

# 软件加密与解密技术及其应用

## 实例专辑

中国科学院成都计算机应用研究所情报室

## 前 言

随着计算机的广泛应用，高技术不断地相互渗透，人们对软件及数据保护已提到议事日程，为了使计算机应用工作者尽快地掌握熟悉软件加密与解密技术知识，我们曾在1987年编了《微型、小型计算机操作系统应用技术与软件加密技术汇编》一书，受到了广大读者的欢迎，为使软件加密与解密技术资料内容更加丰富、系统、实用性更强。我们又大量搜集了近期最新资料，编制成《软件加密与解密技术及其应用实例专集》一书。该书是对前一套书下册的一个补充，以大量应用实例为主，具有较高的实用价值，是一本难得的内部参考性的情报资料。我们希望它能成为广大计算机应用工作者，获取成果捷径不可少的工具。

该书主要内容有：加密与解密基本原理与技术、方法与实践、文件保护与恢复等。在资料的汇集中，因时间关系未一一征求作者的意见，有不妥的地方，敬请作者原谅，并对作者积极支持此工作表示衷心感谢。

由于水平有限及时间关系，搜集的资料还不很完全，错误难免，敬请读者指正。

该书在编审的过程中受到了高树清、吴浣尘、罗淳、孟晓玲、刘素琴、徐敏如、张薇薇、梁军、郭俊茹等同志的支持表示感谢。

该书由杨明芳、周永培统编。

编者

宁云才

88, 7.20

# 《软件加密与解密技术及其 应用实例专集》

## 目 录

### 原 理 与 技 术

计算机与现代密码技术.....	( 1 )
数据加密的体制及应用.....	( 12 )
微型计算机应用系统的安全技术.....	( 16 )
密钥管理探讨.....	( 20 )
提高微机数据库信息安全保密性的措施.....	( 24 )
计算机信息保密方法简述.....	( 30 )
一些加密技术.....	( 32 )
FA 密码体制在数据完整性保护中的应用.....	( 36 )
磁盘如何“加锁”？.....	( 44 )
磁盘文件的简单加密与解密.....	( 50 )
IBM—PC 微机磁盘防拷贝和信息加密技术.....	( 58 )
微型机磁盘信息保密探讨.....	( 62 )
谈谈软磁盘加密.....	( 67 )
怎样判定加密算法的和谐性.....	( 70 )
数据加密/解密器 ( DEU ) Intel 8294A.....	( 72 )
软件的加密.....	( 78 )
计算机信息安全保密及安全保密通讯处理机.....	( 82 )
关于运行程序的加密与反加密.....	( 89 )

### 方 法 与 实 践

推荐几种加密方法.....	( 93 )
软件加密及其实现的一种方法.....	( 94 )
软件加密保护方法.....	( 96 )
一个反动态跟踪程序的破译方法.....	( 99 )
微型计算机的序列加密方法.....	( 106 )
扇区交错保密法.....	( 112 )

用数码形成密文的方法	( 114 )
加密程序的数据输入方法	( 115 )
IBM PC软件的加密	( 117 )
APPLE—Ⅱ微机磁盘防止复制的一种方法	( 124 )
也谈APPLE—Ⅱ机程序的加密	( 127 )
也谈APPLE BASIC程序的加密	( 132 )
简单可靠的APPLESOFT程序保密法	( 133 )
APPLESOFT程序的加密与解密	( 134 )
APPLE Ⅱ应用软件的加密方法	( 135 )
COMX—PC1机上程序加密	( 140 )
R1机BASIC程序的加密	( 141 )
浅谈 BASIC程序的加密与解密	( 142 )
用LIST执行DOS命令程序保密又一法	( 143 )
DOS3.3下文件的简易加密与解密	( 146 )
修改DOS命令达到加密目的	( 147 )
APPLE Ⅱ CP/M软盘加密方法	( 147 )
用BASIC在DBASE Ⅲ程序中加密	( 151 )
一种较为可靠的磁盘加密系统电磁软盘加密系统介绍	( 152 )
APPLE—Ⅱ操作系统技术分析与磁盘加密方法	( 154 )
APPLE—Ⅱ磁盘加密一法	( 160 )
对汉字码加密的解密	( 161 )
为dBASE Ⅲ增加共享文件加锁功能	( 162 )
dBASE Ⅲ数据库软件的加密和解密	( 165 )
计算机网络中的数据加密和密钥分配	( 166 )
COPYWRITE的威力	( 169 )
如何去掉PROLOK的保护	( 175 )
如何破解SOFTGUARD 2.00版的保护	( 179 )
“坏”软盘起死回生	( 182 )
PROLOK激光加密系统分析与解密方法	( 182 )
谈谈加密软磁盘的解密与拷贝	( 186 )
一种加密dBASE—Ⅱ软件的解密过程与方法	( 188 )
对IBM PC/XT上一些游戏程序的解密方法	( 191 )
用2字节的短程序解密	( 193 )
过程对称的制、解密程序	( 194 )
APPLE—Ⅱ BASIC程序加密后的一种解密方法	( 195 )
一种解密加“P”BASIC程序的方法	( 197 )
“P”BASIC源程序的解密方法	( 198 )
加密BASIC程序的一种简单破译方法	( 201 )
用PASCAL编写加“P”保护的BASIC程序的解密程序	( 203 )

一种简便的 IBM—PC BASIC 程序的解密	( 206 )
用DEBUG解密IBM PC BASIC 程序	( 208 )
用IBM—PC BASIC对P 参数文件解密	( 210 )
APPLE 微机解密小程序	( 212 )
自制 BASIC 快速解密程序	( 212 )
对 BASIC 程序加密的一种新方法	( 213 )
制表软件OFFIC 1.00A 的解密	( 214 )
对加密汉字操作系统( 9 针小字 ) 的去密	( 215 )
五笔字型操作系统探讨	( 217 )
APPLE—II 超级软汉字系统 DOS 2.0的一个错误	( 220 )
使用27916KEPROM实现软件加密	( 213 )
汉字码文件名解密方法	( 221 )

## 文件保护与恢复

MICROVMS和 VMS 的文件保护系统	( 225 )
IBM—PC 软磁盘文件的恢复方法	( 238 )
IBM PC软磁盘文件恢复技术	( 243 )
BM—PC 磁盘文件的恢复及文件属性的修改	( 251 )
UNIX系统中误删文件的恢复	( 259 )
文件的属性及其修改应用	( 261 )
数据库系统中恢复管理的设计与实现	( 267 )
一种文件加、解密方法的实现	( 273 )
也谈APPLE程序的保密	( 276 )
文件加密与解密	( 278 )
PC—1500 机程序的恢复	( 280 )
LASER 机程序中部分程序段的删除方法	( 281 )
用 PC—TOOLS 工具盘拷贝加密盘	( 282 )
二字节文件解除各种版本 BASIC 的“P”	( 283 )
恢复内存中(或加密) BASIC程序的一种简易方法	( 283 )
IBM—PC/XT 磁盘文件的一种保护方法	( 284 )
探讨关于修复dBASE III中数据文件的方法	( 285 )
APPLE II 机的dBASE II 命令文件保密	( 286 )
如何修改磁盘文件的卷名	( 287 )
大批量文件在硬盘上的保护方法	( 288 )
COMX 机数据保护妙法	( 290 )
多功能磁盘管理软件(COPY)( PLUS4.3)	( 10 )
APPLE—II DOS3.3磁盘文件的删除与恢复	( 291 )

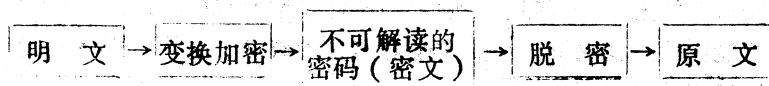
# 计算机与现代密码技术

复旦大学 李为鉴 鲍振东

**摘要:** 本文简要地介绍了加密的基本概念和加密的一般方法。在此基础上较具体地介绍了二进制的加密方法。文中着重介绍的DES体制、RSA体制和浙缩体制都是现在密码技术的研究成果,是在计算机上对信息进行加密的具有代表性的几个加密体制,很有应用价值。

在数据处理中,为了保密起见,可设法将数据进行适当的交换,使之成为“面目全非”的密码信息,这个过程通常称为数据加密;这是数据保密的一种重要手段,有关加密的各种方法不断地出现、讨论十分活跃。特别是计算机用于军事、经济、信贷、储蓄、管理等各个领域后,人们对数据信息的保密要求越来越重视。与之相应地就有一个如何将加密后的信息还原成原来数据的“本来面目”这个过程称为信息脱密。很明显,如果一个加密后的信息,很容易被别人找到脱密的办法,那么,这就是一个价值不大的加密系统,所以,研究加密系统,使得加密后的信息很难被别人找到脱密的办法,就形成一门新的学科——密码学。

我们可将加密到脱密的过程用下面的简图表示:



上述过程所研究的原理,手段和方法,就是密码学的基本内容。

密码学可以追溯到4000年前的象形文字时期。在数千年的密码历史中,真正用密码机进行实际使用,这还仅仅是进入二十世纪以后的事,A·S·Vernam于1920年所发明的电传打字机中的加密方式就可算是最早的“机械化密码”。

## 加密的一般方法

加密的方法大体上有以下两种:

### 1. 代码法

设明文中所用到的所有词所组成的集合为A,密码字符或数字的集合B。我们确定一种——对应的关系。

$$f: A \rightarrow B$$

对于每一个 $a \in A$

$$f(a) = b \quad b \in B$$

且 $f^{-1}(b) = a$

加密和脱密就是依靠着这种对应关系所编制的密码词典。显然,用这种方法进行加密,机动性很差、工作效率不高,如果利用电子计算机,那就要占用大量的存储空间以存放该密码词典。因此,这在实际上是被认为不适宜应用于计算机的加密方法。

## 2. 密 法

密码法通常有代替法、置换法和乘积密码法。

代替法，将明文中的每个字母用别的字母或符号来代替，从而达到加密的一种方法。

置换法：将明文中的字母的排列顺序进行改变，从而达到加密的一种方法。

乘积密码法：通过混合采用代替法和置换法，使明文变换成难于破译的密文的一种方法。

事实表明，特别是在第二次世界大战以后，关于密码方法的研究表明，乘积密码法是一种较可靠的加密方法。

## 密码体制的概述

如上所述，乘积密码法是混合地使用代替法和置换法所得到的一种比较好的加密方法。用不同的混合法所得的加密方法就形成不同的加密体制。



一般地说，加密体制就是用各种加密方法组合起来所形成的一种算法；并且通过该体制的用户所选定的单独密钥的作用来决定算法的具体实现。

在这种情况下，如果整个算法也能保密的话，这对密码的用户来说，当然是再好也没有了，然而，密码体制是由电子线路、计算机程序来实现的，因此，就密码体制来说，在一般情况下是难以保密的。可见，密码体制的设计必须把注意力集中在这样一点上：使企图破译密码的人，在不知道密钥的情况下，难以实现对密文的破译。这样，保护整个数据的机密，就变成保护密钥的机密就可以了。

密码体制的好坏，应该有一些准则，按照C.E.Shannon的提法，应该具备以下五个准则。

- (1) 破译密码需要极大的工作量。
- (2) 密钥的长度很小。
- (3) 加密和脱密所进行的操作比较简单。
- (4) 即使产生错误，错误的扩散也很小。
- (5) 信息被加密后并不改变原信息的长度。

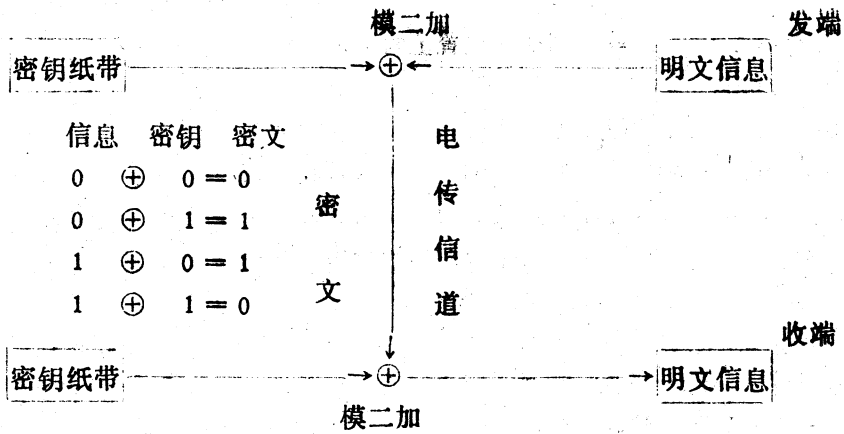
由于现代计算机的发展，算法可以组合到硬件中去，还可以用微编码和微程序，因此，在不降低容量的范围内，算法可以搞得复杂化些。

计算机都是二进制运算的，所以有关二进制信息的加密方法，很自然地引起人们的重视。

早在1920年弗纳姆就提出过一个加密方案，如下图所示：

这个方案的实现要求发端与收端有相同的二进制数码序列的密钥纸带，且双方要保持同步，对于发端来说，将明文信息与密钥序列模二相加后形成密文，并通过电传信道将密文发送出去，对于收端来说，与发端同步，并将收到的密文与密钥序列通过模二加法而得到原来的明文。

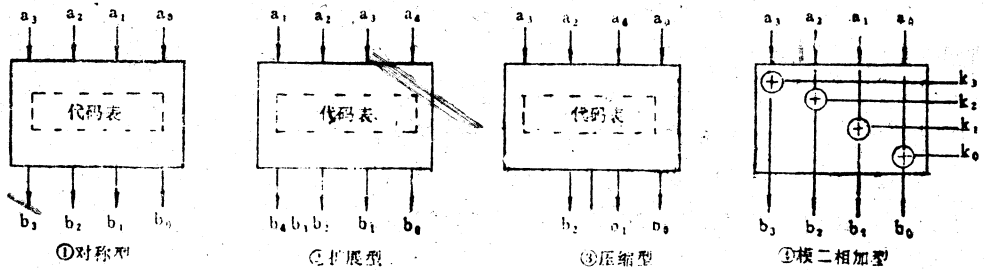
在这个方案中，特定的密钥序列只用一次，因此，即使密文被窃，也是难以破译的，密钥只用次一换，这就称为“一次一密体制”，显然，当明文信息量很大时，密钥序列也就相



应地很长。

对于二进制信息，也可进行代替和置换，简述如下：

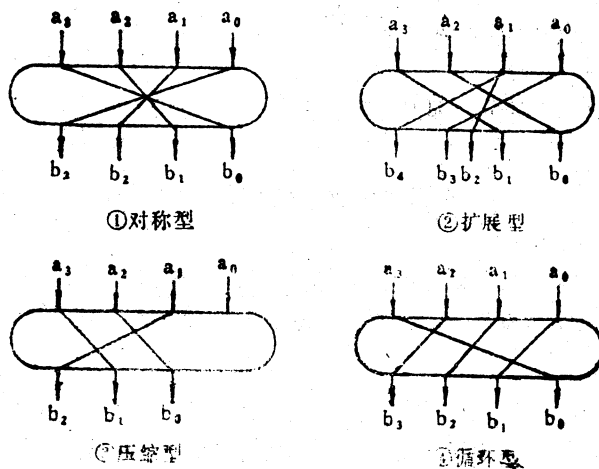
代替：设a为输入，b为输出。



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	11	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	11	5	11	3	14	10	0	6	13

只对压缩型举例说明：对于输入 $(a_3 a_2 a_1 a_0)_2$ 按照以下的代码表中的 $(a_3 a_0)_2$ 行与 $(a_4 a_3 a_2 a_1)_2$ 列交叉位置上的数的二进制数作为输出。

例如：以 $(110111)_2$ 作为输入，则输出应为：





$(11)_2 = 3$  行与  $(1011)_2 = 11$  列交叉位置上的数 14

故输出应为  $(1110)_2 0$

置换：同样，设  $a$  为输入， $b$  为输出

只对扩展型举例说明：由下表可知，将输入的第 32 位作为输出的第一位；输入的第一位作为输出的第二位；……；输入的第 4 位既作为输出第五位，又作为输出的第七位；……；输入的第 32 位作为输出的第 47 位；输入的第一位作为输出的第 48 位。

①	1②	2③	3④	4⑤	5⑥
4⑦	5⑧	6⑨	7⑩	8⑪	9⑫
2⑬	9⑭	10⑮	11⑯	12⑰	13⑱
12⑲	13⑳	14㉑	15㉒	16㉓	17㉔
16㉕	17㉖	18㉗	19㉘	20㉙	21㉚
20㉛	21㉜	22㉝	23㉞	24㉟	25㊱
24㊲	25㊳	26㊴	27㊵	28㊶	29㊷
⑳	29㊸	30㊹	31㊺	32㊻	⑳

例如，对于输入

```

1 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
0 1 0 1 1 1 0 1 1 0 0 1 1 0
18 19 20 21 22 23 24 25 26 27 28 29 30 31
0
32

```

根据上表，经扩展后将输出

```

0 1 0 1 1 1 1 1 0 0 1 0 1 0 1 0 1 0 1 0 0
1 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 1 1 0 0 1
0 1 1 0 0 1

```

我们知道，密码体制的要害是难以被攻击的特征，因此，一个密码体制，总应该有某种关键所在。设计陷门函数，并在密码体制中应用陷门函数，这就是一个很重要的手段。

所谓陷门函数是指具有以下性质的函数  $F$ ：

对于定义域  $X$ ，值域  $Y$ ，若

(1) 计算函数  $F$  的算法存在，即对于  $x_i \in X$ ， $F(x_1, x_2, \dots, x_n) = y, y \in Y$ ，是容易实现的。

(2) 对于几乎所有的  $y \in Y$ ，要求出  $x_i$  所需要化费的计算时间或占用的存储空间，实际上都是不可能实现的。

例如，一个大的合整数  $n$ ，分解它为素因数的问题。在每次运算操作时间为  $1-\mu s$  的机器上，当  $n=30$  位十进制时，约化费 3.9 小时；当  $n=100$  位时，需 74 年；当  $n=200$  位时，要用  $3.8 \times 10^9$  年

因此，若  $X$  是由 100 位以下的所有的素数对组成的集合，则对于  $(x_1, x_2) \in X$ ，若定义函数  $F(x_1, x_2) = x_1 \cdot x_2$ ，由于一个大的合数分解成两个素数的乘积是一个要化费很多时间的问题，所以，这里所定义的  $F$  便可看作是一个陷门函数。

## 几个具体的加密体制

加密体制有很多，这里就几个具有代表性的加密体制作出介绍。

### 一、DES体制

DES是Data Encryption Standard的缩写，是美国国家标准局的数据加密标准。这个体制于1971—1972年初由IBM公司的沃尔特·塔奇曼(W·Tuchman)和卡尔·耶迈斯(C·Meyers)研究成功的，在1973—1974年，美国国家标准局曾两次公开征求能用于电子计算机的保密方案，经过挑选采用了DES系统。经过几年的研究和讨论于1977年1月批准，1977年7月15日生效。后来，花了将近17年人的讨论没有找到简捷的方法来攻击。如果用穷举法来攻击的话，即使一个微秒穷举一个密钥，共有 $2^{56}$ 个密钥，就要花费约2288年的时间。

DES体制是将加密的信息按每64比特分成组，然后用密钥 $k_i (i=1, 2, \dots, 16)$ 经过16次迭代加密运算，从而得到64比特的密文作出输出。

DES体制的加密算法，如图1所示：

这里，作三点说明：

(1) 对于 $f(R, K)$ ，采用如下的体制(图2)

(2) 这个体制经过如此复杂的16轮操作，其目的就是为使明文尽可能增大其混乱性和扩散性，使得输出不残存统计规律，以达到破译者不能从反向推算出密钥。

(3) 在算法中所使用的密钥 $K_1—K_{16}$ ，是用户从给定的64bit的主密钥经过一定的运算而产生的，并不是对用户直接给出16种密钥。

由于在主密钥的64bit中，有8bit是用作奇偶检验码，剩下56bit才作为有效密钥使用，因此，用户可选用密钥可有 $2^{56}$ 种。

因此，使即在给定输入(明文)和输出(密文)的情况下，要想从中推出密钥 $K_1, K_2, \dots, K_{16}$ ，必须用穷举法去搜索 $P_{10}^{56}$ 种情况，这在实际上是难以完成的，尽管对这个系统有不同看法，但是塔奇曼和迈耶斯认为：DES系统在可以预见的将来(约5年到10年时间内)是保密的，用计算机来破译DES密码，将花费极大的费用和力量，因而也将是不现实的。

### 二、RSA体制(即由Rivest, Shamir和Adleman三人提出的体制)

1976年由W. Diffie和M. Hellman [1] 在“密码学新方向”中提出了公开密钥密码的设想。所谓“公开”就是加密密钥可以公开，而解密密钥予以保密。这样，关于密钥分发问题就很方便了，公开密钥就可以像电话号码本那样地公开。譬如，A发给B，那么A只要查一下密钥本上有关B的加密密钥，然后，A就可按这个公开密钥对明文进行加密后发送给B，最后，B就根据自己的解密密钥而将密文变换成原来的明文。

对于任何一个明文P，如果明文很长的话，可以将它分段，每一段信息N可以看成0到 $n-1$ 的一个数。这里n是一个很大的合数，取数r，将N加密为M，使得 $N \equiv M \pmod{n}$ 。称(r, n)为加密密钥，它可以公开。可以选取一个适当的s，使得 $M \equiv N \pmod{n}$ 。称(s, n)为脱密密钥，将它进行保密。

对于n, r, s的选取，可以如下进行：

1. 先取两个非常大的随机素数P, q(p, q位)均为几百位且这两个数之间相差近一百，令

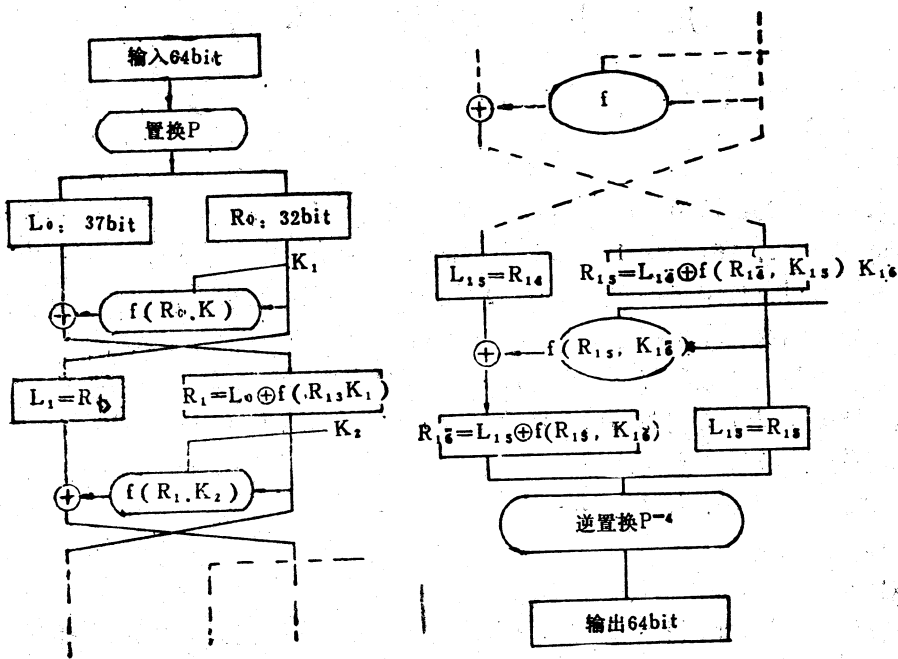


图 1

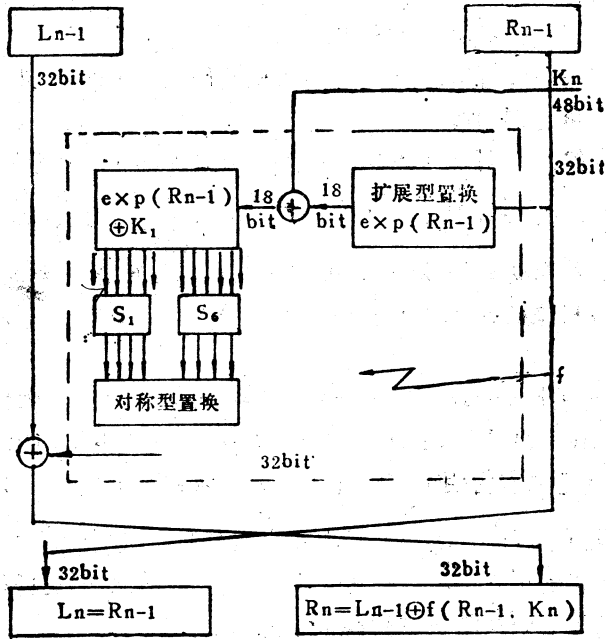


图 2

$n = P \cdot q$ .

2. 随机地取一个大整数  $r$ , 使它带足,  $GCD(r, P-1) = 1$ ,  $GCD(r, q-1) = 1$ .

3. 由  $s \cdot r \equiv 1 \pmod{(p-1) \cdot (q-1)}$ , 求出整数  $s$ .

