后，从整数栈中弹出一个整数作为用户钥值。由于访问权限的种类一般是有限的，即$b$值通常较小且固定，可以按照如下方式计算文件$F_j$的锁向量$\boldsymbol{L}_j=(L_{j1},L_{j2},\cdots,L_{jb})$

$$L_{jx}=\sum_{i=1,r_{ij}=x}^{n}2^{K_i-1}\qquad\text{对所有现有用户 } i\ \ (x=1,2,\cdots,b;\ \ j=1,2,\cdots,n)\tag{1}$$

式中　$K_j^{'}$和$K_i$分别是文件$F_j$和用户$U_i$的钥值。下面用例子说明用户钥锁表和文件钥锁表的构造。

例1　考虑一个文件系统，其存取控制矩阵如表1所示，有5个用户$U_1,U_2,\cdots,U_5$和5个文件$F_1,F_2,\cdots,F_5$，按照上面的方法，从整数栈中依次为每个加入的用户弹出一个整数作为该用户的钥值，即$K_1=1,K_2=2,\cdots,K_5=5$，然后按式(1)计算文件锁向量，得：$\boldsymbol{L}_1=(4,10,0,1)$；$\boldsymbol{L}_2=(10,0,17,4)$；$\boldsymbol{L}_3=(0,20,0,2)$；$\boldsymbol{L}_4=(16,9,4,0)$；$\boldsymbol{L}_5=(0,8,1,2)$；

如要计算文件$F_4$的锁向量$\boldsymbol{L}_4$，由式(2)计算的$\boldsymbol{L}_4$各个分量如下：$L_{41}=2^{5-1}=16$；$L_{42}=2^{1-1}+2^{4-1}=9$；$L_{43}=2^{3-1}=4$；$L_{44}=0$。

表1　存取控制矩阵

| 用户 | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ |
|---|---|---|---|---|---|
| $U_1$ | 4 | 3 | 0 | 0 | 3 |
| $U_2$ | 2 | 1 | 4 | 2 | 4 |
| $U_3$ | 1 | 4 | 2 | 3 | 0 |
| $U_4$ | 2 | 1 | 0 | 2 | 2 |
| $U_5$ | 0 | 3 | 2 | 1 | 0 |

注　0：无访问权；1：可读；2：可写；3：可执行；4：所有权限

## 1.2　验证访问权限算法

如果用户$U_i$希望以权限$r_{ij}=t,t\in\{1,2,\cdots,b\}$访问文件$F_j$，系统执行Check_Access_Right算法，作出访问控制判断，返回1接受，返回0拒绝。

Check_Access_Right算法如下：

输入：$K_i,\boldsymbol{L}_j=(L_{j1},L_{j2},\cdots,L_{jb}),r_{ij}=t$

/*用户$i$钥值，文件$j$锁向量，用户$i$请求文件$j$的访问权$r_{ij}$*/

**输出：** $r$

1) $r \leftarrow [L_{jt} / 2^{K_i-1}] \bmod 2$

2) return $r$

**证明** 假设用户 $i$ 对文件 $j$ 有访问权 $r_{ij}=t$，由式(1)有

$$L_{jt} = \sum_{j=1, r_{ij}=t}^{n} 2^{K_i-1}$$

则

$$r = [L_{jt} / 2^{K_i-1}] \bmod 2 = 1$$ **证毕**

**例2** 考虑例1的情形，如果用户 $U_2$ 要以权限4访问文件 $F_3$，则 $r = [L_{34} / 2^{K_2-1}] \bmod 2 = [2/2^{2-1}] \bmod 2 = 1$，输出结果为1，系统接受访问要求；如果用户 $U_5$ 要以权限4访问文件 $F_4$，则 $r = [L_{44} / 2^{K_5-1}] \bmod 2 = [0/2^{5-1}] \bmod 2 = 0$，输出结果为0，系统将拒绝访问要求。

### 1.3 插入和删除新用户算法

假设有新用户要加入系统中，从整数栈弹出一个整数作为用户钥值，然后重新计算所有文件锁向量。

Insert_User算法如下：

**输入：** $i$, /*新用户 $i$ */

$r_{ij}$, $1 \le j \le n$ /*用户 $i$ 对文件 $j$ 的访问权限，$n$ 是系统中当前文件数*/

**输出：** $\boldsymbol{L}_j$, $j = 1,2,\cdots,n$ /*所有的文件锁向量*/

1) $K_i \leftarrow$ 整数栈的栈顶元素

2) for $j=1$ to $n$ do

$L_{jr_{ij}} \leftarrow L_{jr_{ij}} + 2^{K_i-1}$ /*计算文件锁向量的每个分量*/

3) return $\boldsymbol{L}_j \leftarrow (L_{j1}, L_{j2}, \cdots, L_{jb})$, $j = 1,2,\cdots,n$