上述办法选取的  $n, r, s$ , 由于  $n$  很大, 在不路道  $P, q$  的情况下, 要从  $n$  出发进行分解是很难的, 因此, 在公开  $(r, n)$  的情况下, 要求得  $s$  是困难的。这就达到了保密的目的。

现将用  $(r, n)$  作为加密密钥,  $(s, n)$  作为脱密密钥的算法及其正确性证明如下,

$$N^r \equiv M \pmod{n} \quad \text{加密}$$

$$M^s \equiv (N^r)^s \equiv N^{r \cdot s}$$

$$\text{由于 } r \cdot s \equiv 1 \pmod{(p-1) \cdot (q-1)}$$

$$\text{所以 } r \cdot s = k \cdot (p-1) \cdot (q-1) + 1$$

$$\text{故 } N^{r \cdot s} = N^{k \cdot (p-1) \cdot (q-1) + 1}$$

$$\text{由 Fermat 定理可知, } a^{p-1} \equiv 1 \pmod{p}$$

$$\text{所以 } (N^r)^s \equiv N \pmod{p}$$

$$(N^r)^s \equiv \pmod{q}$$

$$\text{最后即得 } (N^r)^s \equiv N \pmod{n}$$

1979年, 由 Bob Blakley 和 G.R. Blakley 发表的长篇文章 "Security of Number Theoretic Public Key Cryptosystems Against Random Attack" 防止随机攻击的数论公开密钥密码体制的“保密性”。详细地论述了该体制的理论依据和具体实现。整个体制实质上是 RSA 体制的推广和发展, 关于这个体制的详细论证, 可参阅上面提供的文章。现将该体制的具体算法叙述如下:

对于给定的  $g, w, u$

1. 随机地选择奇正整数  $a$ , 使得

$$g-1 < \log_2 a < g + \frac{w}{2} - 1$$

2. 对于每个质数  $r < u$  作

$$\text{GCD}(r, a), \text{GCD}(r, 2a+1), \text{GCD}(a, (u-1)/2)$$

如果这三者中有一个不是 1, 则再重做 1。

3. 判定  $a$  与  $2a+1$  是否同时为质数, 若不是则重复做 1,

4. 随机地选择奇正整数  $b$ , 使得

$$g+w-1 < \log_2 b < g + \frac{3}{2}w - 1$$

5. 对于每个质数  $r < u$ , 作

$$\text{GCD}(r, b), \text{GCD}(r, 2b+1), \text{GCD}(b, (u-1)/2)$$

如果这三者中有一个不是 1, 则返回到 4。

6. 判定  $b$  与  $2b+1$  是否同时为质数, 若不是则再重做 4

7. 作  $\text{GCD}(a, b), \text{GCD}(a, 2b+1)$

$\text{GCD}(2a+1, b), \text{GCD}(2a+1, 2b+1)$ , 若其中一个不等于 1, 则返回到 1。

8. 解以下六对同余式

$$(i) A \equiv 0 \pmod{2a+1}$$

$$A \equiv 1 \pmod{2b+1}$$

$$(ii) B \equiv 1 \pmod{2a+1}$$

$$B \equiv 0 \pmod{2b+1}$$

$$iii) C \equiv 0 \pmod{2a+1}$$

$$C \equiv -1 \pmod{2b+1}$$

$$(iv) D \equiv -1 \pmod{2a+1}$$

$$D \equiv 0 \pmod{2b+1}$$

$$(v) E \equiv 1 \pmod{2a+1}$$

$$E \equiv -1 \pmod{2b+1}$$

$$(vi) F \equiv -1 \pmod{2a+1}$$

$$F \equiv 1 \pmod{2b+1}$$

当这六对同余方程的解都不是发信信息时，就转向 9，否则就转向 1。

9. 计算  $P=2a+1$ ,  $q=2b+1$ ,  $m=p \cdot q$

$$v=2ab$$

由  $u \cdot d \equiv 1 \pmod{v}$  求出其最小正整数  $d$ 。

然后，以  $(u, m)$  为加密密钥， $(d, m)$  为脱密密钥，那么，很明显有  $(x^n)^d \equiv x^{nd} \equiv x^{k \cdot 2ab+1}$ ，故有  $(x^n)^d \equiv X \pmod{m}$ ，

为了使得所取的  $m$  较大，且  $P, q$  相差也较大，所以，就要求使  $m$  满足以下的关系式

$$2g < \log_2 m < 2g + 2w$$

如果要求选取的  $P, q$  的范围为：

$$2^g < P, q < 10 \times 2^g$$

则有

$$2g < \log_2 (P \cdot q) < 2\log_2 10 + 2g$$

即

$$2g < \log_2 m < 2\log_2 10 + 2g$$

按上述范围取  $P$  和  $q$ ，那么  $w = \log_2 10 = 3.321\dots$ 。称  $w$  为宽度。

如果取  $P, q$  为两个十进制的 100 位的质数，那么，以上述范围，可求得  $g = 328.870\dots$ ，称  $g$  为规格。

为了保证加密密钥中的  $u$  能使下面的不等式成立

$$x^n > m \quad (\text{对于每一个正整数 } x \geq 1)$$

只须取  $u > 2g + 2^w$  即可，这是因为

$$x^u > x^{2g+2^w} > x^{2\log_2 m} \geq 2^{2g} 2^m = m$$

因此，如果取  $g = 350$ ,  $w = 5$ ，那么， $u$  应取为  $u > 2g + 2^w = 2 \times 350 + 2 \times 5 = 710$

随着计算机网络的发展，公开密钥密码体制较好地解决了通信用户数量极其庞大时密钥的分发问题，可以与数量不确定的大量用户进行保密通信，这显然是它的一大优点。然而，在具体的实施中，对于公开密钥表的使用和管理能否有可信赖的公正第三者这也是一个实际问题，保密通信总是在互相认识的双方为了保守信息的机密而采用的通信方式，在这种情况下，通信双方之间或几方之间当然是经过周密而又慎重的考虑才确定加密和脱密的办法，因此，有关密钥的分发数量也是相当有限并且受到严格限制的。完全的公开究竟又有多大意义呢？有关这些问题，只能在实际使用的过程中获得满意的解答。

### 三、渐缩体制

对于给定的正整数  $a_1, a_2, \dots, a_n$  和  $S$ ，求使得

$$S = a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

成立的  $\{0, 1\}$  解（即  $x_i \in \{0, 1\}$ ）。这就称为渐缩问题。这个问题当  $n$  相当大

时, 解的选择方式用穷举法去做的话, 共有 $2^n$ 种不同的方式。因此, 对于一般的 $a_i (i=1, 2, \dots, n)$ 来说, 渐缩问题是一个NP问题。

当 $a_1, a_2, \dots, a_n$ 满足以下的关系式时

$$\begin{aligned} a_2 &> a_1 \\ a_3 &> a_1 + a_2 \\ a_4 &> a_1 + a_2 + a_3 \\ &\vdots \\ a_k &> a_1 + a_2 + \dots + a_{k-1} \end{aligned} \quad (3.1)$$

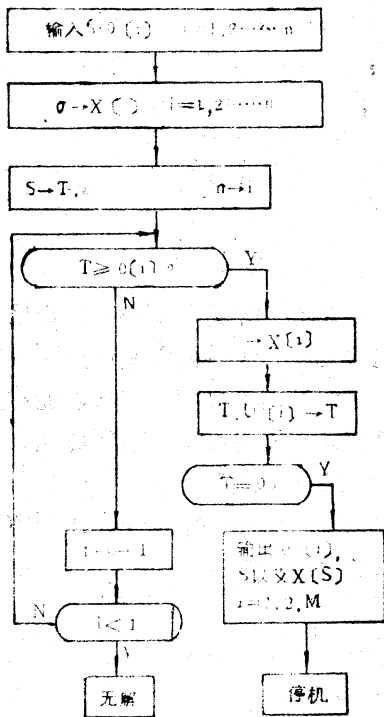


图 3

渐缩问题的解是容易得到的。

例如:  $a_1, a_2, a_3, a_4$

1 4 6 12

S 19 =

因为:  $19 > 12$  故应取  $x_4 = 1$

$19 - 12 = 7$

$7 > 6$  故应取  $x_3 = 1$

$7 - 6 = 1$

$1 < 4$  故应取  $x_2 = 0$

$1 = 1$  故应取  $x_1 = 1$

$1 - 1 = 0$  正好结束

所以, 就解得  $x_1 = 1, x_2 = 0, x_3 = 1,$

$x_4 = 1$

对于满足(3.1)的渐缩问题, 可以有以下的算法。(图3)

在介绍过渐缩问题后, 我们再来建立渐缩体制。

设已知 $a_i (i=1, 2, \dots, n)$ 满足条件(3.1), 任取正整数 $m$ 满足。

$$m > \sum_{i=1}^n a_i$$

取得 $w$ , 使得  $(w, m) = 1$ , 即 $w$ 与 $m$ 互质

由  $w' \cdot w \equiv 1 \pmod{m}$  解出  $w'$

作  $a_i' \equiv a_i w_i \pmod{m}$

对于已知的  $\{0, 1\}$  信息  $x_1, x_2, \dots, x_n$ , 利用  $a_i'$  来进行加密, 得到密文S

$$S \equiv \sum_{i=1}^n a_i' x_i \pmod{m}$$

接收者收到S后, 可用以下的方法来脱

$$S w' \equiv S' \pmod{m}$$

$$\begin{aligned}
\text{因为 } S' &\equiv S_{w'} \equiv \sum_{i=1}^n a_i' w' x_i \\
&\equiv \sum_{i=1}^n a_i w' x_i \\
&\equiv \sum_{i=1}^n a_i x_i \pmod{m}
\end{aligned}$$

所以，就可以解出  $x_1, x_2, \dots, x_n$ 。

在这个体制中， $(a', m)$  可以分开， $w'$  作为解密密钥。

### 参考文献(略)

文献出处：《电脑应用时代》1985年3期1—8页

## 多功能磁盘管理软件 COPY] [PLUS 4.3

徐 凯 泉

在APPLE机上进行磁盘作业时，为了处理类似拷贝文件，修复DOS，改写扇区，编辑扇区内容，恢复被删除的文件；检查文件错误，查询起始地址，了解磁盘文件空间分布等多种复杂问题，需要自编多个小程序，使用起来很不方便。在这里有必要推广多功能磁盘管理软件COPY] [PLUS 4.3。它能使你在APPLE磁盘作业中得心应手，事半功倍。

该软件共有16项功能，下面将其经常使用的几个部分重点介绍如下：

将含有COPY] [PLUS 4.3文件的磁盘放入驱动器D中，打入BRUN COPY] [PLUS 4.3，待片刻便出现以下菜单（此时可将该磁盘取出）：

- |                 |                         |
|-----------------|-------------------------|
| 1. CATALOG      | 9. TRACK/SECTOR         |
| 2. COPY         | 10. VIEW FILES          |
| 3. BEYCOPY      | 11. FIX FILE SIZES      |
| 4. DELETE       | 12. CHANGE BOOT PROGRAM |
| 5. LOCK UNLOCK  | 13. UNDELETE FILES      |
| 6. RENAME FILES | 14. SECTOR EDITOR       |
| 7. FORMET       | 15. NEW DISK INFO       |
| 8. VERIFY       | 16. BOOT DISK           |

1) CATALOG列出磁盘目录，可列出四种目录供选择。

①NORMAL普通标准目录，相当于原CATALOG命令

②W/FILE LENGTHS以10进制和16进制两种方式列出当前磁盘上所有文件长度。

其中B型文件，同时列出起始地址。

③W/DELETE FILES列出当前磁盘中曾经建立过而又被删去的文件目录

④W/HIDDEN CHERS列出包含有隐藏字符（如^A、^D等）的文件名。

2) COPY用来拷贝文件，可有三种形式：

①FILES拷贝单个文件。一次可选择多个文件，拷贝到另一磁盘上。

②DISK将一个磁盘上全部文件拷贝到另一磁盘上。

③DOS将一个磁盘上的DOS系统拷入另一磁盘，而不删去其他文件。这对于有些磁盘DOS损坏不能引导时具有修复功能。

4)5)6)7)部分因使用较为容易，在此不再重复。

8) VERIFY检查磁盘

①DISK按磁道检查全盘。若遇到错误，便显示磁道和扇区号。

②FILES按文件逐个检查。当遇到文件不能调入内存时将显示文件名。

③D-SPEED用来检查磁盘速度，并指出与标准速度之差别。

9) TRACK/SECTOR MAP磁道/扇区图。将显示每个文件在磁盘中所在的位置，可供修改时查找文件位置使用。

10) VIEW FILES显示磁盘中文件所在扇区中的内容。其中①按16进制机器码表示②以文本形式显示。

11) FIX FILE SIZES压缩文件在磁盘中所占多余扇区。有些文件在反复修改中，在磁盘上所占实际空间发生了变化，有时文件长度变短，而原有空间较长，因此利用功能可使多余空间得到释放，增加磁盘应有的容量。

12) CHANGE BOOT PROGRAM此功能可用来改变引导程序名。可在整个磁盘中，随意命名任何一个文件名（可以是A型和B型）为引导程序名（只能选一个）。以后当引导该磁盘时，便立即运行新命名的程序。

13) UNDELETE FILES恢复曾被删除的文件。选择此功能，先列出全部被删文件的名称。你可用回车任意指定哪几个文件需要恢复。再按G即可恢复你失去的文件。若原删去的文件所在空间已被新文件占用，系统将告诉你该文件不可恢复。

14) SECTOR EDITOR扇区编辑。选择此功能后，系统显示：

```
[1] [J] [K] [M] MOVE CURSOR  
[B] EGINNING, [E] ND, [R] EAD, [W] RITE  
[H] EX OR [T] EXTENTER, [ESC] TO EXIT  
[P] ATCH DISK READ WRIT E
```

当我们要修改扇区内容时，可先选[R]，将某一扇区内容读入内存，然后用[I] [J] [K] [M]键来移动光标到需要修改的地方去。再选择[H]按16进制机器码修改（显示于屏幕左边）；若选[T]则用文本形式修改（显示于屏幕右边）。[B]表示将光标移至文首。[E]表示将光标移至文尾。修改结束后按[W]可将已修改的内容存入原扇区。按ESC键即退出该功能。

15) NEW DISK INFO重新设定初值。通过此功能还可设PRINTER ON状态，对必要的内容进行打印。

16) BOOT DISK退出此软件。在D1驱动器中，插好待起动的软盘，按回车后即可引导。

文献出处：《电子与电脑》1988年4期4—5页



# 数据加密的体制及应用

哈尔滨工业大学 朱志莹 曹珍富

## 一、引言

计算机科学和其他科学的交叉渗透,产生了许多离散结构的问题,解决这些离散结构问题所创造的方法,往往产生新的科学分支,突破传统方法的应用范围。

我们熟知,使用计算机的一个基本手段是将所述问题数字化,例如数字信号处理、图象处理等都可归结为数据处理。而数据处理除了数据率庞大需要“压缩”以外,还有数据的加密处理。例如,随着计算机的遍及,“电子信件”将会逐步取代传统的书面信件,人们希望“电子信件”能象传统的书面信件一样只有合法接收者才能知道信件内容,而任何第三人都只能看到“信封外皮”无法知道信的内容。这就要求对信的内容(可以数字化)提供加密。再如计算机中重要数据的加密、重要软件的加密等等,都是数据加密的例子。

因为传统的加密方法是严格保密的,不便在更多的场合下使用(往往只能是政府的军事、外交等部门所专用)。1976年11月,美国的Ciffie和Hellman [1]突破了传统的密码技术,提出了数据加密的一种新的体制,该体制要求每一用户都有自己的两种密钥——加密密钥和解密密钥,其中加密密钥公开,便于通信者用来加密明码,而解密密钥严格保密起来。非法接收者想从公开的加密密钥解出解密密钥几乎是不可能的。这是轰动全球的公开密钥体制(public key cryptosystem),寻找这种体制可以归结为寻找所谓的陷门单向函数,本文将依次介绍陷门单向函数以及目前人们已经找到的一些实现方案。

## 二、陷门单向函数

首先,我们将Diffie和Hellman提出的公开密钥体制用数学语言来描述:

设用户 $x$ 的加密密钥是 $E_x$ ,解密密钥是 $D_x$ ,则要求 $E_x$ 和 $D_x$ 实现互逆变换,即

$$D_x \cdot E_x = E_x \cdot D_x = I,$$

其中 $I$ 是恒等变换,而 $E_x$ 公开,这里要求两条:① $D_x$ 和 $E_x$ 都容易计算;②仅从公开的 $E_x$ 寻找 $D_x$ 是困难的。

现在我们一般地看一下两个用户 $a$ 和 $b$ 是怎样实现秘密通信的。设用户 $Q$ 开送信息 $P$ (称为明码)给用户 $b$ ,则有

(1)  $a$ 用 $b$ 公开的加密密钥 $E_b$ 对 $P$ 进行变换,得到密码:

$$C = E_b(P)$$

(2)  $b$ 收到 $a$ 发送的 $C$ 后,用只有他自己知道的解密密钥 $D_b$ 变换 $C$ 恢复明码 $P$ :

$$D_b(C) = D_b(E_b(P)) = I(P) = P$$

由此可见,寻找适合下面三个条件的函数 $f(n)$ 是构造公开密钥体制的一个重要途径:

(a) 对 $f(n)$ 的定义域中的每一个 $n$ ,均存在函数 $f^{-1}(m)$ ,使 $f^{-1}(f(n)) = n$ ;

(b)  $f(n)$ 与 $f^{-1}(m)$ 都容易计算;

(c) 已知 $f(n)$ , 求 $f^{-1}(m)$ 是非常困难的。

我们把满足这三个条件的函数称为陷门单向函数。这个概念虽然Diffie和Hellman在1976年是提出来了, 但他们没有找到任何例子, 直到1978年, Rivest, Shamir和Adleman [3]才找出第一个陷门单向函数, 被称为RSA体制, 后来Merkle和Hellman [4]也找到一个陷门单向函数(被称为MH体制)此后, 人们对这两个体制作了大量的具体工作, 例如见Simmons和Norris [5]以及胜野裕文 [6]等的论文。

### 三、RSA体制和MH体制

现在我们来介绍这两个体制的具体内容。

#### (一) RSA体制

设 $n$ 为 $[1, m-1]$ 中的任一整数,  $m=pq$ ,  $p$ 和 $q$ 是两个不同的大素数, 选取 $s>0$ ,  $s$ 与 $(p-1)(q-1)$ 互素(即最大公约数等于1)。于是可以找到一个 $t$ ,  $0 < t < (p-1)(q-1)$ 且 $st-1$ 被 $(p-1)(q-1)$ 整除。

定义 $\langle x \rangle_m$ 为整 $x$ 被 $m$ 除的余数,  $0 \leq \langle x \rangle_m < m$ 。则RSA体制的陷门单向函数定义为:

$f(n) = \langle n^s \rangle_m = 1$ ,  $f^{-1}(1) = \langle 1^t \rangle_m$ 。现在我们来分析用户怎样使用RSA体制; 首先, 如果信息量太大, 可以把信息分段使得每段信息代表的数字小于 $m$ 。设 $P$ 为某一段信息, 则用户按照对方公布的 $s, m$ 将 $P$ 变换成:

$$C = E(P) = \langle P^s \rangle_m;$$

对方收到 $C$ 后译码如下:

$$P = D(C) = \langle D^t \rangle_m$$

$$\text{即 } D(E(P)) = \langle (P^s)^t \rangle_m \langle P^{st} \rangle_m = P_m$$

显然, 窃听者即使收到 $E(M)$ , 但因为他不知道 $t$ , 故无法译出 $P$ , 从前面的叙述可以看出, 这里的 $s, m$ 可以看成是加密密钥, 是公开的; 而 $t$ 是解密密钥, 需要严格保密。在 $m$ 选择两个很大的素数乘积时, 要分解 $m$ 是非常困难的, 因此窃听者从 $m$ 难以获得 $p, q$ , 也就难以获得 $(p-1)(q-1)$ 从而由 $s$ 也就难以获得 $t$ , 这就是说, 从加密密钥出发, 想找出解密密钥是困难的, 另外, 用户到素数表中随机地寻找两个大素数 $p, q$ 是容易的, 随机地选取 $s$ 使得 $s$ 与 $(p-1)(q-1)$ 互素也是容易的; 再利用欧几里得算法 [2] 求出 $t$ 使到 $0 < t < (p-1)(q-1)$ 和 $st-1$ 被 $(p-1)(q-1)$ 除尽也是件容易的事, 因此加密密钥和解密密钥都容易计算。至此, 我们验证了RSA体制符合Diffie和Hellman所提的要求。

#### (二) MH体制

用户选定一组数字 $a_1, a_2, \dots, a_n, m$ ,  $w$ 有下列性质:

$$1. \sum_{i=1}^{j-1} a_i < a_j, \quad j=2, 3, \dots, n;$$

$$2. \sum_{i=1}^n a_i < m;$$

3.  $w < m$ , 且  $w$  与  $m$  互素。

于是, 我们可作如下变换:

$a_j = \langle w \cdot a_j \rangle_m, j = 1, 2, \dots, n$ , 并且公开  $a_1, a_2, \dots, a_n$  作为加密密钥。

假设用户之间的通信信息以二进制数的向量来表示, 即信息  $P$  写成  $(b_1, b_2, \dots, b_n)$ ,  $b_j = 0$  或  $1$ , 则用户将用对方公开的  $a_1, a_2, \dots, a_n$  加密得到:

$$C = E(P) = \sum_{j=1}^n a_j b_j$$

对方收到  $C$  后, 先算出  $\hat{C} = \langle W^{-1}C \rangle_m$ , 这里  $W^{-1}$  满足  $WW^{-1} - 1$  被  $m$  整除。于是有等式:

$$\hat{C} = \langle W^{-1} \sum_{j=1}^n a_j b_j \rangle_m = \langle \sum_{j=1}^n a_j b_j \rangle_m = \sum_{j=1}^n \hat{a}_j b_j.$$

然后从  $\hat{a}_j$  的特殊性质解出  $b_1, b_2, \dots, b_n$ 。

由此可见, 在 MH 体制中, 加密密钥是  $(a_1, a_2, \dots, a_n)$  它是一组无规则的数字, 而解密密钥则是  $W$ , 窃听者在不知道  $W$  时, 想从

$$C = \sum_{j=1}^n a_j b_j$$

解出  $b_j$  来是不可能的。因为这时  $a_j$  没有特殊性质, 求解这个问题(是著名的 Knapsack 问题)已经是属于 NP-complete 类的难题。

本节我们介绍了两个有名的公开密钥体制——RSA 和 MH, 它们分别是利用数学中的两个著名难题产生的。RSA 体制是利用寻找两个大素数容易而分解两个大素数的乘积困难这个事实; MH 体制则利用解一般的 Knapsack 问题困难, 但作某种变换使具有一类特殊性质时求解该问题又变得容易这个事实。我们看到, 类似这样的问题还很多, 可以用来构造另外的公开密钥体制。

## 四、若干应用

很明显, 凡是涉及到数据加密处理的问题都可以应用公开密钥体制。例如我国的电报信息加密, 图象加密和电子邮件等等, 都可以应用。

### (一)、电报信息加密

大家知道, 我国的电报码是用十进制的四位数表示一个字, 一份电报文  $P'$  便是一个很大的正整数, 将  $P'$  分成若干段, 便每段代表的数字  $p$  适合前述 RSA 或 MH 体制, 由此可见, 只要在邮电局增设某些计算机(不一定要求速度很快的例 Z80 型微机), 将 RSA 或 MH 体制编成程序存在磁盘上, 就可实现电报信息的加密了。这在军事、外交等方面也有实际的应用价值。

### (二)、图象加密

首先把图象数字化, 例如一帧图象的象素数目为  $256 \times 256$ , 行列素分别写成  $\{R_j\} = R_1 R_2 \dots R_{256}$ ,  $\{S_i\} = S_1 S_2 \dots S_{256}$ , 其中  $R_i$  和  $S_i$  都是十进制的三位数。在图象的加密过

程中,由于不改变每个象素的灰度级,而仅仅改变象素的地址,因此信息量一般不会增加。对  $\{R_i\}$  和  $\{S_i\}$  采用RSA体制或MH体制加密变换,可实现图象加密。

### (三)、电子信件

当计算机在全国联成网以后,人们希望通过计算机传送信息,而发送信息的用户希望仅收方才能知道的信息内容,任何其他用户无法获知,这只有用公开密钥体制才便于实现。

总之,只要能够数字化的一切领域(包括已经有分支和将来发现的新分支),都将与数据的加密处理发生关系,而解决这个关系的工具是公开密钥体制。因而研究新型加密体制就是一个重要问题。

## 五、结束语

公开密钥体制是一个新兴的应用科学领域,它是传统密码学的一个突破。但实现这种体制的陷门单向函数只是暂时的,因为陷门单向函数的提出,一般都是基于某些数学难题,但今天的难题明天就有可能解决,因此使用公开密钥体制大有提心吊胆之感。这个缺陷告诉我们,没有一劳永逸的公开密钥体制供人类使用。

文献出处:《电脑学习》1986年3期5—7页

---

上接19页

数据的敏感以及通常的实践来决定。

一次性口令最安全,但难于记忆并往往给用户增加麻烦。

编者注:本文还有《硬件的物理安全与环境安全技术》等略

# 微型计算机应用系统的安全技术

上海市电子振兴领导小组办公室 劳诚信

## 一、计算机应用的安全概念

计算机已广泛应用于社会的各个领域，大大提高社会生产力和工作效率。但是如果不注意安全问题，也可能产生严重的后果，影响计算机的推广使用。因此，安全问题是保证计算机正常运行，发挥计算机效能的不可忽视的重要因素。计算机应用系统的安全技术已引起人们越来越广泛的重视。

在讨论计算机应用系统的安全技术之前，首先要明确计算机应用的安全概念。

计算机应用安全主要指设备防护，避免自然灾害（如火灾）和计算机系统的非授权使用。从更广义上来说，计算机安全是以保证微型计算机系统安全，不间断地运行和保护计算机硬件、计算机程序及其数据文件的措施，以保护用户的财产。

具体地说，安全技术和措施可协助用户解决以下问题：

- 采取措施使电源失常和中断所造成的数据处理中断降为最低；
- 降低火灾、烟尘和水患破坏的风险；
- 提供安全运行环境，避免用户数据文件的偶然性破坏，减少操作差错；
- 保护软盘中存放的程序和数据文件；
- 防止未经授权人员擅自使用微型计算机系统；
- 防止未经授权人员由系统取得复制资料，修改已存入系统的数据或数据文件非法输入；
- 确保关键信息不被偶然或故意地从磁盘删除；
- 用软件方法开发高水平安全措施，保护已存入计算机系统的信息；
- 如果用户的计算机系统出现故障，能尽快地恢复正常运行。

计算机安全专家公认，保护计算机系统的软件（数据文件和程序）和硬件（设备）有四层安全措施：第一层是法律和社会管理，它通过国家法和地方法及社会上允许的行为方式提供保护；第二层是用户本单位的规章制度，规定有关微型计算机系统使用方法和手段，以及人事结构和人事政策提供保护；第三层是物理保护，是指使用防盗和防火技术，软盘保护，用特殊方法限制实际接触微型计算机系统；第四层是电子的和程序的，是指由硬件器件和在操作系统、数据文件和应用程序中，用软件方法提供的保护。

## 二、法规及管理的安全措施

### 1. 制定国家信息安全法规

瑞典是世界上第一个制定信息安全法的国家，1973年就制定了数据法，现在世界上已有二十多个国家订了信息安全法，规定人员数据方面所遵循的准则，做到信息保护有章可循，有法可依。

- 为了保护软件产品不受侵犯，国际上也建立了保护法，常用的有：专利保护法；版权

保护法；商业秘密保护法等。

·关于信息安全的技术规范，许多国家也正在制定。例如，加拿大制订了政府用小型计算机安全标准；美国制定了政府机关计算机房标准，数据加密标准，计算机网络身份认证指南等。

## 2. 加强计算机安全制度

·对工作人员进行安全性教育，这是改善和实施有效安全保密措施的重要前提。

·严格规定信息资源共享的等级和范围，明确批准授权使用人员。

·采用分工负责制，即由个人负责不同的操作步骤，相互制约可大大减少信息破坏的可能性。

·对于从事有密级的信息工作人员，坚持先审查后录用。

·严格规定计算机系统使用方法和操作安全，有效地解决允许“谁”可以操作，“哪一个”信息可被操作等。

# 三、软件与数据安全技术

依赖于计算机信息系统的业务，无论大小，都须保证数据的安全。软件与数据安全是指保证数据安全是指保护信息、数据文件和计算机程序，使之免于受到偶然的或人为的非法泄露、修改、破坏。数据安全措施应对正常的日常操作造成最小妨碍，对用户带来最小不便的条件下，提供必要的保护。

有效的数据安全措施有四个显著的特点：

(1) 保密性，未经批准不能提供有关信息（数据文件）或计算机程序；

(2) 整体性，未经批准不得修改数据和程序；

(3) 可用性，当计算机安全事故发生时，抵制和克服事故后果的能力；

(4) 可检查性，监视和鉴别计算机系统操作正常和异常的能力，从而可检测出计算机安全的缺口。

## 1. 数据安全加密技术

加密是一项保护有价值的程序和数据的程序。它使程序和数据文件对任何未被授权但要破译这些程序的人变得难以理解。密码技术包括了密码和代码在内的加密书写的全部领域。且术语“密码”意味着一种加密书写的方法，即把“清晰文本”中的信息或可读材料，利用给定的方法，变为“加密文本”。为了简单起见，下面介绍加密技术几种非常基本的形式，以未加密文件信息“Buy gold”为例加以说明：

·消隐加密，利用某种设备或标法将未加密文件的真实字母隐藏起来，或进行伪装。例如分解上述文本信息“Buy gold”，每次只用其中一个单词并使其出现在每句中的第五单词处，这样，信息“Buy gold”的密码就变为下列形式：

Product is a good buy. It has ten percent gold contant.

·互换加密，利用预先约定的方法打乱原始信息中字母的正常排列次序。例如约定使用下列方法：

(1) 第一个单词的第一个字母；

(2) 第二个单词的第一个字母；

- ( 3 ) 第一个单词的第二个字母;
- ( 4 ) 第二个单词的第二个字母;
- ( 5 ) 第一个单词的第三个字母;

.....

按此规律排列下去, 原文本信息 "Buy gold" 的加密信息就以下列形式出现:

BGUOYLD

· 替代加密, 根据预先确定的某种钥匙, 用替代字母来取代原始信息中的 每一个字母

例如, 按某个给定的数目左移字母表中的字母 ( 本例左移三位置 );

未加密文本: ABCDEFGHIJKLMNOPQRSTUVWXYZ

加密文本: DEFGHIJKLMNOPQRSTUVWXYZABC

按照这一规律, 文本信息 "Buy gold" 的密码将以如下形式出现,

EXBJROG.

## 2. 文件安全保密的数据压缩技术

数据压缩技术不但可减少程序和数据文件对存储空间的需求, 而且用于文件安全保密。数据压缩或文件压缩就是删去空格和/或零而不失去任何信息的处理过程。由于机器处理数据的方式, 通常存储的数字和/或字母数据中有多余的零和/或空格。存储文件的直接开销是存储介质; 而压缩后的文件只需较少的存储空间, 从而可在软盘或硬件上存储更多的程序和数据文件。间接开销则是在进行备份拷贝时的传输数据时间。

有许多数据压缩方法可以使用, 最简单的方法是消零法, 这项技术包括消去零或空格, 或两者兼有。这类压缩方法被称之为标准方法。下面是一个零和空格 ( 用b表示 ) 的消零法例子:

原始数据: A1000000XB25000000MbbbbbbCOST

压缩后数据: A1#6XB25#7N%5COST用符号#表示零, 符号%表示空格, 例#6表示删除6个零; %5表示在原始数据中有5个空格。

格式替代法是一种较为复杂的数据压缩方法。用下例加以说明:

原始数据:

AW10004MFQ00000F32000GBCX4

AW20000DBF00000F300000BCX<sub>1</sub>

AW3000RBA00000F301214BCX<sub>7</sub>

代替格式表

AW = #

000 = \$

00000F3 = %

BCX = •

则压缩数据为

#1 \$4MFQ%2 \$0•4

#2 \$0DBF% \$00•1

×3 \$2RBA%01214•7

还有许多其他的数据压缩方法, 效果较好, 但是更为复杂,

### 3. 口令字控制的安全技术

为了防止非法访问数据文件，避免通过传输线在磁盘和其他计算机之间进行数据文件的非法传送，为防止程序和数据文件被复制，口令控制的安全技术可能提供可靠的安全措施。为达到这个目的，口令控制字应该是：

- 难猜
- 对用户易于记忆
- 容易改变
- 得到系统的良好保护

口令控制方案应考虑下述问题：

#### (1) 口令字选择

口令字可由用户选择或由系统指定。用户选择的口令字安全性较差，因为大多数人倾向于选择具有个人意义的字或数（例如：生日、门牌号码、电话号码、家属的名字等），结果非常易猜。但是采用用户选择口令字的主要优点是易于记忆。由安全管理程序指定的、甚至由计算机系统本身随机产生的口令字远比用户选择的口令字安全，然而，系统指定的口令字更难以记忆，往往需要写下来。

#### (2) 口令字的物理特性

任何口令字的物理特性都是由字母或字符的长度及其编排所确定的。口令字空间的大小往往确定“破译”口令字代码的难易程度。例如采用三个字符组成口令字，而且

- 如果只能用字母符号就有可能拥有17576个不同的口令字；
- 如果只能采用数字字符，就有可能拥有1000个不同的口令字；
- 如果能采用字母和数字符号，则有可能拥有46656个不同的口令字。

#### (3) 信息内容

某些口令字系统采用了从一个口令字中获得符加信息的技术以便核实口令的有效性。这里举例说明被称为自动检测位的办法。假定用户采用数562作为口令字，采用检测位，安全管理程序要求用户在562这个数后加上一个9作为最后一位从而变成5629，而“9”则是根据下列算式获得的检测位：

$$\begin{array}{r} \text{口令字} \quad 5 \quad 6 \quad 2 \\ \text{乘数} \quad 3 \quad 2 \quad 1 \\ \text{计算} \quad 5 \times 3 = 15 \\ \quad \quad 6 \times 2 = 12 \\ \quad \quad 2 \times 1 = 2 \end{array}$$

$$\text{乘积之和} \quad 15 + 12 + 2 = 29$$

和数被10除，余数“9”作为口令字符加检测位。

#### (4) 口令字有效期

当前的口令字方案允许指定的口令字使用不定周期时间（有效期）。固定周期时间（例如一个星期或一个月）和一次性使用。

不定周期时间的优点易于记忆，缺点是脆弱而易被破译；

固定周期时间的优点易于记忆，比较安全，其缺点脆弱和易破译程度依赖于时间间隔，间隔越短，安全性越高，是使用最普通的一种方法，其变动的频率由工作的环境。（下转15页）



# 密钥管理探讨

华中工学院计算机系 王 华

**摘要** 本文提出了一种多级分散密钥管理的模型,根据该模型和Vandermonde矩阵的有关性质,设计和实现了一个行之有效的多级分散密钥管理的算法

**主题词** 计算机安全

## 一、前言

由于计算机网络,数据库等迅速发展和广泛应用,数据的安全问题已提到议事日程上来了,越来越多的人从事于这方面的研究。密码技术是对传输及存贮在各种媒体上的数据实施保密的一种有效手段,它能对高度机密的数据提供高程度的保护,可以作为其它数据保密措施的补充。密码技术的关键是加/解密算法和密钥管理。特别是当采用公开加/解密算法的方法,密钥的安全管理就显得尤为重要,尤其是对文件加密,最基本的要求是需要一个既切实可行又安全可靠的密钥管理体制。

本文针对密钥管理未引起人们足够重视的现状,对密钥管理问题进行了一些探讨,提出一种多层次分散管理密钥的模型,并依据此模型设计了一种多级分散密钥管理的算法。

## 二、密钥管理介绍

密钥是保密系统的核心。由于加/解密算法是公开的,知道了密钥也就知道了明文,所以密钥管理在加密系统中起了维护、巩固加密系统安全的重要作用

密钥管理常有两种方法:多重加密的方法和分散保管的方法。

多重加密的方法常用于要求保护的密钥较多的情况。这种方法首先用加密密钥的密钥对这些密钥加密,然后再对加密密钥的密钥加密,直至加密密钥的密钥较少时才停止加密,最后用硬件方法或其它方法对少量的加密密钥的密钥进行保护。这种方法实质上就是对大量的密钥保护问题转换为对少量加密密钥的密钥保护问题。该方法适用于网络中的数据通信保护问题。

分散保管方法常用于要求保护的密钥较少的情况,比如文件加密系统。它的思想方法是:把要保管的数据按某一分解函数分解 $n$ 个不同的子数据,设定某数 $K$ ,对任意 $K$ 个不同的子数据相结合可以恢复原数据,而任意 $R$  ( $R < K$ )个子数据相结合不能恢复原数据,然后把 $n$ 个子数据分给 $n$ 个保管员进行保管,分散保管的方法体现了又分散又集中的辩证思想。这种方法可以提高密码系统的灵活性,可靠性,能防止因意外事件而丢失密钥的情况。

由分散保管方法的介绍我们可看出:这种方法并不能满足实际的需要。下面对一般分散保管的方法进行了补充,提出了多级分散保管的模型。

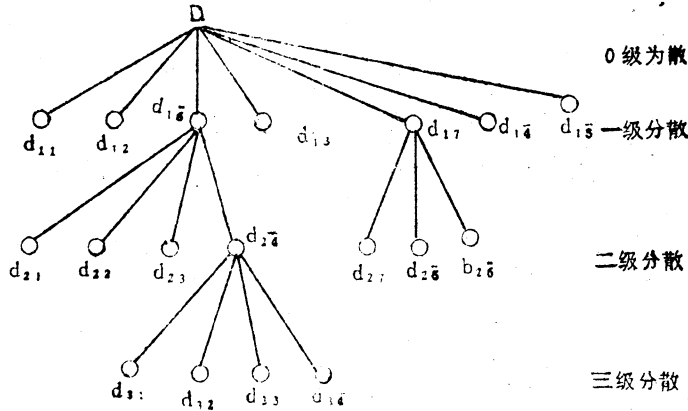
## 三、多级分散保管密钥

实际处理问题时,仅用一级分散保管的方法是不能满足要求的。因为整个社会的管理是多层次的管理结构,各企业、组织、社会团体的管理也是一种层次管理的结构,所以对数据

的处理必然要求不同地位的人员和不同性质的组织具有的处理权限不一样，下面设计的多级分散密钥保管方法反映了这种权限要求，是解决密钥子数据加权分配问题的一种有效方案。

多级分散保管的思想为：设分解函数 $f_1, f_2, \dots, f_n$ 分别表示为第1级、第2级、 $\dots$ 第 $n$ 级的分解函数，设 $d_i$ 为第 $i$ 级分散后的一个结果，则对该数据进行第 $i+1$ 级分散也就为 $f_{i+1}(d_i) = (d_{i+1,1}, \dots, d_{i+1, m_{i+1}})$ 对 $(d_{i+1,1}, \dots, d_{i+1, m_{i+1}})$ 存在一个整数 $m_{i+1} \geq K_{i+1} \geq 1$ ，使得 $(d_{i+1,1}, \dots, d_{i+1, m_{i+1}})$ 中任意 $K_{i+1}$ 个数据能恢复 $d_i$ ，而任意小于 $K_{i+1}$ 个数据不能恢复出 $d_i$ 。

上述思想能用树很具体地表示出来，下图说明了一个三级分散的例子：



分散结果为  $\{d_{11}, d_{12}, d_{13}, d_{21}, d_{22}, d_{23}, d_{31}, d_{32}, d_{33}, d_{34}, d_{24}, d_{25}, d_{26}, d_{24}, d_{15}, d_{16}, d_{17}, d_{14}\}$ 。

多级分散保管密钥会产生一系列问题，最主要的是如何解决好后余问题。比如在上列中数据  $\{d_{27}, d_{28}, d_{26}\}$  和数据 $d_{17}$ 对恢复 $D$ 作出的贡献是相同的，为此须对不同级分散的子数据作标记，以便在恢复时能消去冗余操作。下节设计的算法中，标记的设置是由分散函数完成，不必另外再进行标记处理。

由上模型可知，不同级的分散可用不同或相同的分散函数。一个好的多级分散密钥管理系统的核心是设计一个性质较好的分解函数。分解函数的选择是多级分散密钥管理的难点和重点。下节介绍了一个多级分散密钥管理分散函数的算法，该算法在IBM-PC机上调试通过。

#### 四、一种多级分散密钥管理的分散函数

一般而言，密钥是一个矩阵。下面的分散函数是针对密钥为矩阵的情况设计的。

所设计的运算均限于有限域GFIP。其中 $P$ 为大素数 $\Lambda P > \max_{1 \leq i, j \leq n} (K_{ij}) \Lambda P > n + 1$ 。

设密钥 $K$ 形为

$$K = \begin{pmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \dots & \dots & \dots & \dots \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{pmatrix}$$

第一级分散:

对于任意给定的  $x_{1i}$  利用下式能得到一相应的  $n$  维向量

$$Y_{(i)}^{(1)} = \begin{pmatrix} Y_{(i)}^{(1)} \\ Y_{(i)}^{(1)} \\ \vdots \\ Y_{(i)}^{(1)} \end{pmatrix} = \begin{pmatrix} K_{11} \cdots K_{1n} \\ K_{21} \cdots K_{2n} \\ \dots\dots\dots \\ K_{n1} \cdots K_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ X_{1i} \\ X_{1i}^2 \\ \dots \\ X_{1i}^{n-1} \end{pmatrix}$$

将  $(X_{1i}, Y_{(i)}^{(1)})$  作为第一级密子数据分配给权限较高的管理员。选择不同的  $x_{1i}$ , 就能得到不同的密子数据, 则在  $z_p$  中最多可以得到  $H$  个不同的子数据, 又  $P$  是一个足够大的大素数, 所以子数据的个数能满足所需要的个数。

现讨论恢复问题:

因为  $\forall i \neq j (i \neq j \Rightarrow x_{1i} \neq x_{1j})$  为真, 所以在整数域中,  $\gcd(\{x_{1i} - x_{1j}\}; P) = 1$ , 如果  $i \neq j$ , 由此结论可得: 对于任给  $n$  个不同的  $x_{1i}$  所组成的范得蒙行列式都有:

$$\begin{vmatrix} 1 & 1 & \dots & 1 \\ x_{1i1} & x_{1i2} & \dots & x_{1in} \\ \dots\dots\dots \\ x_{1i1}^{n-1} & x_{1i2}^{n-1} & \dots & x_{1in}^{n-1} \end{vmatrix} = \prod_{\substack{n \geq i > j \geq 1}} (x_{1ij} - x_{1ij}) \neq 0$$

所以

$$X = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_{1i1} & x_{1i2} & \dots & x_{1in} \\ \dots\dots\dots \\ x_{1i1}^{n-1} & x_{1i2}^{n-1} & \dots & x_{1in}^{n-1} \end{pmatrix} \text{ 的秩为 } n, \text{ 即 } x \text{ 为满秩矩阵}$$

因此  $x$  存在逆, 设为  $x^{-1}$ , 则  $K$  的恢复安式为:

$$K = \begin{pmatrix} Y_{11} & Y_{12} & \dots & Y_{1n} \\ Y_{21} & Y_{22} & \dots & Y_{2n} \\ \dots\dots\dots \\ Y_{n1} & Y_{n2} & \dots & Y_{nn} \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_{1i1} & x_{1i2} & \dots & x_{1in} \\ \dots\dots\dots \\ x_{1i1}^{n-1} & x_{1i2}^{n-1} & \dots & x_{1in}^{n-1} \end{pmatrix}^{-1}$$

那么, 任给  $n$  个不同的子数据利用上述公式就能恢复  $K$ , 而任意小于  $n$  个子数据却不能恢复  $K$ 。经过上述处理, 我们实现了密子数据的一级分散。

第二级分散:

为了使问题简化, 对于各级分散都采用相同的分解函数, 第二级分散步骤如下:

① 设  $y_{(i)}^{(1)}$  为第一级分散的结果

$$y_i = \begin{pmatrix} y_{i1} * t_1 & t_{12} \cdots t_{1n} \\ y_{i2} * t_2 & t_{22} \cdots t_{2n} \\ \dots\dots\dots \\ y_{in} * t_n & t_{n2} \cdots t_{nn} \end{pmatrix}$$

其中:  $\forall i (n \leq i \leq 1 \wedge t_i = \sum_{j=2}^n t_{ij})$  为真。

$t_{ij} (i=1 \dots n, j=2, \dots, n)$  为随机数。

②用第一级分散的方法对 $\bar{y}_i$ 分散

对任意的 $x_{2i} \in GF(P)$ ，利用下式可得一个 $n$ 维向量

$$y_i^{(2)} = \begin{pmatrix} y_{i1}^{(2)} \\ y_{i2}^{(2)} \\ \vdots \\ y_{in}^{(2)} \end{pmatrix} = y_i \bullet \begin{pmatrix} 1 \\ x_{2i} \\ \dots \\ x_{2i}^{n-1} \end{pmatrix}$$

③将 $(x_{1i}, x_{2i}, y_i^{(2)})$ 三元组作为第二级的子数据分散给权限较低的密钥管理员，

然后对 $y_i^{(2)}$ 作类似处理，可以实现三级分散。依此类推，可实现所需要级数的分散。

由上面分散过程我们可看到，不同级分散的结果维数不同；各级子数据权限值（所谓权限值就是某子集数据恢复密钥 $K$ 的可能性），有如下关系，用 $P$ 表示：

第0级子数据的权限： $P(K) = 1$

第1级子数据的权限： $P(x_1, y^{(1)}) = \frac{1}{n}$

.....

第 $k$ 级子数据的权限： $P(x_1, \dots, x_i, y^{(K)}) = \frac{1}{n^k}$

利用这种方法进行密钥管理时，恢复密钥的关键是矩阵的求逆。如果子数据少于 $n$ 个，想恢复上一级的数据是相当复杂的，只能用猜测法进行恢复，更何况，上述运算都是在有限域 $GF(P)$ 上进行的，更增加了利用猜测法进行恢复的复杂性。

## 五、小 结

密钥管理在加密系统中特别是加/解密算法公开后起了至关重要的作用。多级分散保管密钥是一种比较理想的方法，既能实现权限分散又具有一般分散保管的优点。在多级分散保管的具体设计中，关键是分解函数的设计，本文对上述各方面都进行了较具体的分析。

### 参 考 文 献

- [1] Ernst L. Leiss《Principle of Data Security》
  - [2] Andrews Tanenbaum《Computer Network》
  - [3] H. 卡茨安《标准数据加密算法》
  - [4] 卡尔. H. 迈耶, 斯芬. M. 万斯《密码学 计算机数据保密新领域—保管系统设计和实现指南》
- 文献出处：《计算机工程与设计》1987年5期44—47页

# 提高微机数据库信息安全保密性的措施

复旦大学计算机科学系 曹文君

**摘要：**本文提出几种实用的能提高微机数据库信息安全保密性的措施和实现技术。它们涉及到行政管理制度和安保措施等多方面的综合论述

## 一、问题的提出

在信息社会里，一个企事业单位拥有一个建立在数据库基础上的管理信息系统，是能在激烈的竞争中赖以生存和发展的有利条件。但是若不重视对系统的安全保密性建设，一个缺乏可靠安全保证的系统会使拥有者蒙受巨大的损失，特别是对于一些要害部门，例如：银行，股票证券公司，政府部门，公安机关，军工仓库…等。因此，对于信息的安全保密性的考虑已经成为数据库设计的重要问题之一。

目前，微机是我国在各种计算机管理信息系统和数据处理系统中应用最广泛的一种计算机机种，防止利用计算机犯罪，保护微机数据库中信息的正确性和完整性，防止机密文件的外泄是广大微机用户共同关心的一个问题。近几年来，我们在微机上成功地开发了商业经营管理，人事档案管理，财务管理，仓库管理，生产管理等多个管理信息系统（MIS）。根据不同系统的特点和用户的特殊要求，采取了多种能提高数据库信息安全保密性的措施，收到了良好的实际效果。

## 二、提高数据库信息安全保密性的行政措施

为了提高微机数据库中信息的安全保密性，首先应该从思想上和组织上重视这个问题，制定一套行之有效的系统管理的规章制度。根据我们的实践，主要采取以下几点措施。

### 1. 人员组织

系统执行成员必须包含有系统操作员，系统维护员，资料保管员等组成有严格分工的一套系统运行班子。在该组织中采取分工负责制，各司其职。系统操作员的职能是计算机操作的执行者，负责录入原始凭证上的数据，核对输入数据的正确性，进行信息的加工，查询，修改等操作并向有关领导提供综合的一级或二级管理信息。系统维护员的职能是维护系统中的各种硬设备和软件，保证系统有良好的可运行状态。资料保管员的职能是保管系统中的一切资料文档，避免系统资料受到不正当的侵犯。

### 2. 信息查询

系统中的信息分为二级，一级信息指将录入的原始数据加工成日常报表中的二维信息，二维信息指将一级信息再加工综合成辅助管理决策信息。

信息查询时，应按照查询人员的级别和司职范围分档开放信息数据库内容。一般部门负责人只允许知悉与自己业务有关系的那部分一级信息。二级信息一般只对单位最高决策层领导人员开放。

查询者需要查询信息时，应当向系统操作员提出要求，由系统操作员代为执行，查询者

不得擅自上机操作查询数据库中的任何内容。

### 3. 信息修改

当需要修改数据库中已经核准过的某些信息内容时，首先必须提出书面修改申请报告，由有关领导人员核准后，再由系统操作员执行。为了确实保证修改后数据库内容的正确性，在执行修改操作以后，必须打印出修改前后有关文件内容清单，便于核实修改的操作情况，并将申请报告和修改内容清单作为系统资料存档，以备后查。

### 4. 资料保管

系统资料包括在纸介质上的有：原始凭证，报表，报告，清单，计算机软、硬件技术文字档案等；在磁介质上的有：磁盘，磁带，磁性卡片等。存档的资料都要分类编号，标明建档日期，修改日期，修改内容和操作员姓名，资料借阅记录等，对一些有密级的资料还应采取特别的安全措施。

## 三、提高数据库信息安全保密性的技术措施

在上节中我们从系统管理的角度叙述了几种提高数据库安全保密性的行政措施。但是光从管理上采取安全措施是远远不够的，提高数据库安全保密程度的关键在于数据库设计时必须采取一些有效的技术手段，例如：身份验证，信息掩蔽，软件加密等，这些措施可以分别在系统级和用户级上用软件实现。以下为我们在数据库设计时所采用的几种安全保密的技术措施，提供读者在建立数据库时作为参考。

### 1. 用户身份验证技术

采取系统用户身份验证措施有三点作用。第一可以防止闲散人员玩弄计算机误入数据库信息系统，引起信息库数据被破坏的可能性。第二可以有效地阻止与系统操作无关人员擅自查取有关信息。第三在多用户之间起到信息操作互相隔离的目的。

具体实现方法为：

(1) 将数据库中的信息和信息处理软件按种类划分为若干个子目录存放，每个子目录可根据需要再细分为孙目录，……。整个数据库存贮成为一棵树状结构。对于每一用户的信息处理的流程限制在沿着事先规定好的一目录路径上进行，例如：主子<sub>1</sub>—子<sub>1</sub>—子<sub>1</sub>—子<sub>11</sub>。

(2) 数据库管理系统给每个用户分配一个标识代码和一个保密口令。在数据库设计时建立一个用户存取权限控制文件，用来登录系统中所有用户的标识代码，保密口令，操作目录路径。

(3) 进入数据库操作状态采用批命令执行方式。用户在开始访问数据库之前，启动批命令文件，当出现系统核对身份提示信息时，输入用户自己的标识代码和保密口令，系统根据输入信息在用户存取权限控制文件中核对身份，然后进入相应的子目录工作图。

实践证明，该方法能有效地避免非用户操作人员越过软件引导部分中的身份核对程序直接进入某个信息处理目录区的可能性。这是因为若甩掉身份验证关，就要知道规定的操作目录路径，而由于目录路径的复杂性，要找到一条正确的路径是极其困难的。

为了解决对密码口令本身进行保密的问题，我们对上述方法进行了适当改进，采取了密码口令自动修正和存贮技术。用户每次上机操作后，系统会自动地修改掉原来用户的密码口令。具体改进的办法有：

(1) 每个用户具有一张身份软盘, 在盘中至少应有二个文件, 一个是密码口令数据文件, 另一个是密码口令生成和替代执行文件。

(2) 用户上机操作时, 首先将身份软盘片插入B驱动器卡口。系统启动后, 自动读取用户密码口令, 与用户输入的标识代码一起与存取权限控制文件中的内容相匹配, 核对用户的身份。

(3) 当用户上机操作结束时, 系统通过执行在软盘上的密码口令生成文件, 取六位随机整数作为该用户新的密码口令, 分别去替代密码口令数据文件和存取权限控制文件中的相应内容, 从而达到密码口令的更新工作。

## 2. 信息掩蔽技术

如果一个非法用户跳过身份验证进入某个目录区, 虽然不能进行有效的信息处理操作, 但可以通过查看本目录区文件名打开一些数据库文件, 查阅其中的信息内容。所以对于一些需要保密的信息文件, 要采取一些信息掩蔽手段。首先使这些库文件不易打开, 其次即使打开了亦难读懂信息内容。

下面叙述我们在微机数据库设计中所采用的信息掩蔽方法:

(1) 对于机密信息文件采取加密措施, 以dBASE关系数据库为例。

加密方法:

USE<文件名>#

保密口令: (用户自己给出保密口令)

机密文件, 不准打开!

解密方法:

USE<文件名>#

保密口令: (用户自己给出保密口令)

加密与解密步骤中所给出的保密口令一致时, DBMS 系统才允许打开该文件, 否则就不能打开文件, 达到对信息文件保密的要求。

(2) 用库文件字段项名规范化技术来增加数据库信息的难读性。

具体实现方法:

A. 将库文件结构中的字段项名一律规范化为: 项1, 项2, 项3……。用抽象的字段项名代替具有现实意义的字段项名。

B. 编制一个转换程序, 将规范化的字段项名复原出它们所代表的真正含义。

C. 在正常的查询操作或报表输出的情况下, 可以通过如图4所示的变换操作来实现

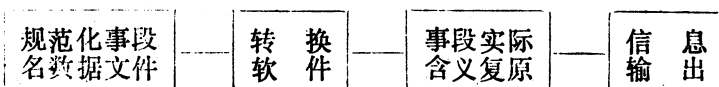


图4

图5所示为某部队材料仓库物资管理信息系统输出某舰艇零件库存报表的部分。图6所示为该报表数据在数据库中的存贮文件形式。

另件名称	X 舰艇另配件库存量						
	一月	二月	三月	四月	五月	六月	.....
主轴承	5	3	0	7	4	2	"
推进器	4	5	1	6	3	4	"
美人架	8	10	4	9	5	7	"
舵叶	11	8	7	12	9	10	"
.....	"	"	"	"	"	"	"

图 5

记录号	项 1	项 2	项 3	项 4	.....
0001	5	4	8	11	"
0002	3	5	10	8	"
0003	0	1	4	7	"
0004	7	6	9	12	"
0005	4	3	5	9	"
0006	2	4	7	10	"
.....	"	"	"	"	"

图 6

从上例中可以看到，使用字段项名规范化技术，非法用户即便打开了数据文件，亦难理解每个数据项所标识的真实内容。

### 3. 软件加密

人们往往只从保护软件产权的角度去考虑采取应用软件的防窃措施，其实应用软件的被窃直接影响到数据库中信息的安全保密性。这是因为非法用户能从研究分析应用软件中的内容获得窃取保密信息的技术手段，因此应用软件加密问题应受到足够的重视。

我们在开发数据库应用软件时采用以下二种软件保护措施。

(1) 一般来说软件难读性是一种有效的防窃手段。在缩写应用软件时尽可能采用可编译的宿主语言，这是因为经过编译后的可运行目标程序是很难读懂的，因而增加了分析软件功能的困难性。而编译后的应用软件的运行速度普遍比经过解释运行的软件快，所以这是一种一举两得的好方法。

(2) 在汉化CCDOS的支持下，利用汉字码对执行文件名进行加密处理。当文件名中包含了残缺的汉字码，其他人无法用一般的方法获得其完整的文件名，也就无法对该文件内容进行操作，这样就达到了对该文件内容进行加密的目的。

具体实现的办法：

A. 将需要保密的执行文件用半个汉字作为文件名。

例如：建立三个执行文件，它们分别为：

ヨ.PRG “ヨ”为半个“王”字



大.PRG “大”为半个“大”字

春.PRG “春”为半个“春”字

当建立文件的过程完成后，用DIR命令去列文件目录区记下的文件名时，显示或打印的内容为：

```
.PRG  
.PRG  
.PRG
```

这些文件名中的半个汉字都隐去了，这是因为汉字的内码由二个字节组成，且这二个字节的最高位均为“1”。当删除半个汉字后（左半个或右半个），相当于内码成为单字节，由于它的最高位为1，所以它既不是ASCII字符又不是汉字，系统无法显示和打印出它的标识内容。

B. 加密软件的复制，可以采用Copy命令实现，复制文件仍是加密的。

例如：Copy C: \*.PRG B:

Copy B: \*.PRG C:

C. 加密软件的执行采用调用的方式。

例如：要执行某个加密文件，可以调用过程文件

A.PRG来实现。A.PRG中的内容为：

```
CLEAR ALL
```

```
DO C: [.PRG] (运行加密文件)
```

```
RETURN 注：[ ]中内容可以省略。
```

以下列出我们开发的一个商业财务管理信息系统中的二个实际加密应用软件文件，图7为一个分支菜单，图8为一个执行过程文件

C: 1.PRG (文件名中含有半个汉字)

```
? " 童涵春药店计算机财务管理项目工作单(日)"
```

```
? " 1.凭证单据输入
```

```
2.打印传票"
```

```
? " 3.综合科目汇总表数据
```

```
4.打印科目汇总表"
```

```
? " 5.财务日常规处理
```

```
6.打印本日各类帐单"
```

```
? " 7.财务工作模块初始化"
```

```
ACCEPT"请选择工作:"To CN
```

```
STORE" "+CN To gz (" "中为半个汉字)
```

```
Do& gz
```

```
RETURN
```

图7

```
DO C: [.PRG]
```

```
? " 明细帐处理结束"
```

```

? " 现在处理现金, 银行日记帐"
DOC: [.PRG]
DOC: [.PRG]
? " 现金, 银行日记帐处理结束"
? " 现在处理科目结余"
DOC: [.PRG]
? " 科目结余处理结束"
? " 现在处理总帐"
DOC: [.PRG]
? " 总帐处理结束"
? " 日常规处理结束"
PETURN

```

图 8

#### 四、结束语

本文提出了几种能提高微机信息数据库安全保密性的有效措施。这些措施实施简便, 系统资源开销少。

```

C: 5.PRG
SET TALK OFF
? "日常处理"
? "现在处理明细帐"
DO C: [.PRG]

```

值得一提的是几乎所有的保密措施都要求系统提供一定的运行时间上和存贮空间上的资源, 也就是说, 保密要付出一定的代价。由于微机系统一般资源不很丰富, 所以使用一种保密性能虽好但资源消耗大的保密措施是不切合实际的。另一方面, 目前微机用户的计算机使用技术水准还不很高, 对承担过分复杂的保密操作也有困难, 所以对数据库中信息的保护手段的选择应该从可行性, 有效性, 合理的资源开销和简单的操作几个方面综合考虑, 才能为广大的微机管理信息系统的用户所接受。

文献出处: 《电脑应用时代》1988年3期38—42页

编者注: 本文还有《图1, 图2, 图3》等略

# 计算机信息保密方法简述

陈 付

大量的各类信息在计算机网内存贮和传送，这对网内用户的信息服务带来了极大的方便，效率也很高，但失密的可能性也增大了。

在一个计算机网络系统内，可能造成信息失密的部位是用户终端，系统控制中心和传输信道。造成失密的部位不同，使用的保密方法也有所区别。

## 用户终端的信息保密方法

### 一、防止用户采取不正当手段存取信息

1. 校验允许存取口令。所谓允许口令，就是在用户与系统之间预先规定的某种“口令”。用户利用系统时，先将自己的允许存取口令出示给系统，经系统检查与存贮的存取口令一致时才允许用户利用系统。

2. 用户终端权限限制。限制某些终端用户对信息资源的使用范围，以实现对其信息资源的保密。用户的权限等级是在系统登记时授予的。用户在利用系统的功能时，由系统校验其权限等级。

3. 存取控制矩阵和存取控制表。这是把所有用户建立的文件以及能否被其他用户使用的信息和使用方式（只读、读写、删改……），以矩阵或控制表的形式赋给系统。每当用户使用某文件时，系统会根据存取矩阵或存取控制表核实其使用权限。

4. 文件存取口令。对每个文件都规定允许存取口令，用户在存取文件时，系统可以根据这些口令来校验存取是否正当，从而实现文件的保护。

5. 登录。系统对每个终端用户进行跟踪登记，用户终端的任何存取操作都将被系统及登录下来，当发现非法存取时系统除拒绝服务外，还会向系统中心管理人员报警。这些跟踪登录可为事后的调查提供线索。

### 二、管理上的考虑

应加强用户终端设置场所的安全保卫措施，这对于防止信息被窃取，信息介质被偷窃有重要作用。同时，对终端工作人员的资格审查，以及职业上的纪律、道德约束都是非常重要的。

### 三、防幅射措施

终端设备的信息幅射，通常来源于键接触、磁介质读写、CRT显示刷新、打印，并与设备的主频、电路、结构有关。由于在终端上的信息是未经过加密的，现代技术很容易从幅射中获取大量的有用信息，这种信息幅射带来的失密不容忽视。

防止信息幅射的有效技术措施是对终端设备进行电磁屏蔽。在新建终端机房时，可采用机房整体屏蔽的方法，而在利用已建房屋改造成机房时，可安装组合式双层铜网屏蔽室。采用合理的屏蔽方法，即可有效地防止信息幅射，又可减少终端设置与外电场之间的相互干扰，改善终端的工作环境。