当从系统中删除用户 $U_i$ 时，需要重新计算所有的文件锁向量，并将用户钥值压回到整数栈中以供新加入的用户使用，然后删掉用户行。

Delete_User算法如下：

**输入：** $i$, /*要删去的用户 $i$ */

$r_{ij}$, $1 \le j \le n$ /*用户 $i$ 对文件 $j$ 的访问权限，$n$ 是系统中当前文件数*/

**输出：** $\boldsymbol{L}_j$, $j = 1,2,\cdots,n$ /*所有的文件锁向量*/

1) 将 $K_i$ 压回整数栈

2) for $j=1$ to $n$ do

$L_{jr_{ij}} \leftarrow L_{jr_{ij}} - 2^{K_i-1}$ /*计算文件锁向量的每个分量*/

3) return $\boldsymbol{L}_j \leftarrow (L_{j1}, L_{j2}, \cdots, L_{jb})$, $j = 1,2,\cdots,n$

从以上算法可见，在加入和删去用户时，只需改变文件锁向量的部分项。

### 1.4 增加和删除文件算法

当插入新文件时，只需按照式(1)计算新文件的锁向量即可，不必改变其他的文件锁向量。如果系统要删去文件，直接删掉文件对应的锁向量。

### 1.5 改变访问权限算法

当用户 $U_i$ 对文件 $F_j$ 的访问权限 $r_{ij}$ 从 $x$ 变到 $y$ 时，系统使用用户钥值 $K_i$ 重新计算文件锁向量 $\boldsymbol{L}_j$，计算时只需改变向量 $\boldsymbol{L}_j$ 的第 $x$ 个分量和第 $y$ 个分量，如

$$\begin{cases} L_{jx} \leftarrow L_{jx} - 2^{K_i-1} \\ L_{jy} \leftarrow L_{jy} + 2^{K_i-1} \end{cases} \tag{2}$$

**例3** 仍然考虑例1的情形，假设用户 $U_3$ 对文件 $F_4$ 的访问权限 $r_{34}=3$ 改变为2，由式(2)可计算 $L_{43} = 4 - 2^{3-1} = 0$，$L_{42} = 9 + 2^{3-1} = 13$，于是 $\boldsymbol{L}_4 = (16,13,0,0)$。

### 1.6 增加和减少访问权限算法

新的访问控制方案可以允许一个用户对一个文件有多种访问权限。为了对文件进行细粒度的保护，本文将每一种访问权限区分开来，如用户对文件同时拥有写权限和复制权限，这两种权限并不具有明显的包含关系，简单的将复制权限看成比写权限更高的访问权限不利于对文件的保护。现有的钥-锁对访问控制方案，为了实现用户对文件的多种访问权限，均假设访问权限集为全序集，即各权限按照权力大小由低到高递增，新的访问控制方案则不需要这种假设。

当用户 $U_i$ 增加一种对文件 $F_j$ 的访问权限 $x$，使用用户钥值 $K_i$ 重新计算文件锁向量 $\boldsymbol{L}_j$，计算时只需改变向量 $\boldsymbol{L}_j$ 的第 $x$ 个分量，如

$$L_{jx} \leftarrow L_{jx} + 2^{K_i - 1} \tag{3}$$

当用户 $U_i$ 减少一种对文件 $F_j$ 的访问权限 $y$，可按下式计算向量 $\boldsymbol{L}_i$ 的第 $y$ 个分量

$$L_{jy} \leftarrow L_{jy} - 2^{K_i - 1} \tag{4}$$

例4　考虑例1情形，如果用户对文件的访问权限增加了(见表2)，用户 $U_2$ 增加对文件 $F_2$ 的权限3，可计算 $F_2$ 的锁向量 $\boldsymbol{L}_2$，只需要由式(3)改变 $L_{23}$ 即可，$L_{23} = 17 + 2^{2-1} = 19$，得到 $\boldsymbol{L}_2 = (10,0,19,4)$。又如，用户 $U_5$ 增加对文件 $F_2$ 的权限2和1，则 $L_{22} = 0 + 2^{5-1} = 16$，$L_{21} = 10 + 2^{5-1} = 26$，得到 $\boldsymbol{L}_2 = (26,16,19,4)$，当用户 $U_5$ 减少对文件 $F_2$ 的权限1，只需按式(4)计算，$L_{21} = 26 - 2^{5-1} = 10$，得到 $\boldsymbol{L}_2 = (10,16,19,4)$。

表2　用户对文件多权限访问控制矩阵

| 用户 | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ |
|---|---|---|---|---|---|
| $U_1$ | 4,2 | 3 | 0 | 0 | 3 |
| $U_2$ | 2 | 1,3 | 4 | 2 | 4,1 |
| $U_3$ | 1 | 4 | 2,3 | 3 | 0 |
| $U_4$ | 2,1 | 1 | 0 | 2,4 | 2 |
| $U_5$ | 0 | 3,2,1 | 2 | 1 | 0 |