## 系统控制中心的信息保密方法

系统控制中心管理、存储着计算机网络中的大量信息，由系统信息失密造成的危害将是全局性的。

### 一、管理上的考虑

系统中心应设置在便于管理的特定场所，以利于保卫工作的开展。

系统中心的工作人员，由于拥有相当大的权限，又掌握着丰富的专门知识和具备各种技术手段，所以对系统的机密保护有决定性的作用，必须对其从管理上严格把关，即对他们进行严格的资格审查、有相应的纪律约束和明确的权限限制。

### 二、技术保密措施

1. 校验允许存取口令。系统中心的输入输出装置有被利用来对系统内的信息进行非法存取的可能性，因此对中心内的作业也必须严格校验允许存取口令。

2. 存储器保护和文件保护。存储器保护是将存储器划分成若干个区域，通过这些区域规定有关的存取属性（禁止写入、禁止读写、禁止执行及它们的组合使用）来实现的，这些属性由硬件进行校验，当违反存取保护属性时，便发出中断信息，控制系统采取终止程序执行等措施，从而对存储器信息给予保护。

文件保护的方法与前述文件存取口令相同。

3. 登录。对系统中心工作人员的操作，系统同样具有跟踪登录的功能，便于对系统中心人员监督和检查。

4. 信息加密。前述的各种信息保密方法，只是控制存取信息的一些手段，对信息本身并没有采取保密措施。所以只能在一定程度上起到信息保密的作用。防止信息失密的最重要的方法是对信息加密。

信息加密是围绕计算机系统内处理的信息本身实施信息保护的方法，所以说它是最本质的保密方式，它既适用于需要信息保密的各种领域，又能提高保密效果。

在信息发送端，按照预定的规则对信息加密借助加密信息进行传送和存贮。在信息接收端，按照同样的规则将信息译码后使用。只要密码化规则不泄漏出去，信息就能有效地得到保护。

加密信息也有被破译的可能，但只要使企图破译的代价足够高，也就可以有效地保护密码信息了，为此，有必要经常变更密钥。

信息加密通常用在重要机密需由通信线路传送及存入文件的情况，由专门的机要人员掌握实施。用硬件和软件均可实现对信息的加密。

信息加密已成了一项专门的学科，有各种复杂的技术和方法，这里不再赘述。

对系统中心的各类设备，同样存在着防电磁辐射的问题，不应忽视。

### 传输信道上的信息保密

连接终端和系统中心的通信信道是信息保密的重要环节。对传输信道不正当的存取的最大威胁是“窃听”，窃听一般难于被发现。对付窃听的唯一有效的对策是信息的密码化。当信息密码化后再进行传送时，即使被窃听，也能防止信息泄密。

目前信息加密技术的发展趋势是高强度、多层次加密，即使用算法错综复杂的密码和进行多次加密，从而使在信道上传输的是经过高度伪装的信息，很难破译。

文献出处：《计算机世界》月刊1987年10期38—39页

# 一些加密技术

贾建平

## 一、前言

随着计算机网的普及，人们对通讯的保密性要求越来越高。比如银行连网，要求网上传输的数据绝对保密。当前认识到的对网上传输信息的威胁主要有以下两种基本方式：

1. 窃听：窃听着可从网上利用数据传输设备获得信息。

2. 窜改：攻击者可以把网上传输信息改掉或换掉或用过期信息代替现传信息，以达到破坏通讯的目的。这在各种军事网上屡见不鲜。

对于上述破坏目前已有许多防止方法。本文也针对上述破坏进行了讨论。

## 二、加密方法

加密是对在潜在着敌手的环境中传送的信息的机密性进行保护的唯一实用的方法。在这种环境中，采用常规的物理手段来保护信息，要么是不可能的，要么是不切实际的。加密用数学变换方法控制数据，使得攻击者不能在攻击条件内从所传输的密文件还原出原文。为了使合法收方也能使用这些加密数据，收方应能够通过一逆转换方法迅速地将加密密文还原为原文。

一般加密是通过以一个称为密钥的数据和要传输的原文作为输入的算法来实现。（密钥是一个关键输入参数）图1是加密和脱密过程示意图。

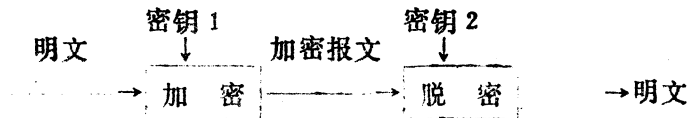


图1 加密过程

如果密钥1 = 密钥2就是常规加密法。这种加密法以美国1978年制定的数据加密标准（DES）为代表。这种加密法中加密密钥必须保密，通讯双方也必须事先知道密钥。因为这样的算法通信双方共有一个密钥，所以，密钥事先必须由信息通信的一方传给另一方，或从某第三方给双方。传送密钥又不能用明文传输，否则被敌方接受，加密算法就失去了意义。由此可见常规加密算法的密钥必须由人传送，这样就带来了麻烦，因为人可以说也不是一种保密的通讯信道。为此提出了公开密钥加密算法。

公开密钥算法中密钥1 ≠ 密钥2，密钥为一个公开的密钥（pk），可以让所有人知道；但密钥2为一个保密密钥（sk）。信息上传的经过加密（pk加密）的信息，只有拥有sk的用户才能对其脱密。

公开密钥算法应当具有下列特征：

1. 用户必须能有效地计算公开的和秘密的密钥对pk和sk。
2. 不允许在知道pk的情况下去有效地计算sk。

3. 继加密之后再脱密, 应能还原出原来的报文 (X), 即  $D_{sk}(E_{pk}(X))=X$

4. 用一段明文进行攻击的手段不能破译出用加密密钥pk加密的密文。

在传统加密算法中, 由于知道了加密密钥便也知道了脱密密钥, 因而其密钥可以以简单的方法随机选择 (在选择时, 也要考虑到某密钥的弱保密性), 但在公开密钥加密算法中, 公开密钥与秘密密钥间的关系是有意弄成迷惑不解的, 因而计算公开密钥与秘密密钥都必需特殊步骤。

公开密钥加密算法最富有代表性的是RSA与陷门渐缩算法。它们的算法都可由数学描述, 这里就不多说了。文献( )上有详细描述。

### 三、对窜改的预防。

前言中已说过, 在信息的传输过程中, 信息可能遭到有意或无意的窜改。

目前, 网络界正在研究的数字签名就是使得接收方在接收到信息后, 通过检验数字签名来判断是否是某一方发来的数据。

数字签名就是把加密算法的过程逆过来, 用脱密算法和私人密钥sk对信息加密, 收方用加密算法与公开密钥pk对数据进行脱密。因为只有一方掌握着这个私人密钥sk, 所以当用户收到经过sk加密后信息, 一定知道它来自掌握sk的一方。当然仅仅用数字签名并不能保证信息不被窃听, 因为任何人都知道公开密钥pk, 所以每个人都可以对用sk加密的信息进行破密。但是如果把数字签名和前面讲的加密法相结合会达到即能加密信息, 又能使接收方证实信息来自某一方。

其步骤如下: 某一甲方拥有保密密钥与公开密钥对 ( $sk_1, pk_1$ ), 乙方拥有密钥对 ( $sk_2$  和  $pk_2$ )。甲方要传送数据于乙方。甲方要事先用其sk对报文加密, 再用乙方公开密钥  $pk_2$  对其加密, 这样只有乙方才能脱密此密文。其过程示于图 2。

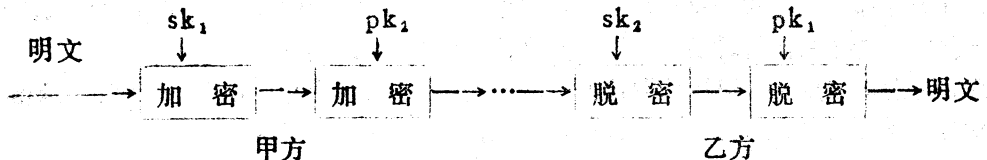


图 2 数字签名与加密

这样就可保证了文件在网上传输有了法律效力。

但仅仅上述加密, 依然不能解决信息在信道上被恶意破坏或敌对方重发网上的过期信息以扰乱收方正常工作。

通过鉴别技术, 就可让人们有高度把握去确定一串电文是否被改变了。鉴别是靠验证叫做鉴别码 (AC) 的位模式来实现的, 它是在假定数据或已知数据是正确的情况下, 事先计算好并附加到电文中去的。

鉴别码 (AC) 必须是报文 (TXT) 的函数 ( $\phi$ ) 并具有以下性质:

1. 对不同的报文  $TXT' \neq TXT$ , 敌手要计算  $\phi(TXT')$  是不可行的。
2. 对于一个敌手来说, 要找一个不同的报文  $TXT'$ , 使得  $\phi(TXT')$  等于  $\phi(TXT)$  这在计算上是不可行的。
3.  $\phi(TXT)$  在下面的意义下是均匀分布的: 对于  $TXT' \neq TXT$  时,  $\phi(TXT') = \phi(TXT)$

TXT) 的概率是  $1/2^C$ ; 其中  $C$  为  $AC$  的位数。

实现鉴别方案的一种方法是利用某种链接技术来获得错误传播特性而实现的。我们在这儿介绍的是另一种方案。

当有一串明文  $X_1, X_2, \dots, X_n$  要发送时, 利用检错技术中多项式 (CR(码)) 对  $X_1, \dots, X_n$  计算。计算结果得  $X_{n+1}$ , 最后把  $X_{n+1}$  附在明文后面得一报文  $X_1, \dots, X_n, X_{n+1}$ , 把此报文加密就得到密文  $X_1, \dots, X_n, X_{n+1}$ , 密文就可在信道上传输。

线路上传送报文  $Y_1, Y_2, \dots, Y_n, Y_{n+1}$ 。由 CRC 的性质及加密的保密性知经过加密的 CRC 计算机结果完全符合  $AC$  要求,  $AC$  可以用冗余校验代替。收方在收到  $Y_1, \dots, Y_n, Y_{n+1}$  后, 经过检测  $AC$  便可知信息中途是否被改变。

CRC 的冗余计算可用软件或硬件实现, 硬件实现速度较快些。在软件实现方面人们已经作了大量研究, 发现了一些快速计算方, 文献 (3) 就是一种。

当然, 经过上述加密, 数字签名、鉴别技术后, 依然不能保证敌对方不发送过期信息来破坏信息传输。防止过期信息的重发在银行系统政府文件及军事应用上是非常重要的, 为此提出了打烙印法。

所谓打烙印法就是把传输信息的时间或是第几次发送信息的次数加入到密文中去, 以便收方识别。这儿我们只介绍把传输信息的次数加入密文的方法。

设甲方第  $m$  次发送信息  $X_1, \dots, X_n$  于乙方, 信息  $X_1, \dots, X_n$  经过冗余校验后得  $X_{n+1}$ , 再对  $X_1, \dots, X_n$  加密后得  $Y_1, \dots, Y_n$ 。我们不能把  $m$  简单加在密文后面, 因为如果这样加密的话, 敌对方只须对报文分析, 把  $m$  段代码保留下, 其余换为过期信息, 便扰乱了信息传输。如何把  $m$  内容加入到密文中以解决上述问题呢?

本文是把  $m$  和 CRC 冗余计算结果加在一起  $m + X_{n+1}$  来解决问题的。

在发、收方都有一计数器, 当发方发出一信息于收方, 得到收方发来的信息已被接受指示后, 计数器加 1; 而收在得到经验正确的信息后, 其计数器也加 1, 并发已接受发来报文指示于发方。这样在任何时候, 发、收方保持有同样数值寄存器。

发方在发送信息时, 把计数器值  $m$  和 CRC 烙印计算结果  $X_{n+1}$  加起来, 得到一新值  $X'_{n+1}$ , 然后加密得  $Y_{n+1}$ , 与  $Y_1, \dots, Y_n$  一起发送出去。

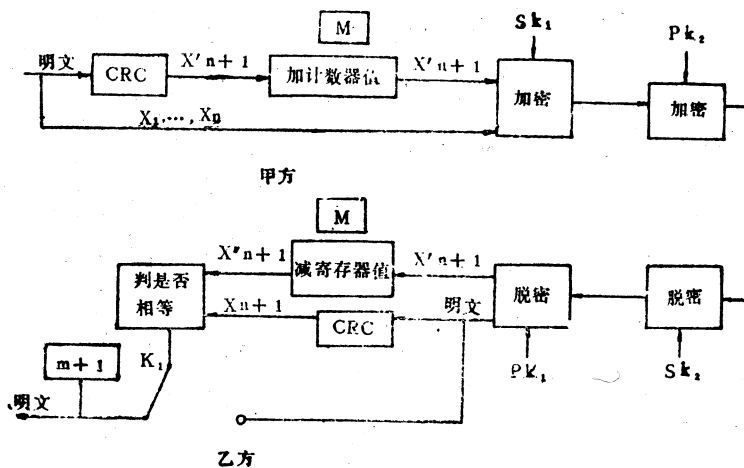


图 3 算法流程示意图

收方收到发来信息后, 把信息  $Y_1, \dots, Y_n, Y_{n+1}$  还原为  $X_1, X_2, \dots, X_{n+1}$ , 然后把  $X_{n+1}$  减去计数器值, 如果此值等于对  $X_1, \dots, X_n$  经过冗余运算结果, 便说明信息的确来自发方。否则, 拒绝接收。

整个加密、签名、鉴别、打烙印过程于图 3。

开关  $K_1$  当判断机构输出正确时关闭, 使明文通过, 并且使寄存器值加 1; 否则,  $K_1$  断开接收方拒绝接受信息。

#### 四、算法实现及结论

上述过程可用软件也可用硬件实现, 硬件实现会使加密速度非常快, 但要使用一定数量的器件。用软件实现速度要慢些。我们在实现时, 采用了软件方法, 为了使加密速度快些, 采用了汇编语言。在编写软件时对公开密钥算法的选择也相当重要, 我们在实现时, 选用了 RSA 法。

上面提出的算法完全适用于远程通讯。这样的算法就目前的技术水平来说, 具有很强的加密强度。当然还可以提出许多加密措施, 算法的过程也可以更加优化。这个算法的实现用硬件对将来更具有意义, 如有可能, 我们将着手于算法的硬件实现。

#### 参 考 文 献

1. 卡尔迈耶等著, 卢起骏等译, 计算机网络保密系统的设计与实现指南, 科技文献出版社重庆分社。
2. 远程信息处理业务中的密码学, 计算与密码 1987, 3
3. 计算机远程通讯中的快速 CRC 校验, 通讯与计算机, 1988, 1



# FA密码体制在数据完整性保护中的应用

中国科学院软件研究所 周同衡 白光野 谢丽华

**摘要:** FA密码体制是一种建立在自动机理论基础上的时序密码体制, 本文介绍了怎样应用FA密码体制保护数据的完整性, 我们认为在数据完整性保护方面, 它比其它密码体制, 比如DES更方便。

## 一、前言

在信息的传输和存储中的一个重要问题, 是信息的保护问题。信息的保护包括两个方面的内容, 一是信息的隐密性的保护, 另一是信息的完整性的保护。前者是密码学的传统课题, 后者是对密码学提出的新课题。所谓信息的隐密性保护是大家比较熟悉的, 它指未经许可的人不能了解所传输或存储的信息的内容。而信息的完整性保护是指不容许对传输中或存储中的信息进行篡改, 伪造, 防止用过期的信息假冒当前信息, 以及对通信者, 存储者身份的鉴别等。

信息完整性保护涉及很多复杂的问题, 对通信者, 存储者身份的鉴别, 如数字签名, 电子合同等, 都是密码学应用中的一些专门领域, 它们还涉及许多法律上的问题, 本文难以详述。在这篇文章中, 我们着重讨论对传输和存储中的信息防止篡改和伪造的问题, 即数据和系统整体性保护的问题, 以及一类新型的密码体制——FA密码体制在这种保护中的应用。

## 二、数据和系统的整体性保护

在本节中, 我们对数据和系统的整体性保护问题作一个说明, 并介绍目前采用的一些保护方法。

设  $Z_{2^n} = \{(a_1, \dots, a_n) | a_i \in \{0, 1\}\}$  表示所有  $\{0, 1\}$  上  $n$  维向量或  $n$  长序列所组成的集合。

数据的隐密性保护和数据的完整性保护是两个不同的范畴。对于数据的隐密性保护并不能取代对数据的完整性保护。

如果A、B双方采用DES并以约定的密钥K进行通讯。A方发出的报文是

$$X = X_1 X_2 \dots X_n$$

$$X_i \in Z_{2^{\dots}}$$

B方收到的加密后的报文是

$$Y = Y_1 Y_2 \dots Y_n$$

$$Y_i \in Z_{2^{\dots}}$$

其中

如果在传输过程中, 加密后的报文Y被未经允许的第三方截收, 这个第三方虽然不能了解X的内容, 但它可以采用多种方法反A、B双方的通信进行破坏。第一, 它可以对传输中的信息进行篡改, 如果它以Y'来代替Y

$$Y' = Y_1 Y_2 \dots Y_i Y_{i+1} \dots Y_n$$

或

$$Y' = Y_1 Y_2 \dots Y_i Y_{i-1} \dots Y_n$$

B方在收到Y'后, 进行解密得到X'

$$X = X_1 X_2 \dots X_i \dots X_n$$

或

$$X' = X_1 X_2 \dots X_i X_{i-1} \dots X_n$$

B方并不一定能察觉X'不是A方发出的原报文。

第二, 它可以用A方过去用密钥K发出的报文Y来代替Y

$$Y = Y'_1 Y'_2 \dots Y'_m$$

其中,

$$Y'_j = \text{DES}(KX'_j)$$

$$X' = X'_1 X'_2 \dots X'_m$$

X'是过期报文, B方在收到Y后也不能察觉出Y是过期报文。对于存储中的信息也可能发生类似的现象。

为了防止这类现象的发生, 必须加强通信和存储系统的完整性保护。我们用S来表示一个通信系统或一个信息存储系统中的文件系统。信

$$S = \{X^1, X^2, \dots, X^l\}$$

对于通信系统来说,  $X^1, \dots, X^l$ 即表示通信双方交换的信息——报文; 对于一个信息存储系统来说,  $X^1, \dots, X^l$ 即表示系统中存储的信息——文件。要保证系统的完整性, 必须保证系统能察觉对于系统中每个文件(或报文) $X^i$ 的改变, 并且能察觉对于文件(报文)次序的改变。我们把前者叫做文件(或报文)的整体性保护, 把后者叫做系统的整体性保护。

我们使用一个函数SN来保护文件 $X^i$ 的整体性。SN以S中的文件 $X^i$ 为输入, 结果是一个数字。

$$\text{SN}: S \rightarrow M$$

其中M是一个定长数字或不定长数字的集合。

对于S中的每个X, 附加SN( $X^i$ )。形成

$$X^i, \text{SN}(X^i)。$$

函数SN满足条件:

① $X^i$ 中的任何改动, 使SN( $X^i$ )不发生改变的可能性是很小的。或者, 我们换句话说, 对于 $X^i \neq X^{i'}$ ,  $P(\text{SN}(X^i) = \text{SN}(X^{i'}))$ 是很小的。

②对于通讯者或存储者,  $X^i$ 和SN( $X^i$ )之间的关系是可以检验的。即是说, 在知道SN的情况下, 对于任意的 $X^i$ , SN( $X^i$ )是容易计算的。

③对于不了解函数SN的人, 即使他掌握了很多资料, 如大量的 $X^i$ , SN( $X^i$ )对, 对于新的 $X^i$ ,  $X^i \neq X^j$ , 也很难构成相应的SN( $X^j$ )。

对于这样的函数SN，在文件 $X_i$ 后面加上相应的 $SN(X_i)$ ，这样即能保证文件 $X_i$ 的整体性。如果我们每个文件 $X_i$ 再加上的时间或次序的信息， $Z_i$ ，相应的 $X_i, Z_i, SN(X_i Z_i)$ 则能保证系统的整体性。

SN的作用有类似于纠错编码的地方，但SN构成的秘密性又具有密码学的特征。

例如，我们可以采用DEC的CBC方式或其它方式来构成所需的SN函数。

设文件 $X_i$ 为

$$X_i = X_1 X_2 \dots X_n \quad X_i \in Z_2^{264}$$

对于通信双方或存储者约定的密钥K，我们有

$$DY_1 = \text{DES}(K_1; X_1)$$

$$Y_2 = \text{DES}(K, X_2 \oplus Y_1)$$

.....

$$Y_n = \text{DES}(K, X_n \oplus Y_{n-1})$$

$$Y_{n+1} = \text{DES}(K, Y_n)$$

取

$$SN(X_i) = \text{DES}(K, Y_n) = Y_{n+1}$$

或者

$$Y_1 = \text{DES}(K, X_1)$$

$$Y_2 = \text{DES}(I(Y_1), X_2)$$

.....

$$Y_n = \text{DES}(I(Y_{n-1}), X_n)$$

$$Y_{n+1} = \text{DES}(K, Y_n)$$

其中

$I(Y_1)$ 是从Y中抽取56位的一个函数。我们取

$$SN(X_i) = Y_{n+1}$$

在 $X_i$ 中发生一位变化时， $SN(X_i)$ 不发生变化的可能性大约是 $n/264$ 。这种可能性是非常小的。同时SN也是很容易检验的，并且由于DES本身的安全性，在不知道密钥K的情况下，对于任意的文件 $X_i$ ，要构成 $SN(X_i)$ 是很困难的。

### 三、关于FA密码体制

FA密码体制是陶仁骥在研究有限自动机可逆性问题基础上提出的一种新型密码体制，这是一种采用密文反馈的时序密码体制。它的结构如图1。

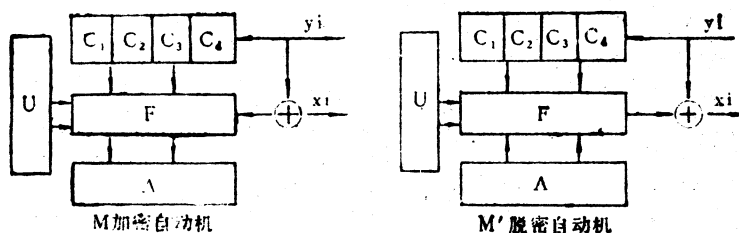


图1 FA密码体制

其中  $A = \langle Y_A, S_A, \delta_A, \lambda_A \rangle$  是一个64位的线性移位寄存器, 叫做密钥线性寄存器。  
 $Y_A$  是  $A$  的输出字母集,  $S_A$  是  $A$  的状态集,  $Y_A = S_A = Z_{2..64}$ .

$\delta_A: S_A \rightarrow S_A$ , 是  $A$  的状态转移函数。

$\lambda_A: S_A \rightarrow Y_A$ , 是  $A$  的输出函数。

一般情况下, 我们选择  $\delta_A$ , 使  $A$  产生一个周期为  $2^{64}-1$  的  $m$  序列。并且选

$$\lambda_A(x) = x, \quad x \in Z_{2..64}.$$

$U = \langle Y_U, S_U, \delta_U, \lambda_U \rangle$  是一个32位的线性移位寄存器, 叫做工作密钥性移位寄存器。其中,  $Y_U$  是  $U$  的输出字母集,  $S_U$  是  $U$  的状态集。

$\delta_U: S_U \rightarrow S_U$ , 是  $U$  的状态转移函数。

$\lambda_U: S_U \rightarrow Y_U$ , 是  $U$  的输出函数。

一般情况下, 我们选择  $\delta_U$ , 使  $U$  产生一个周期为  $2^{32}-1$  的  $m$  序列。并且选

$$\lambda_U(x) = x, \quad x \in Z_{2..32}$$

而  $C_1, C_2, C_3, C_4$  是 4 个 8 位的寄存器。

$F: Z_{2..128} \rightarrow Z_{2..8}$ , 是一个从  $Z_{2..128}$  到  $Z_{2..8}$  的函数。

$M = \langle X, Y, S, \delta, \lambda \rangle$  是一个有限自动机,  $M' = \langle Y, X, S, \delta', \lambda' \rangle$  是  $M$  的逆自动机。我们把  $M$  叫做加密自动机,  $M'$  叫做脱密自动机。

其中,  $X$  是明文字母集,  $Y$  是密文字母集。

$$X = Y = Z_{2..8}.$$

$S$  是  $M, M'$  的状态集。

$$S = Z_{2..128}$$

$$S = \{ (s_A, s_U, c_1, c_2, c_3, c_4) \mid s_A \in S_A, s_U \in S_U, c_1, c_2, c_3, c_4 \in Z_{2..8} \}$$

$\delta: S \times X \rightarrow S$ , 是  $M$  的状态转移函数。

$\delta': S \times Y \rightarrow S$ , 是  $M'$  的状态转移函数

而

$\lambda: S \times X \rightarrow Y$ , 是  $M$  的输出函数。

$\lambda': S \times Y \rightarrow X$ , 是  $M'$  的输出函数。

对于明文序列  $x_0, x_1, \dots$ , 和初始状态

$$s_0 = (s_{A0}, s_{U0}, c_1, c_2, c_3, c_4) \in S$$

$$\delta(s_0, x_0 x_1 \dots) = y_0 y_1 \dots$$

如果我们记  $c_1 = y_{-4}, c_2 = y_{-3}, c_3 = y_{-2}, c_4 = y_{-1}$ 。对于任意的  $i, i \geq 0$

$$\delta(s_i, x_i)$$

$$= \delta^{i+1}(s_{A0}), \delta^{i+1}(s_{U0}), y_{i-3}, y_{i-2}, y_{i-1}, y_i$$

其中  $\delta^{i+1}(s_{A0})$  定义为, 当  $i=0$  时

$$\delta_A(s_{A0}) = \delta_A(s_{A0})$$

$$\delta^i_A(s_{A0}) = \delta_A(\delta^{i-1}_A(s_{A0})).$$

$\delta^{i+1}(s_{U0})$  也同样定义。

$$\lambda(s_i, x_i)$$

$$= F(\lambda_A(\delta^i_A(s_{A0})), \lambda_U(\delta^i_U(s_{U0})), y_{i-4}, y_{i-3}, y_{i-2}, y_{i-1}) \oplus x_i$$

$$= y_i.$$

而

$$\begin{aligned} \delta'(s_i, y_i) &= \delta_{A^{+1}}(s_{A_0}), \delta_{U^{+1}}(s_{U_0}), y_{i-3}, y_{i-2}, y_{i-1}, y_i) \lambda_i(s_i, y_i) \\ &= F(\lambda_A(\delta_{A^{+1}}(s_{A_0})), \lambda_U(\delta_{U^{+1}}(s_{U_0})), y_{i-3}, y_{i-2}, y_{i-1}, y_i) \oplus y_i. \end{aligned}$$

FA密码体制是一种译码误差传播有界的时序密码体制。它具有保密性能好，结果很灵活，容易构造，便于用软件、硬件实现的优点。适当的选择A, U, C及F, 我们可以构成多种密级的密码体制。本文由于题目的限制，不作这方面的讨论。下面我们即讨论FA密码体制在数据完整性保护方面的应用。

#### 四、FA密码体制在数据完整性保护中的应用

在前面我们已经讨论过信息的隐密性保护和信息的完整性保护是两个不同的范畴。因而在下面讨论文件Z的整体性保护时，可以不考虑文件Z的保密问题，而仅仅考虑函数SN的构成。

对于用上述结构构成的FA密码系统，一般来说都是编码误差传播无界的。

对于选定的某个F, 如果存在一个最小的常数K, 使得对于任意的

$$S_0 = (s_{A_0}, s_{U_0}, c_1, c_2, c_3, c_4) \in S$$

及明文序列X

$$\begin{aligned} X &= x_1 x_2 \dots x_i \in Z_{2..8} \\ \delta(S_0, X) &= y \end{aligned}$$

其中

$$y = y_1 y_2 \dots, y_i \in Z_{2..8}.$$

并且对于任意的 $X' = x'_1 x'_2 x'_3 \dots$ , 其中 $x'_i \neq x_i$ .

有

$$\delta(S_0, X') = y'$$

有

$$y' = y'_1 \dots y'_k y_{k+1} y_{k+2} \dots$$

即编码时的误差仅影响后面K个字节。

由于我们假设K是满足上面条件的最小常数，因而总存在 $S_0, X, X'$ , 使

$$y' = y'_1 \dots y'_k y_{k+1} \dots$$

中

$$y'_k \neq y_k$$

因为

$$\begin{aligned} y_{k+3} &= F(\delta_{A^{+3}}(s_{A_0}), \delta_{U^{+3}}(s_{U_0}), y_k, y_{k+1}, y_{k+2}, y_{k+3}) \oplus x_{k+3} \\ &= F(\delta_{A^{+3}}(s_{A_0}), \delta_{U^{+3}}(s_{U_0}), y'_k, y_{k+1}, y_{k+2}, y_{k+3}) \oplus x_{k+3} \end{aligned}$$

推出

$$\begin{aligned} &F(\delta_{A^{+3}}(s_{A_0}), \delta_{U^{+3}}(s_{U_0}), y_k, y_{k+1}, y_{k+2}, y_{k+3}) \\ &= F(\delta_{A^{+3}}(s_{A_0}), \delta_{U^{+3}}(s_{U_0}), y'_k, y_{k+1}, y_{k+2}, y_{k+3}) \end{aligned}$$

由于

$$y_i = F, \oplus x,$$

我们只要选择F, 使它对于任意的  $s_A \in S_A$ ,  $s_U \in S_U$ , 及  $a \in Z_{2,8}$ , 使存在  $b_1, b_2, b_3, b_4 \in Z_{2,8}$ , 有

$$\begin{aligned} & F(s_A, s_U, a \oplus b_1, b_2, b_3, b_4) \\ & \neq F(s_A, s_U, b_1, b_2, b_3, b_4). \end{aligned}$$

对于这样选择的F所构成的FA密码系统, 是编码误差传播无界的。

而对于一般情况下所选择的F, 它们都能满足上面条件的要求, 因而我们可以利用FA密码体制的这种编码误差传播无界的性质来构成文件的SN函数。

在构造FA密码系统时, F的选择使它满足这样的要求, 对于任意的  $c_{A0}, s_{U0}, c_1, c_2, c_3, c_4$ , 和明文序列X

$$X = x_1 x_2 \dots x_i \in Z_{2,8}$$

$$\delta((s_{A0}, s_{U0}, c_1, c_2, c_3, c_4), X) = Y = y_1 y_2 \dots$$

Y总是近似于一个随机序列, 这是对于密码系统的安全性所必需的。

因此, 对于任意的  $S'_A, S'_U \in S_A, S'_U \in S_U$  以及  $C = (c_1, c_2, c_3, c_4), C' = (c'_1, c'_2, c'_3, c'_4), C \neq C'$ ,

$$P(F(s_A, s_U, c) = F(s'_A, s'_U, c')) = \frac{1}{265}$$

而对于任意的  $s_0 \in S$ , 以及  $x_1, x_2, x_3, x_4, x'_1, x'_2, x'_3, x'_4 \in Z_{2,8}$ .

$$\lambda(s_0, x_1 x_2 x_3 x_4) = \lambda(s_0, x'_1 x'_2 x'_3 x'_4)$$

的可能性因为

$$\begin{aligned} & \lambda(s_0, x_1 x_2 x_3 x_4) \\ & = \lambda(s_0, x_1) \lambda(s_1, x_2) \lambda(s_2, x_3) \lambda(s_3, x_4) \\ & = \lambda(s_0, s'_1) \lambda(s'_1, x_2) \lambda(s'_2, x_3) \lambda(s'_3, x_4) \\ & = \lambda(s_0, x'_1 x'_2 x'_3 x'_4) \end{aligned}$$

其中

$$s_i = \delta(s_{i-1}, x_{i-1}), 1 \leq i \leq 3$$

$$s'_i = \delta(s'_{i-1}, x'_{i-1}), 1 \leq i \leq 3$$

所以

$$P(\lambda(s_0, x_1 x_2 x_3 x_4) = \lambda(s_0, x'_1 x'_2 x'_3 x'_4)) = \left(\frac{1}{256}\right) = 2^{-32}$$

对于文件  $X = x_1 x_2 \dots x_N$  如果在其中发生了变动, 比如在第一个字节处发生了变动, 得到一个新文件  $X' = x_1 x_2 \dots x_N, x_1 \neq x'_1$ .

由于  $y_{i+4} = F(\delta_{A_i+3}(s_{A_0}), \delta_{U_i+3}(s_{U_0}) y_i, y_{i+1}, y_{i+2}, y_{i+3}) \oplus x_{i+4}$ , 因而如果有

$$y_i y_{i+1} y_{i+2} y_{i+3} = y'_i y'_{i+1} y'_{i+2} y'_{i+3}$$

则有

$$y_{i+4} \dots y_N = y'_{i+4} \dots y'_N.$$

所以我们得出, 对于任意的  $S_0 \in S$ , 及初始向量Z,

$$P(SN_{soz}(X) = SN_{soz}(X'))$$

$$\begin{aligned} & \| P(y_1 y_2 y_3 y_4 = y'_1 y'_2 y'_3 y'_4) + P((y_1 y_2 y_3 y_4 \neq y'_1 y'_2 y'_3 y'_4) \cap (y_2 y_3 y_4 y_5 \\ & = y'_2 y'_3 y'_4 y'_5)) + \dots + P((y_{N+1} y_{N+2} y_{N+3} y_{N+4} = y'_{N+1} y'_{N+2} y'_{N+3} y'_{N+4}) \cap (y_1 y_2 \\ & \dots y_{N+3} \neq y'_1 y'_2 \dots y'_{N+3})) \\ & = 1/2^{32} + \left(1 - \frac{1}{2^{32}}\right) \frac{1}{2^{32}} + \dots + \left(1 - \frac{1}{2^{32}}\right)^{N+3} \frac{1}{2^{32}} \\ & < \frac{N+3}{2^{32}}. \end{aligned}$$

而且对于任意的文件X, 我们随意选择一个32位的0,1序列来作为它的SZ, 符合 $SN_{soz}(X)$ 的可能性为 $1/2^{32}$ .

因此, 对于文件X中发生的改动, 通过 $SN(X)$ 来察觉的可能性很大的.

由于这种SN是由密码构成的, 密码体制本身的安全性即保证了SN的安全性, 对于不知道所采用的 $S_0$ 的人, 是其难成SN的.

SN的长度受到密码体制本身的限制. 因为我们在上面所述的密码体制中所选的C寄存器的长度为32位, 所以我们选择SN的长度为32位. 如果我们要增加SN的强度, 必须对密码体制作相应的修改. 如果我们需要64位长的SN, 可将上述的C寄存器的长度改为64位. 不对密码体制进行修改, 仅靠增加SN的长度是不够的.

如果我们取

$$Z = Z_1 Z_2 \dots Z_s, \quad Z_i \in Z_2 \dots$$

定义

$$SN_{soz}(X) = y_{N+1} \dots y_{N+s}$$

因为

$$\begin{aligned} y_{N+s} &= F(\delta_A^{N+4}(SNO), \delta_U^{N+4}(SUO), y_{N+1}, y_{N+2}, y_{N+3}, y_{N+4}) \\ & \oplus x_{N+s} \end{aligned}$$

其中

$$x_{N+s} = Z_s$$

如果有

$$y_{N+1} y_{N+2} y_{N+3} y_{N+4} = y'_{N+1} y'_{N+2} y'_{N+3} y'_{N+4}$$

则有

$$y_{N+5} y_{N+6} y_{N+7} y_{N+8} = y'_{N+5} y'_{N+6} y'_{N+7} y'_{N+8}$$

因而 $y_{N+4}$ 以后的数字并没有检验作用.

在FA密码体制中具有两个密钥线性移位寄存器, 一个叫做密钥线性移位寄存器A, 一个叫做工作密钥线性移位寄存器U. 前者是用于密钥保护的, 它的初始状态 $S_A$ . 即是通信双方的密钥, 是必须保密的; 后者的初始状态 $S_U$ , 叫做工作密钥, 是可以公开的. 比如, 我们可以选择报文的时间、序号来作为工作密钥. 它的作用是增强对密钥和系统的保护.

由于我们选择用的U是一个能产生m序列的线性移位寄存器, 对于 $S_U \neq S'_U$ ,

$$\delta_i U(S_U) \neq \delta_i U(S'_U)$$

因为

$$SN(S_A, S_U, c), z(X) = y_{N+1} y_{N+2} y_{N+3} y_{N+4}$$

$$SN(S_A, S'_U, c), z(X) = y'_{N+1} y'_{N+2} y'_{N+3} y'_{N+4}$$

其中

$$y_i = F(\delta_{A_i}^{-1}(S_{A_0}), \delta_{U_i}^{-1}(s_{U_0}), y_{i-4}, y_{i-3}, y_{i-2}, y_{i-1}) \oplus x_i$$
$$y'_j = F(\delta_{A_j}^{-1}(s_{A_0}), \delta_{U_j}^{-1}(s_{U_0}), y'_{j-4}, y'_{j-3}, y'_{j-2}, y'_{j-1}) \oplus x_j$$

因而

$$P(SN(s_{A_0}, s_{U_0}, c), z(X) = SN(S_{A_0}, S'_{U_0}, c), z(X))$$
$$= \frac{1}{2^{82}}$$

这样，报文或文件X的时间或序号的变动也将引起SN的改变，对于

$$X_i' = (X_i, s_{U_0}, SN(s_{A_0}, s_{U_0}, c), (X_i))$$

这样构成新文件 $X_i'$ ，是允许外人随便改变其内容和它在整个文件系统中的位置。这样，我们即构成了对于文件的整体性及系统的整体性的保护。

用密码来对数据完整性保护，是数据完整性保护的一种方式。当然我们还可以采用很多其它的方法来进行这种保护。在用密码保护数据完整性时，各种密码体制都可以用来构成相应的SN。但是，FA密码体制的结构即非常适应这种数据完整性保护的需要。它可以在不增加设备，不增加额外操作的条件下，很自然地形成所需的SN。并且它本身的强度即保证了这种SN的安全性。

#### 参 考 文 献 (略)

文献出处《计算机研究与发展》1988年第1期15-22页



# 磁盘如何“加锁”？

J. Voelcker等著 李淑卿 程伯武 编译

自个人计算机(PC)问世以来,对一个合法的用户来说,复制软件包是正常的事。1976年美国版权法就规定:为存档或为防止电子以及机械方面的损坏而进行软件复制是合法的。有些软件厂商也以自己的软件系统能使用户方便复制文件而感到自豪。复制的过程很简单:把一个空盘放在一个驱动器上,把有需要程序的软盘放在另一个驱动器上,从键盘上敲入复制命令,马上就可以得到与源程序相同的拷贝。有的计算机操作系统命令更方便,单盘也能复制。复制的文件可以是厂商提供的系统程序、软件包,也可以是用户程序。毫无疑问,这种复制文件的作法给用户带来了极大的方便。但怎样才能制止用户为朋友或出售而进行额外的非法复制呢?

1976年,Microsoft公司的现任主席Bill Gates给一家计算机杂志写了一封信,抱怨一些人复制了他为MITS Altair写的一盘Basic解释程序的纸带,售价500美元。

1985年1月,Future Computing公司的Richardson估计,对每个允许合法复制的商品软件都存在非法的复制品。对于畅销的软件包,某软件公司的经理估计,非法复制的比例可高达6或7比1。尽管有许多技术措施保护软件,但这个比例仍未降低。

面对成千上万的程序员,许多软件公司为保护软件不被复制,大力研究软件保护技术。两个最知名的软件保护技术的研究者是Vault公司的Prolok和Softguard Systems公司的Superlok。

在发展复制保护方案的同时,产生了一些反保护的方法。一些公司已经销售了一些软件包,它们的用途是专门复制那些有保护的软件。虽然任何加上保护的软件都可以被非法复制,但保护的方法的确减慢了非法复制品的流行。

总而言之,软件的复制反复制斗争在用户和软件出售商之间一直没有间断。软件出售商不断研究一些保护措施,以免他们的软件被人非法复制,而一些用户则不断采取一些对抗措施。

## 一、磁盘保护技术及对抗措施

复制保护的原理很简单:使用特殊的技术把信息写到软盘上,普通PC的软盘驱动器可以把这些信息读出来,但不能写。当用户要复制该磁盘时,那个信息就会丢失。嵌入到应用程序里的软件可以检查这个独特的数据,如果该信息存在,则此盘是原件,否则是复制品,则程序中止。

制造厂家磁盘复制机构能把信息写到磁盘上,而普通的PC机构为什么就不能呢?简单地说,实际记录到磁盘上的是一系列的磁通变化,而PC机磁盘控制器要确定这些磁通变化的意义,某些是数据,另一些是同步标志,还有一些可能刚好是随机无用信息。虽然磁盘控

"How disks are 'padlocked'", IEEE Spectrum, June 1986, pp.32-40

制器一次可在一有限格式的范围里写一个扇段的数据，但磁盘复制机构可以把磁通变化按任何希望的次序放在盘的任何位置上。

典型磁盘上的数据排列成一系列的同心圆，叫磁道，每个磁道上的数据划分成扇段（如图1）。每个扇段包含一个标头段，一个数据段和一个在读数据时检测错误的检测和。一个典型的磁盘象IBM PC，每面上有40个磁道，每一磁道上有8或9个扇段，每一扇段通常包含256或512个字节。

每个磁盘上还有一个索引孔，以确定扇段的位置。今天，大多数PC机使用的索引孔只用来确定第一个扇段的位置，而之后的扇段根据第一个来确定，这就是所谓的“软分段”（soft sectoring）方式。早期的磁盘驱动器，软盘是“硬分段”（hard sectoring）的，它用一个索引孔只指示一个扇段的起始位置。

防止磁盘复制的一个最简单方法是在磁盘上写一个或多个“坏”扇段，在这些扇段中，检测和与写入的数据不匹配。当标准的复制程序遇到这些有“缺陷”的扇段时，便停止复制，因为它不能正确地将源盘中的数据读出；或者用“正确的”检测和把数据写到目的盘上。在这两种情况下，目的盘上都无坏扇段。

当保护程序运行时，嵌入的复制保护软件仅仅要求来自坏扇段的请求信息。如果读操作正常进行，该软件终止这一程序。如果磁盘控制器返回一错误代码，则复制保护软件允许继续执行。

这种办法今天难以继续使用，因为修改了标准复制程序，它不识别扇段的好坏，只是按位或按半字节复制。

坏扇段的另一种方法是使用一种专门的技术擦掉一小块铁氧体记录介质。这样产生了一个必须由硬件复制的坏扇段。保护软件把数据写到改变了的扇段上，那么，试图读这些数据时就会返回。而完好扇段的数据字节能正确地读出。但当读/写头接触到改变了的扇段时，同步驱动器的时钟位和数据位一起丢掉，这样驱动器便中断了其余扇段的数据读出动作。如果这个数据可由磁盘驱动器读出，那么认为这个软件已从源盘复制了。

自第一个坏扇段磁盘出现以来，还发展了许多其它复制保护技术，当然相应的对抗措施也应运而生。现简介如下。

### 1. 额外磁道技术（见图2）

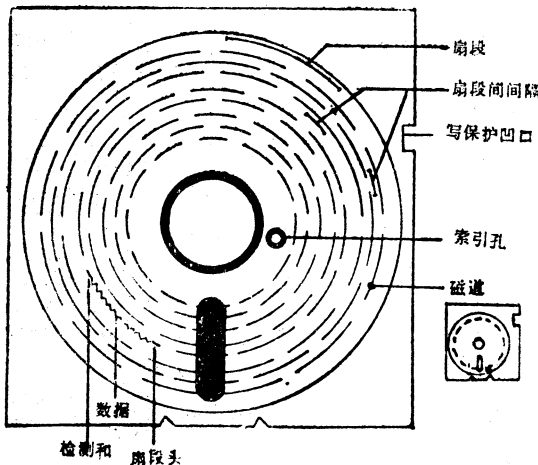


图1 磁盘的一般结构

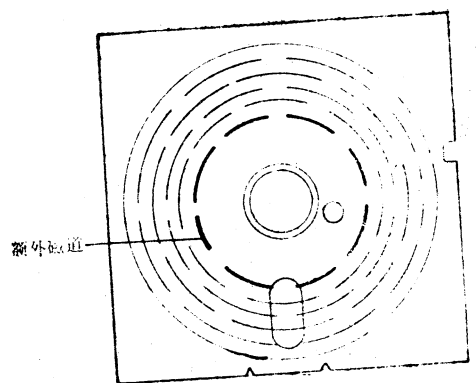


图2 额外磁道技术

这种磁盘的磁道数比标准磁盘的磁道数多(标准磁盘每面有40个磁道),因此用传统的复制程序不可能复制它。如果用传统的技术复制这种磁盘,那么额外磁道上的信息就必然会丢失掉。而如果程序的一部分保存在这些额外磁道上,那么无论怎么修补也不可能使拷贝正确地工作。因此,额外磁道是防止传统软件复制的有效方法。

对抗措施 设计“灵巧”的复制程序,它可以象寻找其他磁道一样找到额外磁道并加以复制。

## 2. 额外扇段技术(见图3)

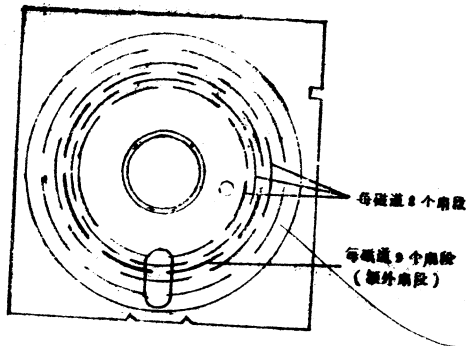


图3 额外扇段技术

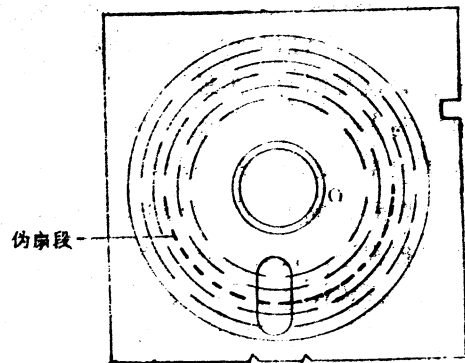


图4 伪扇段技术

IBM PC标准磁盘每面40个磁道,每个磁道8或9个扇段。通过控制扇段之间间隔的长度,特别是在可有更多空间存贮数据的外层磁道,软件生产者可以挤入10个甚至11个扇段。传统复制程序发出的命令是在每个磁道上读、写0到8扇段,因此无疑会丢失插入的任何额外扇段。

对抗措施 由于额外扇段与其它扇段都一样,因此可以设计一种检查额外扇段的程序,以便读和复制他们。

## 3. 伪扇段技术(见图4)

伪扇段技术可以阻止“知道”磁道上额外扇段的复制程序。伪扇段技术在磁道上按顺序放若干扇段头,后边不接数据。由于复制程序要在其缓冲器分配足够的存贮器为这些伪扇段保存标准扇段的数据值,因此它很快运行出缓冲器空间。

对抗措施 设计能够检查是否每个扇段头跟着数据的程序,它可以毫不费力地复制伪扇段。

## 4. 螺旋型磁道技术(见图5)

大多数标准磁盘在一些同心磁道上写数据。要从一个磁道移到另一个磁道,磁盘控制器控制磁头步进,等待它稳定到某一位置,然后读新数据。而螺旋型磁道技术是在磁盘上搞一些螺旋型磁道,在读/写数据时使磁头步进,这样就完全扰乱了传统的复制程序,使磁盘无法复制。

对抗措施 通过试错,可以确定磁盘上螺旋型磁道的位置和螺距,并实现磁盘复制。但

是要成功地读螺线型磁道，须掌握步进速率和转动速度之间的精确比率，这一点不太容易，因此这种技术相对来说不太可靠。

### 5. 超级扇段技术 (见图 6)

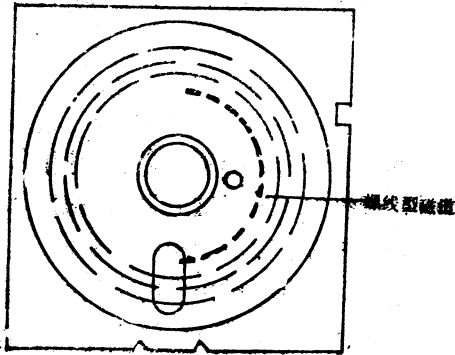


图 5 螺线型磁道技术

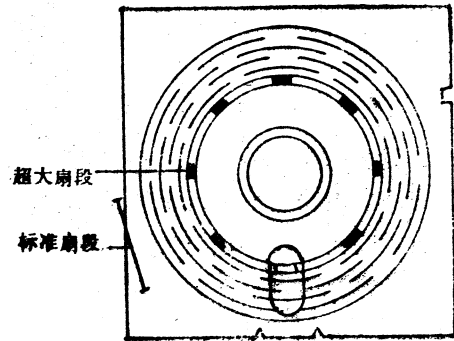


图 6 超级扇段技术

今天，人们往往使用一种超级扇段——也称为接续写 (write-splicing) 的技术作为保护软件的基本方法。这种方法由 Softguard 系统公司研制，它使用市售磁盘复制设备上写一些甚大扇段 (每个扇段接近一个磁道)。磁盘控制器不能写这些扇段，但可以在程控条件下读这些超大扇段。通过把特定数据放在各扇段之间的间隙中，并作为超大扇段的一部分读出，软件制造者们就提供了一种区别原始磁盘和复制盘的方法。传统的复制程序不能写扇段之间的数据，因此这些额外的信息将会损失掉。

**对抗措施** 通过删除检测代码可以战胜接续写的这种技术。Central Point Software 公司还研制了一种磁盘控制器板，只要控制器板发现什么地方和什么时候有磁变换，它就复制这种磁变换，而不管信息的内容是什么。

### 6. 扇段对齐技术 (见图 7)

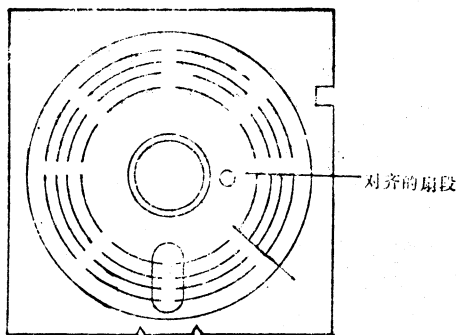


图 7 扇段对齐技术

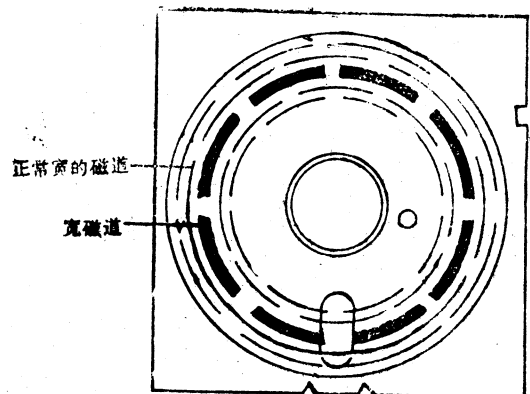


图 8 宽磁道技术

软分段软盘带有表示每个磁道上扇段开始的信号，因此磁盘驱动器只需检查磁盘索引孔的存在以标明每个扇段的开始。传统的复制程序读到磁道时就把它写到目的盘上，不管磁道与磁道是否对齐。然而磁盘复制设备是以相对于索引孔的精确位置把相应的扇段写到每个

磁道中。

保护程序可以通过读一磁道上的某个扇段，步入下一磁道，等待确定的时间，读它找到的下一扇段的号码来检查是否对齐。扇段的对齐和磁盘的转动速度精确地确定了数据应该放在什么地方。

对抗措施 灵巧的复制程序可以测量磁道到磁道的扇段对齐，但这种软件控制不象硬件调节那么准确，因此第二、第三次复制件越来越不可靠。

### 7. 宽磁道技术 (见图 8)

虽然传统的磁盘驱动器的读/写磁头相对比较窄，外侧有消磁头可把不同磁道数据间的干扰减到最小。但市售磁盘复制机构配有宽磁头，能同时在两个或更多磁道上写数据。例如，它们可以在两个相邻磁道上写相同的数据，也可在磁道之间的间隙中写相同的数据，从而建立一个非常宽的磁道。如果读该磁盘的一程序使磁盘驱动器读/写磁头在写有相同数据的两个磁道之间来回步进，那么数据流将不会中断。但在复制的磁盘上，读/写头外侧的消磁头会造成一个间隙，那里没有数据可读。

对抗措施 由于这种技术与PC的磁盘驱动器结构密切相关，所以唯一有效的对策是从程序中消除检查码。

### 8. 弱位技术 (见图 9)

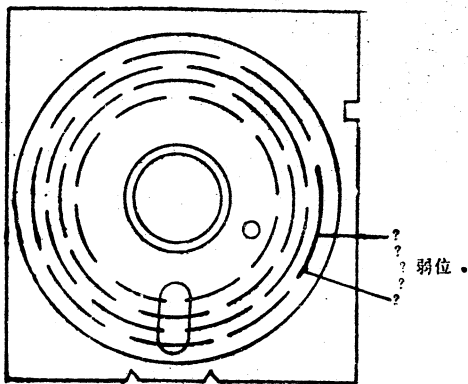


图 9 弱位技术

弱位技术是利用磁盘复制器能写“1”和“0”界限之间的数位的能力。每当传统PC读包含弱位在内的扇段时，它将产生出稍不相同的数据。用传统的技术复制磁盘产生的或许是“0”或许是“1”无法确定。

对抗措施 当写数据时，以极高的速度不停地通/断磁盘驱动器的读/写机构可产生合格的弱位的仿真。

为了防止用户非法复制，研究人员又已经转向加密技术，把保护的软件以加密的形式存

贮在磁盘上，并用专门程序进行译码，然后把它加载到存贮器中。还有的程序设计者把几种复制保护技术综合加以利用，使复制破坏变得更困难。但终究人们还是能想出相应的对抗措施，因此复制与反复制就是在不断地斗争中变得越来越复杂。

## 二、可靠性问题

复制保护研究人员最感棘手的任务之一是：要找到一些不仅能够做到只要一复制就能检测出来，而且还可以使原始盘可靠运行的复制保护技术。有些保护方案大约有5%的故障率，因此这种磁盘不可能使程序正常运行。

熟悉 Prolog 的工程人员指出，磁盘驱动器的调节要求非常精确，稍有差错检测代码就

不能正确无误地识别合法盘。因为 Prolok 依赖于磁盘的微小物理变化，因此为其它目的调节的驱动器也可能会使读/写磁头出错，定位和读出本不想读出的有效数据。当然 Prolok 的新版本可靠一些。

当磁头从一个磁道跨入另一个磁道时，根据读出的数据进行扇段对齐的技术可能也不够可靠。因为磁盘驱动器不同，其跨越速度和转动速度也稍有不同。比通常磁盘转动快或慢些的磁盘，从磁盘特定位置读出的数据将比预期的早一些或晚一些，这样就会弄坏原始磁盘，造成不良的复制。

### 三、用户与研究者

软件的用户和软件开发者之间是有很大矛盾的。许多软件开发者控诉他们的成果被偷窃，例如 Gates 控诉有人盗窃他的 Basic 解释程序，售价 500 美元，几乎接近运行它的微处理机成本。这样的非法复制使软件出版商难以获利。许多用户对部分软件商也不信任，抱怨不能在买软件之前进行测试，因此对有缺陷的软件和膨胀的价格无法提出异议。他们还抱怨买到的软件包常常不能正常地工作，用户还必须花额外的钱加以修改。所以用户反对复制保护，抵制制造厂家对它们的软件采取复制保护措施。

为了解决以上问题，1976 年美国订立了版权法，该法为软件非法翻印的起诉提供了更坚实的基础。1980 年在版权条款中增加了第 117 条，阐明了计算机软件用户的权力。概括起来就是规定软件用户在使用程序的过程中，可以进行某些方面的复制，例如：把程序复制到 RAM 中，或为了存档而复制。还允许修改程序使之适合特殊的应用。但用户不得为朋友或出售等其它目的进行额外的复制，即使用户自己修改的程序也不能流传。

文献出处：《系统工程与电子技术》 1987 年 8 期 52—56 页

# 磁盘文件的简单加密与解密

省局计算机室 马力群

计算机软件开发时间长，耗费大，一旦开发成功，复制却很容易。因而许多单位都想对开发出的软件采取一定的保护措施。因为各种软件的功能作用不同，所以对软件加密的要求也不同。本文向读者介绍几个对磁盘文件的加密与解密的方法。

## 一、IBM-PC DOS的磁盘

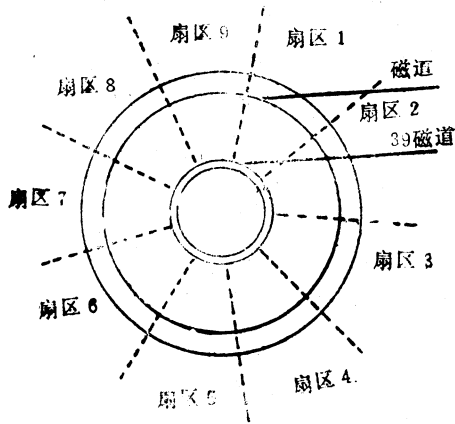


图1 PC-DOS标准软盘的磁道扇区分布图

### 文件结构

在PC-DOS磁盘系中，系统的FORMAT命令向用户提供了一套标准磁盘格式。以双面9扇区标准为例，磁盘格式化后，每面为40条磁道，由盘的外缘开始编号，逐道为0磁道、1磁道、……最内圈为39磁道。每磁道又划分9个扇区，每扇区字长为512个字节(见图1)。

逻辑扇区	0 面									1 面								
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
0	0*	1*	2*	3*	4*	5*	6*	6*	8*	9*	A*	B*	C*	D*	E*	F*	10*	11*
1	12*	13*	14*	15*	16*	17*	18*	19*	1A*	1B*	1C*	1D*	1E*	1F*	20*	21*	22*	23*
2																		
39																		

图2 IBM-PC软磁盘相对逻辑扇区表

这样每张盘被划分成720个扇区，共668,640个字节长度。为方便磁盘管理，PC-DOS将所有扇区进行统一逻辑编号（按十六进制）。0面0磁道的第一个扇点编为0\*扇区，第二扇区编为1\*扇区，……第九扇区编为8\*扇区。接着再从1面0磁道编号，第一个扇区为9\*扇区，依次为A\*、B\*、……。排完0磁道后，再从0面1磁道继续编号为12\*、13\*、……。直到编完所有的扇区为止（见图二）。