## 2 新访问控制方案的性能分析

### 2.1 时间复杂度

设系统中有 $m$ 个用户和 $n$ 个文件，假设一次数学运算的时间复杂度为 $O(1)$。当加入一个新用户到系统时，根据式(1)来计算用户锁向量，所进行的乘积运算为 $m(m-1)/2$ 次，加法运算为 $(m-1)$ 次，因此时间复杂度为 $O(m^2)$。当改变用户访问权限时，由式(2)重新计算锁向量，时间复杂度为 $O(m)$。当验证用户对文件的访问权限时，时间复杂度也为 $O(m)$。当删除用户时，时间复杂度为 $O(m)$，删除文件的复杂度为 $O(1)$。从时间复杂度上看，由于使用了指数运算，新访问控制方案相比于其他方案，其运算时间略长一些，但对于文件服务器，其运算量则非常小。

### 2.2 空间复杂度

设系统中有 $m$ 个用户和 $n$ 个文件，有 $b$ 种访问权限(不包括权限值为0，即无访问权限情形)，$l_{\max}$ 为文件锁向量所具有的最大长度，由式(1)有，$l_{\max} < \log_2 2^n = n$，所有锁向量存储需求为 $O(l_{\max} bm) = O(mn)$。在空间复杂度上新方案比文献[7]方案的空间复杂度 $O(n^2 \log n + m^2 \log m)$ 要好得多，而与文献[8]方案的空间复杂度相当。

### 2.3 溢出问题

设机器处理的整数长为 $w$，根据式(1)，当 $L_{jx} > 2^w$ 时，访问控制方案将会发生溢出。所有的钥-锁对访问控制方案都存在溢出问题，这使得许多方案没有实用性。其他的钥-锁对方案在计算用户(文件)钥值或锁值时，一般都涉及到不同素数的乘积[3~9]，素数乘积趋于 $2^w$ 的速度非常快，因而存在严重的溢出问题。在新的访问控制方案中，计算锁向量所用到的乘法是最小素数2的幂运算，所以新方案已经将溢出问题减轻到最低，但溢出问题仍然存在，解决的方法是对文件或用户进行分组，用户对每一组文件拥有一个钥值，这将是下一步的工作。

(下转第721页)

```
POP A                    ；丢弃PC压入堆栈的错误地址
POP A
MOV A，00H               ；将主程序的起始地址送入堆栈
PUSH A
MOV A，00H
PUSH A
RETI                     ；中断返回
```

## 3　结 束 语

单片机应用系统要正常工作，抗干扰设计是非常重要的。本文对由单片机系统内部和外部产生的干扰进行了讨论，并对不同的通道利用光电隔离的方法进行了论述，同时还提出了利用定时中断抗干扰的软件编制技巧。这对提高单片机应用系统的可靠性是很有效的。抗干扰的措施还很多，如软件编制时可采用“中值滤波”或“平均值滤波”等，相关的问题有待进一步的讨论。

### 参 考 文 献

[1] 孙涵芳. Intel 16位单片机[M]. 北京: 北京航空航天大学出版社, 1995

[2] 武庆生, 仇　梅. 单片机原理与应用[M]. 成都: 电子科技大学出版社, 1998

编　辑　漆　蓉

---

(上接第695页)

## 3　结　论

在访问控制系统中，用单钥-锁对实现方案可以获得良好的动态特性，如增加文件、删除文件和改变访问权限等操作都不必重建整个系统。本文提出的新单钥-锁对访问控制方案除了具备一般单钥-锁对方案的动态特性之外，还具有很多优点，如改变用户对文件的访问权限，只需修改该文件锁向量的部分项。在不需要访问权限递增假设下，实现了用户对文件的多种访问权限，新方案还极大地降低了溢出问题发生。

### 参 考 文 献

[1] Graham G S, Denning P J. Protection-principle and practice[C]. In Proc. AFTPS 1972 SJCC, 1972, 40: 417-429

[2] Denning D E R. Cryptography, Security D[M]. Addison-Wesley, Mass., 1982

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

[4] Chang C C. On the design of a key-lock-pair mechanism in information protection systems[C]. BIT, 1986, 26: 410-417

[5] Chang C C. An information protection scheme based upon number theory[J]. Computer J., 1987, 30(3): 249-253

[6] Laih C S, Harn L, Lee J Y. On the design of a single-key-lock mechanism based on Newton’s interpolating polynomial[J]. IEEE Trans. Software Eng., 1989, 15(9): 1 135-1 137

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

[8] Chang, C C, Lou D C. A binary access control method using prime factorization[J]. Information Science, 1997, 96(1-2): 15-26

[9] Chang C C, Jan J K. An access control scheme for new user and files[J]. International Journal of Policy Information, 1988, 12(2): 89-98

编　辑　徐培红