PC-DOS把所有扇区划分为四个部分：

#### 1. 引导程序区（IPL）

它占有0\*扇区，用来存放磁盘格式字、规格表和磁盘根自举程序。

#### 2. 文件分配表（FAT）

文件分配表占有1\*—4\*扇区，用来记录文件占有空间，在FAT中存放每个文件占有束（Cluster）的链接表。PC-DOS规定每两个扇区为一束，每束占1024字节空间。FAT共有一式两份文件分配表，各占有两个扇区。

#### 3. 文件目录表（DIR）

文件目录表从5\*扇区到B\*扇区，占有7个扇区，用来存放磁盘文件目录。

#### 4. 资料区

资料区从C\*扇区开始，共占有608个扇区，长度有362,496个字节。这个数字也就是用不加S参数的FORMAT命令格式化后得到的磁盘空间。资料区用来存放文件正文。

磁盘文件目录表占有7个扇区，长度为3584个字节，每个文件目录占用32个字节，所以目录表最多可存放112个文件目录。因而当软盘存满112个文件目录后，即使还有很大空间，也无法再存放文件。PC-DOS对目录的每个字节的含意都做了明确的规定。

各字节的含意规定如下：

0—7位 磁盘文件名

8—10位 扩展文件名

11位 文件档案属性，标准规定如下：

01 表明该文件为只读文件；

02 表明该文件为隐藏文件；

10 表明为一个次目录档案；

20 正常文件属性

12—21位 保留

22—25位 文件建立时间、日期。

26—27位 表明该文件在FAT中的起始束号。

28—31位 注明以字节计的文件长度。

IBM-XT磁盘文件结构与软盘类似，所不同的是，磁盘磁道总数为1224个，每磁道划为17个扇区，每扇区为512个字节。引导记录块放在0\*扇区，文件分配表存放在1\*—10\*扇区（十六进制），硬盘目录区从第11\*扇区开始，共占有32个扇区，它的每个文件目录也占有32个字节，因而10MB硬盘总共可管理512个文件目录。应提出注意的是，“浪潮0520A”的硬盘结构与IBM-XT完全一样，而“长城0520A”的硬盘文件目录长度为62个扇区，可存放1024个文件目录。

通过上面分析，可以得到一个结论，只要采用特殊的，非标准方式进行格式化，或者在



文件名、文件属性和文件分配表的确定等方面，不按照标准方式设置，就无法用正常的DIR命令和COPY命令进行工作，也就达到了加密的目的。

## 二、磁盘文件的简单加密

### 1. 在文件名和扩展名中加入无法显示的汉字区位码

我们可以将某些在屏幕上无法看到的区位码，插入到文件名中做为保密字。由于这些字符对应的机内码无法显示，因而当用DIR命令时，看到的只是不完整的文件名。可用来加密的区位码很多，如0001, 0002, 0795等等，我们可以通过实验来确定使用的区位码。

例：用拷贝命令进行加密处理

```
A>COPY TERG.PRG T(0011)R(0795).PRG
```

用DIR命令后，在屏幕上显示为：

```
A>TE R △ PRG
```

有些区位码显示的图形一样，但是区位码却不同，特别是某些正常汉字区位码以外的码，如0795, 1295, 1995等区位，显示的图形都一样，但区位码不同。

### 2. 在文件名中连续插入半个汉字

这种方式是在文件名中，每键入一个汉字后，立即用退格键“←”删去半个汉字，紧接着打入键盘上的任意一个字母，这样连做几次。用此法生成的文件名，在半个汉字的位置上显示出一个空白。因为汉字的机内码是由两个最高位为“↑”的ASCLL码组成，删去后半半个汉字后，还留有一个最高位“↑”的ASCLL码。这个特殊的码即显示不出汉字，也无法显示正常的字符。

例：用COPY（或REN）更改一个文件名A>COPY SHZ.PRG S(3456)H(4567)Z.PRG

过程是这样的：先键入S；再按区位码3456（仑）；然后用退格键删去后半半个汉字；再键入H；按区位码4567（碗）；接着再用退格键删去后半半个汉字；最后打入Z.PRG。现在再用DIR命令观察，显示出

```
A>DIR
```

标示 磁盘A没有标示！

目录 磁盘A: \

```
DEBUG COM 11904 10-20-83 12:00p
```

```
SHZ PRG 1325 1-01-80 12:27a
```

```
S H Z PRG 1325 1-01-80 12:27a
```

3个文件 346112字节空间

除了插入区位码加密，还可以插入某些控制字符，如ctrl+k, ctrl+L, ctrl+A, 或者插入其它字符，如：Alt+126, Alt+127, 等等。可以通过试验，选择保密字符。现在各种版本的CC-DOS很多，有时可能在这种版本下加密的文件，换一个版本就无法读出，也不能工作；这些问题应加以注意。

### 3. 修改文件属性，达到加密目的

在磁盘目录表中，文件目录的第11个字节表明该文件的属性。如果修改这个字节的内

容，使文件成为隐藏文件，就可以达到加密的效果。例如PC-DOS系统的IBMBIO.COM和IBMDOS.COM两个系统文件，在格式化时，被系统设置为“02”属性，当用DIR命令搜索时，该文件即不显示，也不计入文件数。修改属性的方法是这样的。

例：在软盘中有一个DBASE III编写的命令文件，名字为SHGL.PRG，现在把它的属性改为02属性，过程如下：

```

①A>DIR
DEBUG    COM    11904  10-20-83  12:00p
SHGL     PRG    1325   1-01-80  12:27a
          2 个文件    348160字节空间

②A>DEBUG
③-L 100 0 5 1
④-D 100 13F
3A4F: 0100 44 45 42 55 47 20 20 20-43 4F 4D 20 00 00 00 00 DEBUG
COM
3A4F: 0110 00 00 00 00 00 00 00 60-54 07 02 00 80 2E 00 00 .....T
.....
3A4F: 0120 53 48 47 4C 20 20 20 20-50 52 47 20 00 00 00 00 SHGL PRG
3A4F: 0130 00 00 00 00 00 00 7D 03-21 00 04 00 2D 05 00 00.....} 1
...

⑤-E 12B 02
⑥-D 100 13F
3A4F: 0110 44 45 42 55 47 20 20 20-43 4F 4D 20 00 00 00 00 DEBUG
COM
3A4F: 0110 00 00 00 00 00 00 00 60-54 07 02 00 80 2E 00 00..... T ..
3A4F: 0120 53 48 47 4C 20 20 20 20-50 52 47 02 00 00 00 00 SHGL
PRG
3A4F: 0130 00 00 00 00 00 00 7D 03-21 00 04 00 2D 05 00 00 ..... }
! ...
⑦-W 100 0 5 1
⑧-Q
⑨A>DIR
DBEUG    COM    11904  10-20-83  12:00p
          1 个文件    348160 字节空间

```

各步骤说明如下：

- ①显示原磁盘文件。
- ②调入DEBUG。
- ③从内存地址CS:100开始装入0号驱动器(A盘)的内容，由磁盘的第5\*扇区开始装入，共装入1个扇区的内容。
- ④显示装入的第5\*扇区(目录区)的内容，从地址CS:12至CS:13F是文件SH GL.

PRG的目录内容。地址12B的内容为20，它是PRG文件的正常属性。

⑤用“E”命令，把第11个字节的内容（地址CS：12B）改为02。

⑥与第4步对比观察。

⑦写回磁盘。

⑧退回到DOS状态。

⑨把第9步与第1步相比较，可以看到加密前后的情况，SHGL·PRG文件不再显示出来，也不计入文件数，但文件剩余空间仍然不变，说明文件还在盘中。

修改属性后的DBASEⅢ文件，不能用TYPE命令显示，也无法拷贝复制，用行编辑和字处理软件无法修改。但是可以在DBASE-Ⅲ环境下运行，也可以用DBASEⅢ的“MODI COMM”命令进行修改整理。

有关DEBUG软件各条命令的含意和使用方法，这里就不再详细介绍，请参阅有关手册。

我们还可以用上述方法把修改文件属性的工作扩大到批处理（后缀为BAT）文件和子目录（标识为<DIR>）文件当中。但是应注意，不同类别的文件，应该选用不同的文件属性，才能达到加密的目的。这里介绍这种可用于几种类型文件加密的属性

①2 表明为隐藏文件，将排斥任何标准的搜索命令，可用于DBASE的PRG文件和BASIC的BAS文件。

②7 表明为隐藏文件，将排斥任何标准的搜索命令，可用于后缀为BAT的批处理文件。

③17 表明为隐藏文件，将排斥任何标准的搜索命令，可用于子目录的隐藏加密。

在实际使用时，可以把上述几种方法综合使用，例如，先建一个子目录，然后按上面修改属性的方法，对于目录隐藏加密，再将一批经过加密的文件复制到该子目录下面。最后建立一个隐藏的批处理文件（利用修改属性得到）进行启动进行。由于进行了多重隐藏加密，因而增加了解密的困难，相对讲，就加强了文件和数据的安全保密。

在上述几种方法中，子目录的加密，使我们具有对大批数据文件提供保护的可能。目前，微型机价格较贵，一台微机常常很多人使用，既要完成正常的工作，存放许多重要档案和数据，还经常有初学者上机。有了加密的子目录后，可以方便地存放大批需要保护的数据和程序，还避免了逐个更改文件名的烦恼。如果将被保护的文件存放在多级被加密隐藏的子目录下，更增加解密的难度，提高了文件的安全性。

### 三、磁盘文件解密

当我们了解了上述各种加密方法后，解密工作也就好做了，只要用DEBUG软件把修改过的文件名和文件属性恢复原样就可以了。

例如：对前面用半个区位码加密的SHZ·PRG文件解密。可用DEBUG软件，先把磁盘目录区调入内存，然后用“D”命令查找准备解密的文件名。找到该文件名后，可用“E”命令把加密文件名更换一个新名，因为由机内码查找原来的区位码比较困难。最后，将更改后的目录区写回盘中。过程如下：

①A>DIR

```

DEBUG   COM   11904 10-20-83 12:00p
SHZ     PRG   1325  1-01-80 12:27a
SHZ     PRG   1325  1-01-80 12:27a
      3 个文件   346112字节空间
②A>DEBUG
③-L 100 0 5 1
④-D 100 15F
3A4F: 0100 44 45 42 55 47 20 20 20-43 4F 4D 20 00 00 00 00 DEBUG
COM
3A4F: 0110 00 00 00 00 00 00 00 60-54 07 02 00 80 2E 00 00..... T ...
3A4F: 0120 53 48 5A 20 20 20 20 20-50 52 47 20 00 00 00 00 SHZ PRG
3A4F: 0130 00 00 00 00 00 00 7D 03-21 00 04 00 2D 05 00 00 .....} .! ...
3A4F: 0140 53 C2 48 CD 5A 20 20 20-50 52 47 20 00 00 00 00 SBHMZ
PRG
3A4F: 0150 00 00 00 00 00 00 7D 03-21 00 10 00 2D 05 00 00 .....} .! ...
⑤-E 141
3A4F: 0141 C2.53 48.53 CD.53 5A.53
⑥-D 120 15F
3A4F: 0120 53 48 5A 20 20 20 20 20-50 52 47 20 00 00 00 00 SHZ PRG
3A4F: 0130 00 00 00 00 00 00 7D 03-21 00 04 00 2D 05 00 00 .....} .
! ...
3A4F: 0140 53 53 53 53 53 20 20 20-50 52 47 20 00 00 00 00 SSSSS PRG
3A4F: 0150 00 00 00 00 00 00 7D 03-21 00 10 00 2D 05 00 00.....} .
! ...
-W 100 0 5 1
⑧-Q
⑨A>DIR
DEBUG   COM   11904 10-20-83 12:00p
SHZ     PRG   1325  1-01-80 12:27a
SSSSS   PRG   1325  1-01-80 12:27a
      3 个文件   346112字节空间
A>

```

对文件属性的解密，可以参照上面的例子进行。特别要注意的是用“W”命令写回磁盘时，千万不要写错扇区号，否则会冲掉磁盘内容。在硬盘做这项工作时，更应慎重，硬盘的目录区是从第11\*（十六进制）扇区开始。

磁盘（包括硬盘）子目录管理区占有2个扇区（1024字节）的空间，子目录区的“文件目录区”满了之后，PC-DOS系统会自动申请新的目录区。子目录的“文件分配区”（FAT仍设在主目录区的FAT中。

子目录的文件目录扇区号可以这样得到：

先在主目录区找到该子目录的内容，它的第26、27位注明在分配表的起始束号（十六进制），然后代入下用公式：

子目录文件起始扇区号 = (起始束号 - 2) × 2 + C (均以16进制计算) 公式中的C是指软磁盘资料区的起始扇区号，如果是硬盘则应代入“+31”，这是指IBM-XT或“浪潮0520”，如果是“长城0520”则应+51。任何磁盘文件的起始扇区号均可用此公式求得。

子目录的“文件目录”由起始扇区开始存放，占有一个束（两个扇区）。对于目录里的文件操作，可参照前面有关主目录文件的操作去做。对于有多重子目录加密的文件，要经过反复细致的查找，这是一件非常麻烦的事，而且如果磁盘中有加密子目录时，用TREE命令是无法查出的，有没有更快更好的解密方法呢？下面介绍一种更简便的方法。

在PC-DOS系统中有一个检查磁盘文件命令CHKDSK，先用带有/V参数的CHKDSK命令查看磁盘，这条命令可以把磁盘中的一切文件，包括隐含文件和隐含子目录全部显示出来。然后和DIR命令显示的结果进行比较，就能发现是否有加密文件。

例如有一块软盘装有隐藏文件和隐藏子目录，现在对它进行解密。

先把PC-DOS系统的RECOVER.COM和CHKDSK.COM文件拷入C盘（或B盘）。

把待解密盘插入A驱动区，连接打印机，打入命令。

```
C>CHKDSK/V ✓
```

将盘上所有根目录和子目录的文件名，包括隐藏文件，全部列在打印机纸上：再打入命令

```
C>RECOVER A: ✓
```

该命令认为磁盘上所有的文件都出现问题，目录区已出现破坏，需要修复。它扫描文件分配表，按照文件分配表的链接情况，在文件目录区重新建立一个文件目录，新文件目录以

```
FILEnnnn.REC
```

形式出现。其中nnnn是连续的序号，它反映的文件次序，与前面用CHKDSK/V命令显示出的次序完全相同，而且文件的内容并没有改变。这条命令把每个子目录管理区也做为一个独立的文件加以编号列出来，同时把所有隐含文件改变为显型文件。整个工作过程如下：

```
C>DIR A:
```

```
DEBUG   COM      11904  10-20-83  12:00p
SHZ     PRG       1325   1-01-80  12:27a
          2 个文件      333824字节空间
```

```
C>CHKDSK/VA:
```

```
Directory A:
```

```
A: \DEBUG.COM
```

```
A: \SHZ.PRG
```

```
A: \S H Z.PRG
```

```
Directory A: \TE
```

```
A: \TE\HZQWM.PRG
```

```
Directory A: \TE\ROI
```

A: \TE\POI\DY.BAS

A: \TE\POI\ART.BAS

A: \TE\POI\IJN.POK

362496 bytes total disk space

2048 bytes in 1 hidden files

2048 bytes in 2 directories

24576 bytes in 6 user files

333824 bytes available on disk

524288 bytes total memory

297508 bytes free

C>RECOVRE A,

Press any key to begin recovery of the

file(s) on drive A,

9 file(s) recovered

C>DIR A,

FILE0001 REC 12288 1-01-08 12:19a

FILE0002 REC 2048 1-01-80 12:19a

FILE0003 REC 2048 1-01-80 12:19a

FILE0004 REC 1024 1-01-80 12:19a

FILE0005 REC 2048 1-01-80 12:19a

FILE0006 REC 1024 1-01-80 12:19a

FILE0007 REC 1024 1-01-8 12:19a

FILE0008 REC 2048 1-01-80 12:19a

FILE0009 REC 5120 1-01-80 12:19a

9 个文件 333824 字节空间

完成上述工作后，就可以参照用CHKDSK命令得到的目录清单逐个查看核对FIL En-  
nnn.REC文件，并用REN命令重新改回原来的名子。执行RECOVER命令后，后缀为  
REC的文件属性均改为"OO"。所以，有些文件解密后，还要把属性改为原有的文件属  
性。这样解密工作就完成了。

此方法同样适用于硬盘，但是由于硬盘无法备份，稍有不谨，就会破坏所有文件，要特  
别小心注意。

软件的加密与解密是一个很复杂的问题，它的方法繁多，无章可循，需要软件人员发挥  
自己的独创精神。上文介绍的各种方法虽然简单，但在干部档案管理、企业内部各种数据的  
保密方面，仍然是一种方便可行的方法

# IBM-PC微机磁盘防拷贝和信息加密技术

(武汉大学计算机科学系)

王化文 吴亮 张德向 张能胜

**摘要:** 本文介绍了IBM-PC微机的磁盘防拷贝和信息加密技术的一般原理和方法, 并就这些方法进行讨论: 如何利用IBM-PC微机本身的特点和机器系统提供的一些条件, 实现磁盘防拷贝和信息加密。

## 一、磁 盘 防 拷 贝

近几年所见在IBM-PC机上研制成功的防拷贝方法有多种, 从其特点来看, 大体可分为硬件防拷贝和软件防拷贝两类方法。

### 1. 硬件防拷贝

硬件防拷贝的方法有多种, 从原理上都差不多, 即要从物理上进行加工, 这里介绍三种常用的方法:

#### (1) 固化部分程序:

这种方法通常是增加硬件接口或更换某些存储器集成芯片, 将用户必须运行的某些程序段, 数据, 或密钥, 标志等都固化在这些集成芯片中, 这就使得用户必须了解这些信息的内容, 并取出这些信息插入到磁盘的信息中, 使之形成完整的信息, 才有可能完整地复制软盘信息, 从而使得一般用户难于实现拷贝。由于这种方法需要增加或改动硬件, 常用于一些专用的系统中, 因此在推广应用上受到了一定程度的限制。

#### (2) 激光穿孔软盘

利用激光技术也是近年来常用的方法。激光的方向性好, 能量高, 均匀, 利用这些特点在磁盘上打孔作为标记, 同时配上加密软件, 在取密钥之前判断激光孔标志, 并且对防拷贝程序和密钥进行保护。如防止利用调试诊断程序, 例如象DEBUG.COM这样的程序来观察加密程序或数据等, 这样既可以达到防止普通非激光孔软盘复制的目的, 又能阻止通过其它手段来窃取程序和数据, 增加了破密的难度。

#### (3) 掩膜技术

从防复制的原理上和激光穿孔软盘一样, 只是具体使用的技术手段不同。它也是一种物理方法, 是采用半导体加工工艺中的镀膜方法来制造标记, 由于膜标记在普通磁盘上不出现, 所以常用的拷贝程序无法识别, 就是一些高级的拷贝程序对此也是不能辨别, 所以也能达到防拷贝的目的。这种掩膜技术是近年来国外某些公司提出的新的防拷贝技术, 是一种很有发展前途的方法。

### 2. 软件防拷贝

软件防拷贝主要依赖于软件的方法, 在磁盘中隐藏一些一般的COPY软件无法复制的特

殊信息。IBM—PC机磁盘的通用格式规定为0~39磁道，每道8扇段或9扇段（包括单面或双面磁盘）。有些加密盘采用的方法是，在标准DOS系统磁盘格式的39道之外隐藏一些信息，如扩充磁道为43道。这种方法只能防止DOS系统本身提供的普通复制程序的命令拷贝加密程序，而对于高级的复制程序则照样可以进行拷贝。但是由于实际运用当中，许多PC的兼容机其磁盘驱动器不能正确地读取39道以外的信息，因此，这种利用增加磁道的方法很快也就没有再使用了。

除了利用增加磁道的方法以外，还可以变动磁盘的标准格式。这在IBM—PC机的DOS系统中是允许的。IBM—PC机磁盘的读写方式，是根据盘基表提供的参数，按确定的格式来读写的。盘基表（DISK BASE）由11个字节组成，它分别确定每一扇段的字节数，磁头稳定时间，马达启动时间，磁道道间间隙长度和位间隙长度等。盘基表入口地址是放在中断向量表中IE中断的入口处。IBM—PC原装机的入口地址通常是（DOS 2.1V）F000:EFC7，在系统引导后，由DOS将其装入到内存的0:0522处。

目前一些较有效的软件防拷贝方法，主要原理就是修改盘基表中的某些参数。例如，改变磁盘某些扇段的位间隙或某些磁道间隙，并将密钥等重要信息和某些关键程序存放在这些磁盘中的这些磁道中，使得COPY程序无法按正常格式读取这些信息，从而达到防拷贝的目的。

硬盘的防拷贝可以通过软盘的防拷贝来实现，硬盘上存放的加密文件只有用防拷贝的软盘引导启动以后才能执行。所以，某些重要的文件只要以密码形式存放在硬盘上，也就可以达到防拷贝的目的。因为，即使硬盘上的密码文件被复制，它也不能解译，也不能执行。

## 二、磁 盘 信 息 加 密

一般说来，磁盘防拷贝只是防止磁盘上的某一部分关键的信息或某些特殊的标志被复制。实际上不可能也没有必要做到整块磁盘的信息防拷贝。这就需要防止某些文件被窃取，防止使用某些工具软件直接读取磁盘上的信息。要做到这一点，就必须将文件和数据以密码形式存放在磁盘上，这样即使信息被窃取，也不能马上获得正确的信息。这个过程就是通常所称的加密。

信息的加密是和传统的密码学有密切联系的。但磁盘上代码的加密和通讯中的信息代码加密要求是不同的。从原理上讲是一样的，但磁盘上信息加密具有这样的特点：密码不宜增加长度，因为增加密码长度意味着所要加密的文件要增加占用磁盘空间；其二是密钥不能因时间的变化而变化，这就是说对同一磁盘来说运行加密文件其密钥不能随时间来变化，而要考虑其它因素来形成不同密钥；其三是加密算法要考虑机器的运行时间，不能因为加密解密使得大幅度降低对磁盘的读写速度。

加密的过程实际上包括一个代码转换过程。通常所用的代码转换方法有如下几种：

1. 位移法：将一个字节或一个字的信息左移或右移（循环位移）若干位，所移动的位数由某一算法来确定。如字符“A”的ASCII码是65，若循环左移3位则为43，即变成了符号“+”，若确定移位数字的算法有一定的复杂度，则破译便有一定的难度。

2. 逻辑运算：将原代码与密钥等参量进行逻辑运算形成密码。IBM—PC中，BASIC文件的P方式存储就是根据这一原理加密的。

3. 相加法：将原代码加上某一数以形成密码，这和位移法有点类似。



4. 错位法：将原码按一定规律变换成另一代码，如A→I, B→J, C→K, …，所以这种方法也称为代替法。

5. 交换法：IBM—PC机的字长为16位的通用寄存器，可有指令使得高位字节和低位字节交换，或字节的高4位与其低4位交换等，使得信息的某些位交换，达到代码加密的目的。

上述方法都是适于计算机上使用的代码加密方法，但是单一地使用上述某一种方法，或者整块磁盘都使用固定的方法进行数据转换，那将是很容易被解密的。因此，实际上实用的保密系统都是巧妙地综合使用多种方法，而且还可使用相同密钥或不同密钥来进行处理，即转换代码还要同密钥进行运算才能最后得到密码。对于一块磁盘来说，虽然密钥不能因时间而变化，但应根据其它的参数来变化，以增加破密的难度。

需要强调的是：加密和解密的代码转换算法必须是严格可逆的，否则在运算执行前解密就可能出错。不过不可逆的密码算法也是有的，但这样做使得加密解密算法更为复杂，就要以损失机器时间为代价，花大量的时间来做加密解密的运算，这在微机的加密算法中是不可取的。

### 三、防跟踪技术

防跟踪、防拷贝和信息加密是加密系统的缺一不可的三个组成部分。所谓防跟踪就是防止别人对加密信息进行解密和防止别人利用程序调试工具，演示加密系统的执行过程，达到取消防拷贝功能、窃取信息的目的。

例如，在IBM—PC机上的DOS系统中，提供了象DCBUG.COM这样的调试工具，它可以一步一步跟踪执行的程序，并可设置断点等等，是可用来破译密码的强有力的工具。还有一些软件工具，如DISKLOOK等具有检查磁盘上的各扇段的信息等多种功能，也是破密的辅助工具。针对这些情况，防止破密的要害问题就是防跟踪和不允许用户设置断点运行，偷看运行过程，窃取运行中的加密参数。通常的作法是某部分程序执行前改变机器系统的一些功能，使得跟踪、反汇编、设置断点运行、查看内存数据等操作无法正常运行。为达到此目的，常常采用如下一些措施：

1. 设置若干循环程序。这些循环程序除了运行破密者必须观察的一些指令外，还故意设置许多消耗破密者时间的无用指令，增加循环次数和多次调用子程序，把跟踪者引向歧途。提高了程序的复杂度，延长了破密时间。

2. 改变跟踪中断和断点中断的功能。这种方法使破密者无法使用机器系统的原来中断1和中断3的功能，而且又不能再修改已改变的中断向量，如果要强行修改，则机器运行时就中断锁死，使系统出现混乱。

3. 利用时钟中断。这种方法利是用时钟中断定时判断机器的运行情况，一旦发现跟踪或偷看断点信息，则把系统封死，使程序中断运行，破译者也无法查看内存的信息。

4. 使用变形程序。这种方法是通过解码得到的程序还是一种不可读的程序，而要进一步采取程序分段执行、分段加以变换，使之在程序执行过程中动态地完成源程序所要完成的功能，但又不会直接泄漏源程序的执行过程，增加了阅读程序和分析程序的难度，即使反汇编出来，也不能看到源程序的全部正常信息。

5. **重新设置屏幕显示特性。**这种方法是在加密程序不需要屏幕显示对话的执行期间，重新设置显示功能。如屏幕上卷，设置大号字符，改变行宽等。这样，源程序在这期间因为不需要显示而不受影响，但跟踪者运行到此则什么也看不到，从而达到了防止跟踪的目的。

如同信息加密一样，实用的保密系统都是同时采用多种方法来防跟踪，不同的系统所采用的方法也会不同，而且编程的技巧也各不相同，所以，一个好的加密系统，要有很强的综合利用这些方法的能力，这样的加密系统才更为可靠。

#### 四、结 束 语

综上所述，加密技术和防拷贝，防跟踪是紧密相关的，加密是为了防止复制和被跟踪窃取；而能防止拷贝就能达到很好的加密。我们虽然在IBM—PC机上对磁盘的防拷贝和信息加密作了一些研究和实际工作，但用户的要求是多种多样的，还有许多问题有待于进一步研究。如计算机网上用户间的信息加密和密钥分配，同一加密系统中的加密分级，以及数据库的加密数据和非加密数据的共享等等，这些都是计算机安全保密技术要进一步研究的课题。

文献出处：《小型微型计算机系统》1988年1期35—38页

# 微型机磁盘信息保密探讨

华东计算所 朱惠友

## 一、前言

本文对微型机上使用的软磁盘如何进行信息保密进行探讨，国内外有不少公司和个人都致力于这方面的工作，以致来达到保护自己产品的专利。但具体如何实现磁盘信息保密的文章却很少。本文的目的在于抛砖引玉，与计算机的同行共同探论如何更好地完善这个课题。

## 二、磁盘及磁盘读/写

使用过微型计算机的人都知道：微型机的内存对于大量的数据处理还是显得很不够，软磁盘、磁带作为外存，是内存的一种有力的扩充。特别是软磁盘，由于它的价格低、存储容量大、可靠性高及使用方便，更是得到了广泛应用。

目前常用的软磁盘有8"和5 $\frac{1}{4}$ "两种。记录密度分单/双面单/双密度，信息记录方式又分软分区方式和硬分区方式。一般软盘都为软分区盘，信息格式采用IBM 3740格式。这样便可将软磁盘分成若干个磁道(Track)，一个磁道又分成若干个扇区(Sector)，每个扇区又包含若干字节的信息。一个或多个扇区信息将组成一批有效的数据。

每张磁盘上都有称为待服道的磁道。该道上的信息用于辅助操作系统对盘上信息进行管理，其他磁道都用作存放数据，称为数据道。

待服道上一般有三个内容：磁盘自举程序，文件目录表及文件分配表(磁道/扇区分配表)。

磁盘自举程序被启动后，便将文件目录表及文件分配表读进内存，便于操作系统读/写磁盘文件。

文件目录表是一张由一串文件目录组成的表格。磁盘上有多少文件，就要在该表格中登记多少。一个文件目录至少应包括文件标识、大小、属性及磁盘上起始位置等信息。

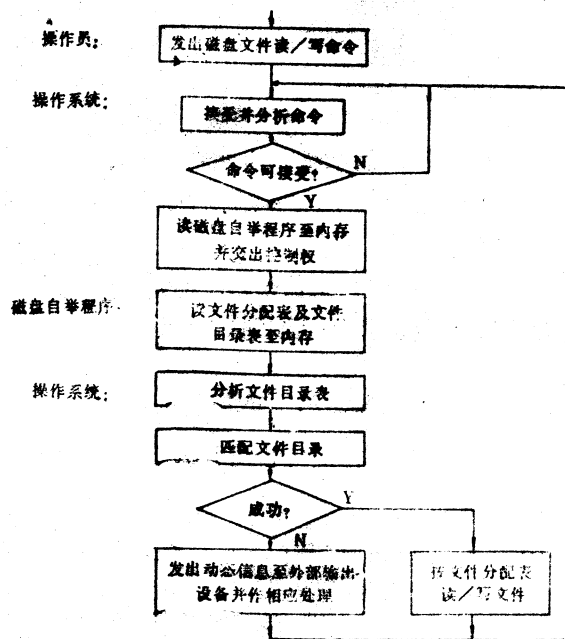
文件分配表(磁道/扇区分配表)辅助操作系统对磁盘空间的管理。写文件时，它为操作系统提供有效空间。读文件时，它指向该文件的下一个区域，指示操作系统按它的指针读出文件。

待服道上的信息都称为关键性信息，破坏了待服道，磁盘上的信息也就失去作用了。

了解了这些，对于理解磁盘上文件读/写是很有帮助的。

一般磁盘文件的读写过程是这样的：(见下图)

具体对磁盘的读/写是由CPU向软盘控制器发出命令，由软盘控制器控制驱动器来实现读/写软磁盘上一个或多个约定扇区格式的信息的。扇区信息格式的不同，将使读/写操作失败。因此熟练及巧妙地使用软件控制器的命令及参数，将有助于更好地使用磁盘。这对于磁盘信息分布及信息保密是十分有用的。



前面提及的这个流程对于磁盘信息加密提出了要求，磁盘加密的目标就是既要保证该流程能成功执行，又要达到能防止别人复制。下面讨论的一切都是围绕这个目标的。

### 三、磁盘信息加密的可能性探讨

由于磁盘文件的读/写都是由操作系统控制处理、由软盘驱动器控制软盘机来完成的，因此磁盘上文件及信息很容易被磁盘/文件拷贝程序拷贝。大多数程序产品编制者都希望对自己编写的产品程序提供保密措施，以保护自己的专利。他们希望在自己的产品程序中加上密码锁。只有知道密码的人才能使用该程序。但这毕竟是不可靠的。所谓的密码（称为Password）都要求用户输入一串字符作为钥匙，然后处理密码部分对这串字符进行处理，并与密码字符串比较是否匹配。若匹配的话，便打开密码锁，取出关键的程序运行，否则不运行。因此只要分析程序引导部分中的密码、或跳过该串密码处理部分，便可使密码锁失去作用。

又有人希望在自己产品盘上加进一部分信息，或对一部分信息作特殊处理。一旦拷贝程序取出这部分信息时，这部分信息能阻止拷贝程序拷贝。但他忽略了这么一点，拷贝程序拷贝信息时，是不对数据进行处理。它仅起一个数据传输作用。

那么是否存在着既能在操作系统控制下对该磁盘进行读/写，又达到能防止别人拷贝、保护自己产品的可能性呢？这种可能性是存在的。修改磁盘原有的信息存放格式、增加磁盘上可用磁道数及改变磁盘编码技术，都将使操作系统中所有对磁盘的操作失效。巧妙及充分利用这三种方法，便能使磁盘信息保密。

## 四、实现方式及举例

### 1. 修改扇区信息格式

通过修改软磁盘上扇区信息格式来使磁盘信息保密是较容易实现的。主要方法是对已经格式化后的软盘中的一道或几道再重新格式化。使两者的扇区格式不一致。然后将待服道上的关键信息甚至整个文件都放至这些道中。在文件分配表中注册这些磁道/扇区为“占用”。并在磁盘上加入一特殊的驱动程序。该程序的目的是在一启动该盘时，可将重新格式化的磁道/扇区中信息读至内存的指定区域，供操作系统使用。当用操作系统中的拷贝程序对该盘进行复制时，重新格式化的磁道就不能复制到目标盘上去。从而保护了关键信息的被复制。

但是磁盘中待服道信息分布是有一定要求的。操作系统是以绝对地址的形式(如扇区/道号)对待服道信息进行读/写操作的。如IBM-PC机所用的双面双密度盘片，若以每道9扇区的格式为系统盘，那么待服道信息分布为：

面	磁道号	扇区号	内容
0	0	1	磁盘自举程序
		2~3	第一张FAT表
		4~5	第二张FAT表
		6~9	} 112个文件表目
1	0	1~3	

PC-DOS根据这些约定对所需信息以绝对地址方式进行读写的。若在加密过程中对其处理不当，将事与愿违。从前面所提到的磁盘文件读写流程可看出，磁盘自举程序对该盘的使用起着举足轻重的关系。且对它并不存在写操作的问题，对它进行保密，就能达到既能防止别人复制，又能保证操作系统对盘读/写操作的正常执行的目的。

可能读者会提出这样的问题：如果也按照重新格式化的磁道/扇区的参数格式化自己的盘后，将源盘的这些磁道内容读至自己盘上，不也可以使用复制的软盘了吗？这个问题的解决比较复杂，涉及到许多方面的问题。只有对磁道信息分布、信息格式、磁盘参数、软盘控制器等比较熟悉的人才能较快地解决这个问题。对于一般的用户，则需花费较大的功夫来解决。有时甚至得不偿失。

笔者曾经用修改软盘格式的方式，对IBM-PC机上几个产品程序加了密码锁。此方法并非十分完美，仅用它作为例子来说明问题，并以此来抛砖引玉。

整个加密码锁过程如下：

- (1) 取一张空白盘片进行格式化(每扇区512字节)
- (2) 将要加密码锁的几个产品程序相继拷贝到该盘上。
- (3) 修改盘最后一道的格式(每区128字节、256字节或1024字节)
- (4) 从该盘的文件目录表中寻找这几个程序的文件目录并将它们按最后一道的信息格式写入最后一道。
- (5) 从软盘文件目录表中删除这几个文件目录。
- (6) 往软盘上加入两个文件A、B。A文件用作为驱动程序，用于将最后一道中的关键

信息(文件目录)读进内存中相应的存贮区,便于以后使用。B文件则将从相应内存区域中删去这几个文件目录。

(7)利用IBM-PC机上的批处理功能,形成批式处理文件。批式处理文件中的文件顺序如下:

A、C<sub>1</sub>、C<sub>2</sub>、……、C<sub>N</sub>、B

其中C<sub>1</sub>、C<sub>2</sub>、……、C<sub>N</sub>是要加保密锁的产品程序文件名。

这样不仅不破坏程序的正常运行,而且又能防止磁盘信息被复制。即使A、B程序也被复制,由于最后一道中的信息未被复制而使A、B程序失去作用。

笔者还曾分析了IBM-PC机上一张产品盘。该盘是通过修改扇区信息格式来防止别人拷贝的。由于操作系统只对该盘进行读操作,因此该盘上扇区信息格式分布是十分有趣的。磁盘的0面0道被格式化9个扇区,每区512字节。0扇区驻留磁盘自举程序。其它各道都格式化每道16个区,每扇区256个字节。这些道都为数据道,存放信息。该盘一被启动,磁盘自举程序按照特定的方式及约定的扇区信息格式将磁盘内容全部读至内存中指定的区域。然后启动该程序。这种方法的实现虽然比上述方法简单,保密性强,但有其局限性,且资源利用率低。一般一个产品不管其大小,都要占用一张软盘。

## 2. 增加磁盘可用道数

一般软磁盘最多只提供40道磁道为用户所用,但各厂家生产出的磁盘是有些区别的。一般有48道/英寸可用磁道。但最内的8道磁道是不太使用的。而且信息存放是有一定要求和标准的。有时这些磁道是有缺陷的。因此在对这些磁道进行测试、证明这些磁道可用的条件下,可将这些磁盘格式化,将关键信息移至这些道中。操作系统中的拷贝程序对于这些道的复制是无能为力的。使用这种方式保密,可不需编写自己的驱动程序。只需通过修改相应的参数、利用操作系统中最低层的基本输入/输出系统的功能来实现。

## 3. 改变编码技术

前面已说过,操作系统对磁盘信息的读写是由磁盘驱动程序驱动软盘控制器,由软盘控制器向磁盘机发出命令来实现的。在软盘控制器的命令中含有编码选择位,以选择不同的编码方式对磁盘进行编码,或以这种编码方式对磁盘进行读/写操作。如IBM-PC机中的软盘控制器 $\mu$ PD 765的命令中就含有MFM/FM选择位。一般都约定某一种编码手段为操作系统所用。但若对磁盘的某些磁道或全部磁道选用另一种编码方式,操作系统中的磁盘驱动程序就失去了其作用。依据这种思想,可将待服道用操作系统格式化后,对数据道按另一种编码手段进行格式化,并在待服道上加进一新的磁盘驱动程序。该程序以新的编码方法对磁盘数据道进行读/写操作。一启动该盘,操作系统先将该盘自举程序读进内存。该盘自举程序接着读进该盘上驻留的驱动程序。一开始对该盘数据道进行读/写操作时,就由该驱动程序执行。

这种保密方式虽然可靠性比前两种都高,但实现复杂,需考虑因素较多,但不失为一种好方法。

## 五、结 束 语

上面简单介绍了磁盘信息加密的三种方法。方法当然不止这三种，但这三种方法的结合使用也将是变化无穷，难于对付的。实际上，也大可不必为这种种保密方法所吓倒。无论怎样加密的软盘，都是要以保证实现前面所指出磁盘文件读/写流程的正常执行为前提的。而通过软件来实现磁盘信息保密，就总有一些异常迹象遗留在盘上。要破译的话，仅存在难易程度的问题。若用修改硬件的方法来实现磁盘信息保密，则只能使特定的机器使用该盘，有利也有弊。望各位专家和同行共同完善这个课题。

文献出处：《小型微型计算机系统》1986年1期39—42页

# 谈谈软磁盘加密

孙晓白

为了保护软件产品的潜在市场，人们开始研究软件的加密技术，同时解密技术也伴随而来。加密和解密经过‘魔高一尺，道高一丈’的矛盾斗争，促使加密技术从低级向高级迅速发展。目前软磁盘的加密技术发展比较快。

防止软磁盘复制的方法有多种。其中奥秘与盘片的构造或读出方式密切相关。

我们知道，软磁盘片上的信息均存放于同心圆磁道（track）之上。而软磁盘的每一磁道又分为若干磁区（sector）。每一磁区又分为三个部分：磁头区（header segment），数据区（data segment）；以及测试错误的总检查区（checksum）。一个IBM PC机所用软盘片，每一面各有40个磁道，每磁道有8或9个磁区。

保护技术之一是错磁区法。这种方法是在软磁盘片上写入一个或若干个错误磁区。错误磁区是指磁区的总检查区与其测试值不吻合。当COPY程序遇到这个错误磁区时，它可能会因为读数据错误而立即停止复制。但是强力拷贝软件可以识别错误磁区，因而可以复制这种保密盘片。

另一种办法是多余磁道法。即在磁盘格式化时，使盘片具有多于标准的磁道数。而一般的COPY程序只拷贝0~39磁道。这样40号、41号甚至42号磁道便无法复制，如果在这些磁道上放入一些关键程序，可使复制者因读不到这些程序而复制失败。当然如果能够找到额外磁道，而将其复制，则可解密。

还有一种方法是多余磁区法。即设法增加磁区的密度。在空隙较大，可以存放多数据的外围磁道，每个磁道可以挤入10个或11个磁区。因为一般的COPY程序只能读写0~8磁区，额外磁区会被遗漏。所以遇到这种盘片，拷贝时也会遭遇失败。当然，如果能够找到额外磁区，并将其拷贝下来，也将破密。

上述几种方法均与软盘的构造有关。

与读出方式有关的加密方法有：调换不同磁道号的信息，改变磁道转速，磁道间距不规则变化等方法。此外，将控制部分的软件固化，也是有效的保密方法。

在加密技术发展的同时，解密技术也在发展。美国COPY WRIT COPY II PC等就是这一类解密产品。这些高级拷贝软件的出现，使得很多软加密的方法失败。Oregon州Portland市的Central Point软件公司和加拿大安大略省多伦多的Quaid软件公司专门出售破密软件，他们出售的软盘片也不加保护。

而另一方面，软件生产公司也在想方设法防止使用者复制。为了防止软件被盗用，在技术上投资不少。甚至有些公司则专门是为了给软件产品加密而成立的。其中最著名的有两家。一家是美国加州New bury Park市的VAULT公司，开发了Prolok；另一家是美国加州Santa Clara市的Soft guard system公司，开发了Superlok。

由于许多软加密的方法失效，人们考虑采用更加高级的加密方法——硬加密。硬加密的有效方法之一是激光打孔。1984年美国VAULT公司声称采用唯一的指纹方法提供软件保



密，就是指激光打孔软盘上的激光孔构造各不相同，如同人的指纹一样。硬加密与软加密技术结合起来，为加密方法开辟了新的途径。

美国VAULT公司出产一系列保密工具，如Prolok software Protection System, Hard Disk Prolok, Filelok, Telelok, Proloker, Unilok, Large Volume Prolok, Proloader Automatic Diskloading Machine等等，下面具体做些介绍。

Prolok Software System在IBM PC-DOS、MS-DOS下运行。它为软件开发者和供应商提供有效的保护手段，以防软盘上应用程序的非法复制。每块5英寸的Prolok软盘片上含有唯一的指纹，没有别的激光打孔盘和它一样。对于它所保护的文件，该盘片上那唯一的指纹便是开锁的钥匙。因为它所保护的文件正是用这唯一的指纹锁住的，在该文件中已嵌入了这个指纹。嵌入指纹的加密过程是由Prolok盘都配有一个加密程序，它可以对于用户的执行程序加密。程序被加密后变成EXE型文件，并且都比原来的程序大10K左右。这是因为在程序之上加入了反跟踪、译码和读写密钥密码程序。加密的操作过程是这样的：将Prolok盘片放入A驱动器，把将要加密的程序放入B驱动器。假设程序名为QQ.EXE，打入命令：

```
A>PROLOK B:QQ.EXE
```

于是PROLOK程序开始执行。该程序运行以后，QQ.EXE便被拷贝到PROLOK盘中，同时已经用该盘的指纹加密。你可以把这个加密好的文件拷贝在硬盘中使用。当该执行文件拷贝在硬盘当中时，因为PROLOK盘片的指纹已经为被加密的文件上锁，如果你想执行硬盘中这个上了锁的文件，你必须用原来的Prolok盘片作钥匙，只有那个盘片上的指纹才和这个上了锁的磁盘文件的指纹完全相同。每当执行被保护的程序时，Prolok处理程序就把软盘上的指纹和原来嵌入文件的指纹进行比较。如果二者匹配则可运行。否则出现错误信息，返回操作系统。此外你可以为原来的Prolok盘做一份拷贝。要运行这个备份拷贝上的已被保护的程序，也要插入原指纹盘才能执行。使用Prolok最好做一备份拷贝。因为Prolok盘片是已经格式化的。如果由于不慎，把原指纹盘重新格式化，或有数据丢失时，可以用此备份恢复原指纹盘。

PROLOK盘片仅对EXE型及COM型文件加密，其中包括：汇编语言程序、编译BASIC、编译C、编译FORTRAN、编译PASCAL。但是对于解释BASIC、ASCII码BASIC及其它一些文本文件则不能加密。这些文件的保密可以使用FILELOK——另一种VAULT产品。FILELOK保护数据文件，它与PROLOK在功能上互相补充。在FILELOK之下建立的文件都用该盘的指纹保密。以后对于该文件的任何一次顺序存取都要有这同一块FILELOK盘出现在系统当中。如果除此之外，你还想使用暗语（Password），则可以使用暗语任选项。在这种情况下加密的文件，存取时除要求使用暗语之外，还需要原指纹盘作钥匙。使用FILELOK具体步骤如下：把应用程序盘放入驱动器A，FILELOK盘放入驱动器B。如果不用Password任选项，则在DOS程序提示符出现之后，输入FILELOK命令行：

```
A>B:FL A:文件名/
```

则该文件加密的过程开始执行。使用此类产品应当注意：FILELOK盘已经格式化，不要再次格式化造成原信息丢失。

HD Prolok具有Prolok盘片的功能，此外还提供另外的功能。这就是在被保护程序装入硬盘时，不必把原HD Prolok盘放在一个驱动器中。

这样就方便多了。这是通过一个叫做HDUTII程序提供的简便方法，该程序可以将指纹装入硬盘。当锁住的程序从硬盘中调出时，首先检查硬盘指纹，而不用软盘驱动器中的原指纹盘就可以运行这个程序。HD Prolok还允许用户设定一个数字，来限制不用原始盘独立运行拷贝的次数。可以执行的备份拷贝也通过使用HDUTIL程序产生。

上述三种加密盘片均可以在IBM PC-DOS或MS-DOS之下工作。需要DOS2.0或以上版本。

目前国内也在开展软磁盘加密工具的研制，并且已作为商品出售。

北京科海电脑系统公司最近推出一种‘虎符’Super Locker高级加密盘。科海声称它的加密系统可以产生240万种不同的盘片。用户买到的每一张盘都是不同的：被加密的程序若离开了加密的盘片便不能运行。该系统采用了X密码法，它多层次、多变、不可拷贝。该系统对于MS-DOS的EXE型和COM型文件加密，加密后文件类型不变，长度也不变；该系统采用多层次硬加密、软加密、反动态跟踪技术；加密操作简单方便。加密后的软件可以抗御美国COPYWRIT COPY II PC等高级拷贝软件。

中国电子技术研究院系统工程部去年年底推出WL威力激光加密系统。据称该系统由各界软件加密系统研制专家共同开发而成。WL能分析软盘上的激光孔，立即做出相应的密码文件。它可以对系统程序及可运行程序加密，也可以对用户数据文件加密；可以生产传送到硬盘的Prolok加密盘，在硬盘生成加密文件；也可以在硬盘上生成加密型数据文件。它有多种加密级的选择a) 指纹加密；b) 指纹和文件类型加密，其中文件类型有①设置系统文件位，可使被加密的系统文件作为引导的系统文件；②设置隐含文件位，可使被加密文件的文件名及所占字节数在终端用户盘目录中消失；③设置只读文件位可使被加密文件只能读不能写，可避免对程序的修改；c) 设置暗藏计数器加密①设置运行次数计数器②设置拷贝次数器；d) 设置口令加密。WL可在终端用户盘上建立系统区，作可引导的加密程序盘；还可以在终端用户盘上设置子目录，将文件直接在指定的子目录中加密。WL软件加密系统是国内开发的功能较全的新产品。

此外，上海计算所推出LK加密程序。它可以对任何·EXE文件和·COM文件进行加密。加密后的程序能防止COPY I、COPY II、COPY WRIT、Super COPY、COPYANY等的非法复制。同时可以防止DEBUG跟踪。它以软件方式加密，不破坏软盘。

激光打孔的加密盘国内也有生产。电子部六所三部和河北省科学院共同研制的适合于IBM-PC兼容机的双面双密度激光加密盘，目前已投入小批量生产。

# 怎样判定加密算法的和谐性

广西师范大学数学系 张师超

## 一、引言

一个加密算法的优劣，直接关系到信息的真正安全与否。在计算机飞速发展的今天，不能随机构造一个或者选一个加密程序就使用它。因为，首先，一些程序可能隐含某种错误，会影响译码的正确性；再一个是，企图破译的人有两种主要破译手段：第一，穷举法；第二，统计法。几乎所有的加密算法包括大素数算法（也称RSA算法，作者分析过，该算法不能避免统计破译方法）不能同时避免上述两种方法的攻击，必须作一些改正。

许多成功的加密程序是由几种方法的综合而写成的。例如，美国的PKS（公共钥匙系统）公司开发的PROTECTOR程序和DEDICAT/32程序就是RSA算法和随机函数等编制成的。

然而，不管我们寻找一种加密算法，还是综合几个加密算法，都应该注意加密算法的和谐性，也就是没有隐含性的错误。本文说明并介绍了几种加密技术的和谐性，希望起到抛砖引玉的作用。

## 二、和谐的加密算法

目前所使用的加密技术主要有三种：信息移位、信息替换和代数方法。

信息移位方法只要求有一个移动表或者有一个有规律的移动次序。信息替换方法要求有一个对照表这两种方法都较为简单，只要细心制造对照表或者记住次序的有规律性，则一般不会发生隐含性错误。但也要对每个字符进行一次检验，是否能被译过来。因为往往会出现个别的字符加密后不能被译过来。只要对每字符都做了检验，并且是正确的，则该加密算法没有隐含性错误。

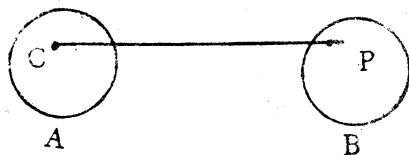
然而，目前用得最多的是代数方法。因为代数中的许多成果是非常完善，具有一些量的因素和代数技巧，将这两者应用到加密中，能在某种程度上避免穷举破译和统计破译。所以，这种应用具有强大的生命力。例如，RSA算法是利用了数论中的一些结果。

不管是哪一种代数方法，它必须将字符转化为数值，经过某种运算后，又将数值转化为字符来达到加密的效果。数值运算所牵涉到的运算一般为：加、减、乘、除、几次方、开几次方、与、和、取模、函数等。

如果设 $c$ 为原文码， $p$ 为加密后的密码，有如下的代数模型：

$$p=f(c)$$

这是代数方法的一般模型。 $f$ 为 $c$ 的某种函数。设 $A$ 和 $B$ 分别为所有 $c$ 和 $p$ 的字符集，则存在如下对应关系：



代数方法可能出现的隐错误就是，可能有两个 $C_1$ 和 $C_2$  ( $C_1 \neq C_2$ )，但是有 $P_1 = f(C_1)$ ， $P_2 = f(C_2)$ ， $P_1 = P_2$ ，这样，对 $C_1$ 和 $C_2$ 变为密文后，就不能转为原文。不失一般性，我们举一个例子看一看。

例：对字符作如下处理，将每个字符转化为数值后，乘以2，然后用94来取模，所得的数值再转化为字符，这个字符就是密码。但是，字符“%”的值为5和字符的值为52的“T”经上述运算：

$$2 \times 5 \text{ MOD } 94 = 10$$

$$2 \times 52 \text{ MOD } 94 = 10$$

于是，将结果10转化为字符“.”，则“%”和“T”都对应字符“.”，那么译码时将“.”译成“%”还是“T”，就出现了矛盾。

所以，不管用何种代数运算时，要保证C和P一一对应，才不会出现上述矛盾。

但是，这种说法也不是绝对的。例如，一个加密算法是这样的，置i的初值为1，对原文的第一个字符作如下运算：

$$P = (C + i \text{ MOD } 94) \text{ MOD } 94 \quad (1)$$

这里的C为原文码的值。将上面运算结果转化为字符P，则P为C的密码。将P转化为C时的方法是：

$$C = (P + 94 - i \text{ MOD } 94) \text{ MOD } 94 \quad (2)$$

对原文的下一个字符， $i = i + 1$ ，然后进行如上同样的处理，一直到所有的字符被加密。

这时，如果有两个C为(1)计算的P相等，但由于 $i \text{ MOD } 94$ 不同而代入(2)时，C也不同，不会出现矛盾。如果i不产生溢出，则该算法是和谐的，它也必须对每个字符进行一次检验。

### 三、总 结

这里，我们并没有增调一个加密算法的保密程度，只探讨了如何检验你的加密算法含有隐含错误，即算法是否和谐。因为一个算法本身不正确，更谈不上它是否具有高程度的保密性。一般说来，用穷举方法将每个字符检验一次。这种方法只保证算法的正确性，并不保证程序没有错误，程序的正误性要用软件工程来验证。但是，该程序可以说对这些字符局部正确的，是可以使用的。

#### 参 考 书 目 (略)

文献出处：《计算机时代》1988年2期26—27页

# 数据加密/解密器(DEU) Intel 8294A

Intel 8294A数据加密/解密器 (DEU—Data Encryption/Decryption Unit) 为了数据加密和解密而设计的一种微处理机外围芯片。它采用美国信息处理数据加密标准规定的算法来加密和解密64位的数据块,它根据64位的文本字进行操作,输入用户定义的56位信息代码来产生64位的密码字。这种加密算法是由芯片内部完成的。这56位信息代码和64位的信息数据按八位字节,通过系统数据总线与8294A来回传送。用户可以随时定义和变更这56位信息代码,这就可以得到不同的密码字。由于8294A有一个DMA接口和三个中断输出端,使数据传输时所需的软件开销减到最小,若采用两个DMA的传输方式,或用更多个8294A并行工作,可得到更高的系统转换速率,这种转换率是400字节/秒的任一倍数。8294A还有一个七位的与TTL兼容的输出端口,可由用户定义其功能。由于8294A执行NBS (National Bureau of Standards) 的加密算法,它可用于数据需要加密的各种场合,如财务管理、银行事务和数据处理等方面。

## 一、8294A的性能指标

Intel 8294A是一个有40条引出腿的大规模集成电路,逻辑框图及引脚排列如图1所示。它采用单一+5V电源、TTL逻辑电平。其主要性能指标如下:

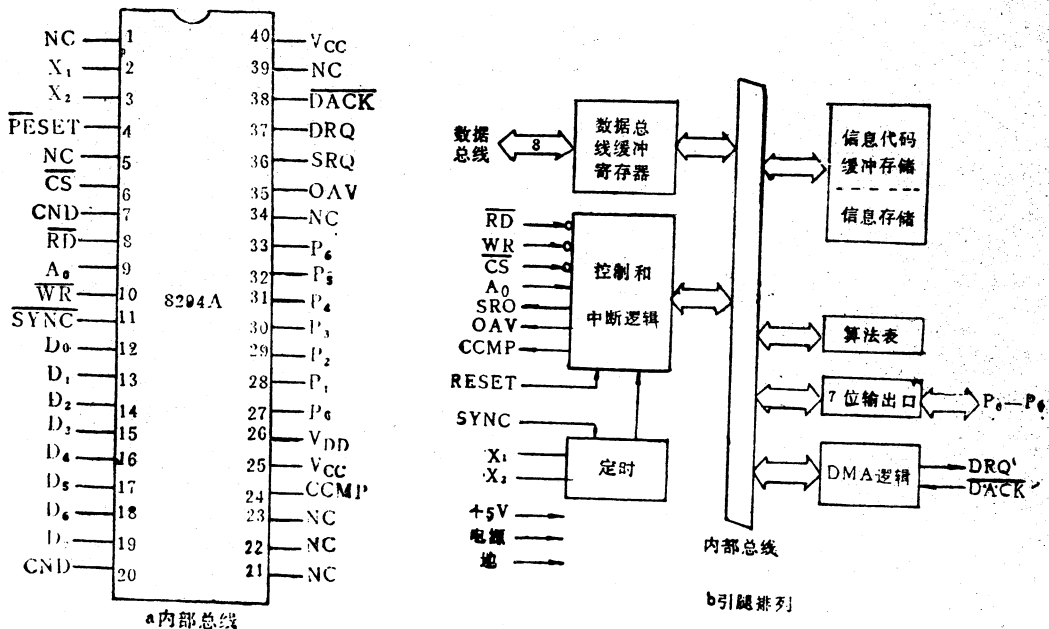


图 1

- 经美国国家标准局检定合格
- 400字节/秒的数据转换速率
- 72 .

- 采用56位输入信息代码的64位数据编码
- 一个DMA接口
- 用于数据交换的三个中断输出端
- 七个用户可定义的输出端口
- 单一+5V电源供电，允许拉偏±10%
- 与iPX—86,88, MCS—85<sup>TM</sup>, MCS—84<sup>TM</sup>, MCS—51<sup>TM</sup> 和MCS—48<sup>TM</sup> 处理器完全兼容
- 执行美国国家信息处理数据加密标准
- 可以采用加密、解密两种方式。

## 二、Intel—8294A的功能简介

### 1. 实现数据转换的操作步骤

(1) “设置方式” (set mode) 命令，允许建立所需要的中断输出。

(2) 输入新的信息代码 (Enter New Key) 命令，后面跟着8个字节的数据输入。该数据为DEU加密和解密而保留的，每个字节必须有“奇”的奇偶校验。

(3) 加密和解密数据命令，置DEU到所需的方式。接着通过写八个数据字节使数据转换，然后再读回八个已被转换的数据字节，就完成了加密或解密操作。在数据转换的间隙可以发出加密或解密命令来改变DEU的基本操作，也就是说，若发一个解密数据命令就可使DEU从加密方式改变为解密方式而无须改变信息代码和已允许的中断输出。

### 2. 数据加密/解密器的内部寄存器

DEU有四个内部寄存器，它们是数据输入缓冲寄存器、命令输入缓冲寄存器、数据输出缓冲寄存器和状态输出缓冲寄存器，其中两个是输入，两个作输出。主处理器可访问这四个寄存器，下表给出了如何访问这四个寄存器。

RD	WR	CS	AO	寄存器
1	0	0	0	数据输入缓冲器
0	1	0	0	数据输出缓冲器
1	0	0	1	命令输入缓冲器
0	1	0	1	状态输出缓冲器

这些寄存器的功能如下：

(1) 数据输入缓冲寄存器—写到这个寄存器的数据是下列三种方式之一，究竟采用哪种方式，取决于命令的顺序。

①一个信息标号的操作码

②加密和解密的数据

③一个DMA的数据块的块数

(2) 数据输出缓冲寄存器——从这个寄存器读出的数据是加密或解密数据的输出。

(3) 命令输入缓冲寄存器——DEU的命令写入该寄存器。

(4) 状态输出缓冲寄存器——可随时从这个寄存器得到DEU的状态，供处理器查询驱动设备和进行数据传送操作。各状态位功能是：

状态位：	7	6	5	4	3	2	1	0
功能：	×	×	×	KPE	CF	DEC	IBF	OBF

OBF—输出缓冲器满。OBF=1表示在数据输出缓冲器中可得到加密/解密功能。当数据读出后，它被复位。

IBF—输入缓冲器满。当数据输入缓冲器或命令输入缓冲器写入时，置IBF=1，当输入的字节被DEU接收后，DEU复位这个标志。（当IBF=1时，不能写入数据或命令）

DEC—加密/解密标志。表示DEU是工作在加密方式还是解密方式。当DEC=1表示工作在加密方式；DEC=0表示工作在解密方式。

8294A在接收了一个加密或解密数据后，要求11个周期去建立DEC位。

CF—完成标志（completion flag）。这个符号用来表明在数据传送约定中的三种情况或它们中某一种。这三种情况是：

①在处理器中，它可以用来代替一个8字节传送的结束标志。

②用来指示KPE标志生效。

③可以用来代替CCMP中断。指出一个DMA操作完成。

KPE—信息代码奇偶错（key parity error）。输入一个新的信息代码后，DEU用这个标志连同CF标志一起来指出奇偶的正确性。

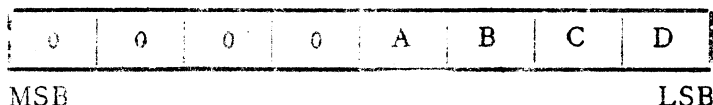
### 3. 命令汇总

(1) “输入新信息代码”命令。操作码是40H。这个命令后跟着输入8个数据字节，它们被保留在信息代码缓冲存储器里，作为加密或解密的数据，这些数据字节必须有“奇”的奇偶性（奇偶性由最低有效位表示）。

(2) 加密数据命令。操作码是30H。这个命令置8294A到加密方式。

(3) 解密数据命令。操作码是20H。这个命令置8294A到解密方式。

(4) 方式设置命令。操作码是：



这里

A是OAV（Output Available输出有效）中断允许。

B是SRQ（Service Request服务请求）中断允许。

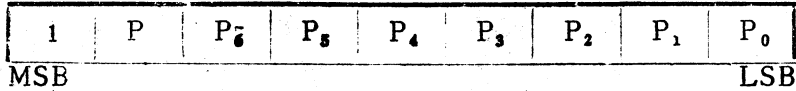
C是DMA（Direct Memory Access直接存储器存取）允许传送。

D是CCMP（Conversion Complete转换结束）中断允许。

这个命令可决定被允许中断是哪一个输出端，A、B、D为1时，分别允许OAV SRQ或CCMP中断。C=1时，将允许DMA传送，当C置位时，也将允许OAV和SRQ中断（A、B位=1）。执行这条命令后，若C位（DMA位）被置位，8294A将要求用一个数据

字节来规定用DMA传送的一个8字节数据块的数目。

(5) 写输出口命令。操作码是：



这个命令可以使命令字节的7个最低有效位，作为输出数据锁存到8294A的输出端口上。初始的输出数据是1111111。这个端口的使用与加密/解密功能有关。

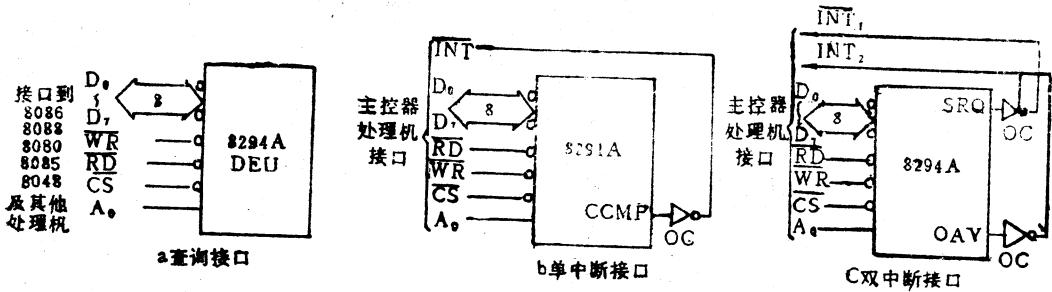


图2 DEU与处理器接口关系

### 三、DEU与处理器接口约定

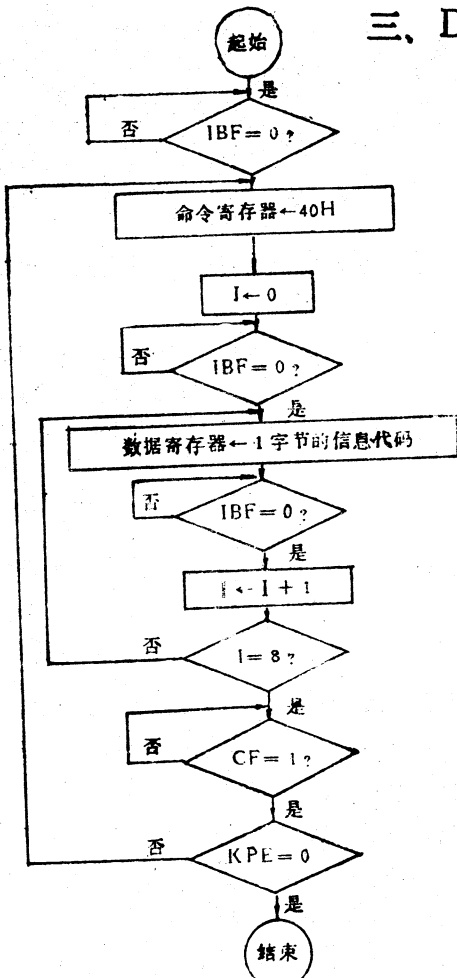


图3 输入一个新信息代码的流程图

#### 1. 输入一个新的信息代码 (Entering a New Key)

输入一个新信息代码的软件流程图如图3所示。输入新信息代码命令发出后，表示新信息代码的8个数据字节被写到数据输入缓冲区里，8个字节被DEU接收后，CF变“真”（CF=1），当KPE有效时，CF变“假”（CF=0），此时CPU可检测KPE标志，如果KPE=1，则表明检测到奇偶错，DEU没有接收这个信息代码。每个字节的最低有效位代奇偶位，每个字节都应是“奇”的奇偶性。

由于CF只能作为8个字节输入完成后的结束标志，所以要设置一个软件计数器来记录输入数据次数。而CF与KPE一起来表示这个新的输入信息代码是否有效。

#### 2. 加密和解密数据

图4给出了两种方式的程序流程图，一种是CF方式，另一种是软件计数器方式，这两种方式都可以用来结束读、写数据。



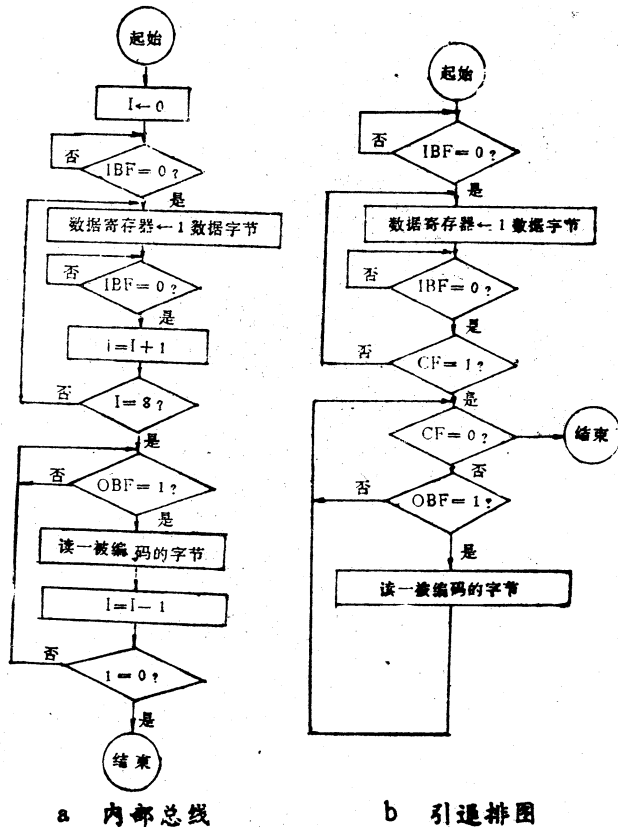


图4 数据转换流程图

式。CCMP中断也可以用来初始化一个服务程序，实现下一个序列的8个数据读出和8个数据写入。

SQR = 1 意味着 IBF = 0；OAV = 1 意味着 OBF = 1。它允许中断程序不用先检测状态位，就可以完成数据传送。然而，OAV 服务程序必须要检测状态位标志和数据转换结束标志。

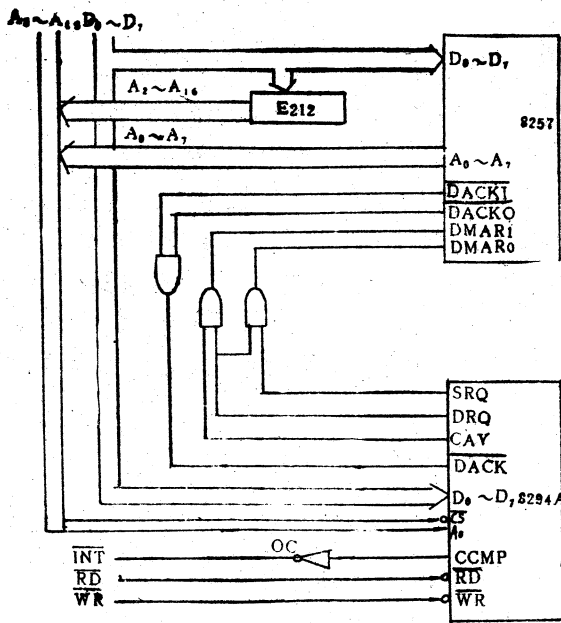
### 3. DMA 传送

图5示出DMA方式的硬件接口，在使用DMA方式时，要求有3个外部“与”门和两个DMA通道（一个输入，一个输出）。因为DEU只有一个DMA请求引脚，所以用二个“与”门将SRQ、OAV结合起来作为DMA通道，分别产生自己的DMA请求，用第三个“与”门将二个低电平有效的DACK端结合起来作为8294A的DACK信号。在DMA传送之前，CPU必须首先初始化二个DMA通道（如图6所示）。然后必须发给DEU一个设置方式命令，使其允许OAV、SRQ中断和DMA传送，CCMP中断可以被允许或者被禁止，视需要而定。设置方式命令之后，必须给出一个数据字节，用它来规定被转换的数据（ $N < 256$ ）的8个数据块的数目。然后DEU产生要求二个DMA通道的请求数目，而不需CPU的进一步干预。当要求转换的数据块数转换完成后，DEU将置位CF，并响应CCMP中断（如果CCMP中断是被允许的），随着下一次写入DEU命令或数据，CCMP变“假”。转换完成后，撤消DMA

在CF标志方式里，CPU将8个数据字节写到数据输入缓冲器后，CF变“真”（CF = 1），表明DEU已接收了8个字节的数据块，CPU是通过检测IBF = 0和CF = 1来实现数据写入的。通过检测CF = 0和OBF = 1来实现数据读出，因为读出8个字节数据后，CF标志变“假”。

在软件计数方式里，CPU通过检测IBF = 0或OBF = 1及软件计数器的次数，来实现读、写。

当加密/解密完成后，CCMP、OAV中断被响应，并且OBF被置位（OBF = 1），CPU读回每一个被转换的数据字节后，再把OVA和OBF复位，第一次被读出后CCMP中断被复位，并保持。CPU读回8个数据字节后的CF变“假”（CF = 0），于是CPU可以测试CF = 0来结束读出方



5 DMA接口

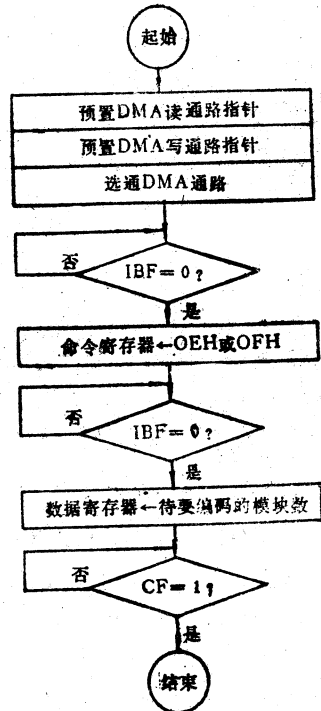


图6 DMA流程图

方式，并且DEU又恢复到加密/解密方式。在发生另一个方式命令之前，被允许的中断输出端保持允许状态不变。

#### 4. 单字节命令

图7给出了单字节命令流程。注意：除了在DMA转换期间外，任何命令都是有效的，这样一来，DEU被设置到一个已知的状态，然而如果一个命令要依顺序发出，则要求另外有一附加的约定，如图8所示，CPU必须等待直到接收这个命令（IBF=0），然后必须发出一个“数据读出”命令，以便清除任何前面的命令序列留在数据缓冲区里的任何内容。

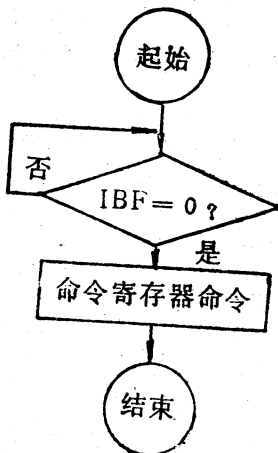


图7 单字节命令

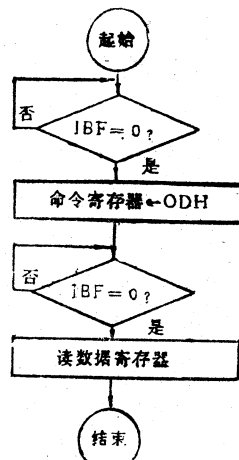


图8 平复约定流程图

郭绪杰摘自《MiCrosystem Components Handbook》Vol. I 1985.

# 软件的加密

曹小多

软件加密是每个软件工作者普遍关心的问题,因为它直接关系到软件编制者的利益.软件加密的目的是使经过加密处理后的软件在转交用户使用时,用户只能获得软件的使用权,而不能看见程序的细节,或对程序进行仿造和复制.为了使读者能了介并掌握软件的加密方法,笔者参考有关资料,结合自己对软件的加密经验和对一些加密软件进行剖析后所获得的结果,在此作一介绍.

大家知道,要读出或复制一份软件,必须经过以下几步才能办到:

- (1) 从磁盘文件目录中获得文件名称.
- (2) 从磁盘中读出该程序.
- (3) 当CPU转交控制权后,即可对程序进行显示,打印及复制工作.

软件加密也就是阻碍以上工作的正常进行.用迫使用户无法获知程序名、无法读出程序、无法获得控制权的办法来达到防止泄密的目的.但是事物往往是相反相成的.有加密的方法也就有解密的办法.加密的方法采用得越多越巧,解密的难度也就越大,软件的保密性也就越强.因此,本文介绍的加密程序采用了多层次加密的方式,可使被加密后的软件具有很强的保密性.以下就以加密程序为例,向大家介绍软件的几种加密方式.

微机中导入DOS 3.3系统以后,加密程序的1~14行的作用完成对DOS 3.3系统的加密处理.系统加密的目的是使转交用户使用的软件无法在标准的DOS 3.3系统控制下进行读写操作.读者可参照程序中采用的方式灵活地设计出自己的DOS命令名称、文件名存放磁道和识别密码,以使经您加密后的软件旁人无法解密.

程序第一行的作用是将DOS占用区内的44033单元的存放数由原17改为3.因为此单元的存放数字可决定文件名目录在磁盘中存放磁道,所以现在将此数改为3后,目录存放磁道即由17改为33.又因磁道总数为35条,其中0至2磁道是用来存DOS系统的,所以文件名存放磁道只可以在3至34磁道中选用.文件名存放磁道改变后,在标准的DOS系统控制下是无法获得文件名的.

加密程序2至6行的作用完成对DOS读写命令的更改.这样可使标准的DOS命令无法读写程序,以下列出这几条命令字符的ASCII码在DOS区中地址:

LOAD——43144~43147

BLOAD——43257~43261

SAVE——43148~43151

BSAVE——43252~43256

CATALOG——43218~43224

只需更改以上内存中各字符命令的ASCII码,即可完成各命令代表字符的更改,使存取命令非标准化.读者可以先设计好各命令新代表字符,将字符转换成相应的ASCII码后写入各命令原占用区中即可.本程序第2行是将LOAD命令更改为LOXD.第三行是将BLOAD

命令更改为BLOXD。第4行是将SAVE命令更改为BSAVE。第5行是将BSAVE命令更改为BXVE。为了使系统能识别各命令的结尾字符，如第6行所示，结尾字符的ASCII码必须加上128后写入。该行是将CATALOG命令更改为CATALOO，字符O是本命令的最后一个代表字符，用此被加上128。更改时若不希望影响其它DOS命令的正确执行，新命令字符的个数应小于或等于原命令字符的个数。

程序第7行是在DOS区中的46989单元写入一个可供所编程序识别的密码。可供加入密码的内存范围为46989~46994共六个单元。

程序24行可对加入的密码进行识别，如果发现密码24行可对加入的密码进行识别，如果发现密码不对就将程序区和DOS区全部清零。

大家知道，运行中的程序如果被中断，用户就有可能对程序进行显示或复制，同时DOS系统中的加密措施亦被暴露。因此防止程序运行被中断是加密工作重要的一环。以下列出APPLE机可进行的几种中断：

- (1) CTRL—C 中断
- (2) RESET 中断
- (3) NMI 中断
- (4) IRQ 中断

其中(1)、(2)项为可用键盘操作方式引起的中断。(2)、(4)项为外设中断，可采用将CPU相应的中断输入端电位拉低的方式实现。为了防止以上几种可人为引起的程序运行中断，加密程序采用改变中断矢量，将矢量指向RUN命令进入点的方法达到防止中断的目的。

加密程序8至9行是在DOS区中建立一个判断CTRL—C中断命令的机器语言输入拦截程序。程序10行的作用是改变输入矢量，将其指向机器语言拦截程序入口。为了保持DOS系统原有功能，本程序占用了“LANGUAGE NOT AVAILABLE(无此套语言)”出错提示字符区。因此如果发生以上错误，屏幕上将不会显示出错提示字符。但这样并不会影响工作。程序第11行的作用是设定RUN命令系统程序高位和低位入口地址，读者需加密的程序如果是机器语言程序，可将程序入口的低位和高位换算成十进制后赋予L和H。程序12行的作用是改变RESET中断矢量指向地址。程序13行是改变NMI中断矢量地址。程序14行是改变IRQ中断矢量。

以上程序完成了DOS系统加密。程序15行的作用是当完成了以上加密任务后，将以上程序清除掉。程序20行以后的作用是完成对所编程序的加密。

程序20行和22行是防止运行中的FPBASIC程序在出错时(例如用户没有遵照程序规定的使用方法操作)引起的中断。必须指出的是，读者在编制程序时应充分考虑到各种出错的可能。并采用可靠排除方式，以防止程序交用户使用中出错后，虽然无法中断运行，但也无法继续工作下去。

程序23行是在内存214单元写入数128。这样可使所有的键盘操作和命令在按了回车键后，都被自动转换为RUN命令来执行。这样可防止程序被显示和打印。

程序30行以后为读者程序区。读者如果进行加密的是FPBASIC程序，可按以下几步完成软件的加密工作：

- (1) 将本文介绍的加密程序作为程序的前段与所编程序连接成一个程序。

(2) 键入RUN命令运行程序, 这时即可完成DOS系统的加密, 并且在加密完成后程序会自动删去以上DOS加密段。

(3) 运行停止后, 键入INIT HELLO (HELLO为文件名, 在文件名中可加入不显示的字符以强化加密效果)。命令运行后即可将被加密化了的DOS系统和加密化了的程序存入磁盘。因为程序是作为HELLO文件存放磁盘的, 所以用户不需知道文件名, 驱动磁盘机后程序即可运行。

读者如果需要进行加密的程序为机器语言程序, 可将加密程序第30行改调入并进行机器语言程序的命令后, 按以上几步操作即可。

在被加密的DOS系统控制下, 如果需要从标准DOS磁盘读取程序, 并写入加密磁盘, 可按以下几步工作。

- (1) 键入POKE44033, 17; 恢复文件名存放磁道。
- (2) 用新改的DOS命令完成标准磁盘中的程序读出。
- (3) 键入POKE44033, X; 将文件名放磁道改为现值。
- (4) 用新改的DOS命令将程序写入加密磁盘。

```
1 POKE 44033, 3
2 POKE 43146, ASC("X")
3 POKE 43260, ASC("X")
4 POKE 43149, ASC("X")
5 POKE 43254, ASC("X")
6 POKE 43224, ASC("O") + 128
7 POKE 46989, 33
8 FOR I=43380 TO I+21: READ
  A: POKE I, A: NEXT
9 DATA 133, 255, 173, 0, 192, 201, 131, 240, 5, 165, 255, 76, 129, 158, 169,
  0, 141, 16, 192, 76, 223, 164
10 POKE 40194, 116: POKE 40195, 169: REM CTRL-C
11 L=223: H=164
12 POKE 40530, L: POKE 40531, H: REM RESET
13 POKE 40573, L: POKE 40574, H: REM NMI
14 POKE 40575, L: POKE 40576, H: REM IRQ
15 DEL 1, 15
20 ONERR GOTO 22
21 GOTO 23
22 RESUME
23 POKE 214, 128
24 IF PEEK(46989) <> 33 THEN
  A$="801<800.COOM N 36: FO
  FD IB FD N EOO3G": FOR I= 1
  TO LEN(A$): POKE 511+I
```

ASC<MID\$(A\$, I,1)>+128:

NEXT, POKE 72, O: CALL-144

30 REM BELOW ARE BELONG TO USER PROGRAM

命 令	失 效	恢 复
CATALOQ	POKE42350.96	POKE42350.169
LOAD	POKE42003.96	POKE42003.32
BLOAD	POKE41821.96	POKE41821.32
SAVE	POKE41879.96	POKE41879.173
RSAVE	41777.96	POKE41777.169

附表中列出的是可使DOS读写命令真正失效和恢复的几条命令，读者可根据编程的实际需要将其加入DOS加密段程序或所编主程序中。例如在给DOS程序系统加密时使读写命令失效，在主程序运行到需用此命令时再使其恢复正常后运行。待运行完毕后再一次使其失效。这样，用户即使获了加密后新读写命令后，仍无法读写程序。

经以上层层加密后软件，即可获得很强的保密性。读者不妨一试。

文献出处：《苹果园》1987年6期26—28页

# 计算机信息安全保密及安全保密通讯处理机

航天工业部七〇六所 刘黎临

**摘要** 随着计算机网络通讯的迅猛发展,系统间数据传输的安全性问题日益突出起来。本文阐述了计算机信息系统安全保密的层次结构和算法,并介绍了美国Honeywell公司推出的安全保密通讯处理机。

## 一、引言

近年来,计算机技术迅猛发展和普及,使得计算机及操作系统、网络和数据库的安全保密问题日益突出起来。特别是自从国际标准化组织(ISO)于1977年提出“开放系统互连参考模式(OSI)”以来,已有越来越多的国际组织和制造厂家接受了这一模型,各种专用计算机网络也都在完成或加紧向OSI标准化网络过渡(标准化程度较高的如美国Honeywell公司的DSA网,DEC公司的DEC-NET;美国IBM和Sperry等公司也都在加紧将其网络标准化),从而推动了计算机网络的迅速发展,推动了大型信息系统的建立。随之,越来越多的应用部门开始通过网络来提供信息服务,系统间的通讯能力和范围在不断扩大。银行、国防、公安和统计部门对计算机的依赖性越来越大,这样,利用网络传送的数据的安全性成为举足轻重的问题。许多重要信息、资源和财富都集中在计算机系统中,通过网络进行交换,而一旦遭受破坏,将会造成严重后果,甚至使某些部门、地区陷入瘫痪,这就是所谓“信息社会的脆弱性”。

计算机信息安全保密问题引起了全世界的关注。我国是近一、二年才开始这方面的研究,1986年7月在青岛召开了首届学术会议。全世界越来越多的学者开始从事数据传输和存储保护技术的研究。在信息安全保密中得到广泛应用的密码学也已经离开了军事和国家保密机关的独占领地,成为一个急剧发展和广泛交流的学术领域。

1985年底,在加拿大多伦多召开了“第二次计算机信息安全国际会议”,共有300多位各国代表参加,其中有许多著名人物,如美国斯坦福研究所高级顾问帕克,瑞典数据监控局局长佛里斯,美国国家计算机犯罪中心主任勃鲁贝克。会议重点讨论了如下两个问题:

1. 新技术革命正在促进世界进入信息社会,以美国为例,到1990年美国拥有微型计算机的数量将达5000万台;办公室电脑化,纸面记录被电脑贮存所代替。计算机网络和微型机改变了世界,甚至将会影响战争的形式。专家们认为“计算机是不用原子弹的战争。”

2. 信息系统安全的重要性,由于逐步进入信息社会,计算机及其网络系统已经成为“机密最集中”及“财富最集中”的领域,是各国情报机关和犯罪分子最注意的目标。

## 二、计算机信息系统安全保密的层次结构

对计算机信息系统的安全保密构成威胁有如下几个方面，

- 自然灾害；
- 系统故障；
- 使用维护人员的粗心；
- 预谋的犯罪。

我们用层次概念来建立计算机信息系统的保护层并描述其完成的功能和实现的方法。为了信息系统的安全及保密，必须从系统本身出发由里向外建立七个保护层次，如图1所示。

我们将计算机信息系统的安全保密机制划分为三大部类七个层次。第一部类为物理安全机构，仅由第一层物理安全保护层构成；第二部类为系统内部完整性和安全保密性机构，由第二层硬件保密保护层、第三层操作系统安全保密保护层、第四层网络保密保护层和第五层数据库安全保密保护层构成；第三部类为法律及制度安全性机构，由第六层系统管理制度安全保护层和第七层法律及社会环境保护层构成。如图1所示。

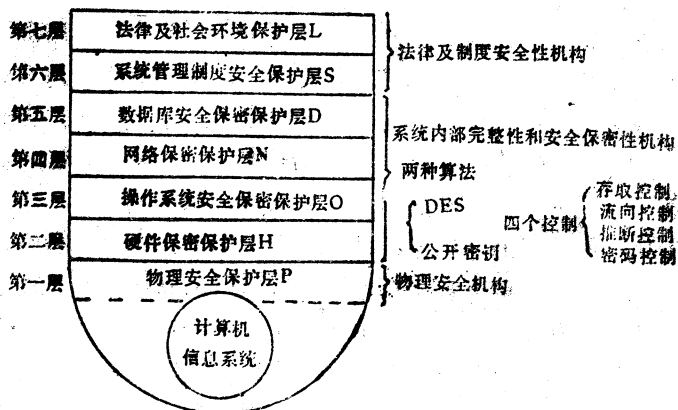


图1 计算机信息系统保护层

要建立整个系统的安全保密，使系统处于不停顿、不泄密的工作状态，应抓住七个保护层及两种保密算法——DES（联邦数据加密标准）和公开密钥在实际环境中的实现。下面侧重介绍层次功能及实现方法。

1. **物理安全保护层** 它应完成如下功能：

- 机房警戒装置；
- 机房锁定装置；
- 机房抗震、防火、防水；
- 在另一物理位置上备有系统中心机。

2. **硬件保密保护层** 防辐射和电磁波干扰；防止通过接收电磁波而剖析系统资源；完成部分固化的软件保密算法及完成最终的保护功能。

3. **操作系统安全保密保护层** 对操作系统（OS）本身进行保护。OS的安全性模式有



三个基本成份：对象 (objects) 集合；主语 (subjects)；规则 (rules) 集合，规定“一个主语对一个对象的存取操作类型”。可以用OS的存取控制矩阵A来表示，其列为对象—— $O_1, O_2, \dots, O_n$ ，行为主语—— $S_1, S_2, \dots, S_m$ 。矩阵项目  $A[S_i, O_j]$  包含一张存取操作类型表  $t_1, t_2, \dots, t_1, t_k$  表示“主语” $S_j$ 拥有“对象” $O_j$ 的存取特权。图2是OS的“存取矩阵”的一部分。

规则 R 主语 S		对象 O					
		主 语		文 件		设备 (磁盘)	
		$S_1$	$S_2$	$F_1$	$F_2$	$D_1$	$D_2$
主 语	$S_1$		CALL	READ WRITE		SEEK	
	$S_2$				READ		SEEK ⋮

项目  $A[S_1 O_3]$ ，存取类型操作表，主语  $S_1$  拥有对文件  $F_1$  的读写特权

图2 OS存取矩阵

4. **网络保密保护层** 我们从OSI参考模式加以考虑，它共分为物理层、链路层、网络层、传输层、会话层、表示层和应用层。可考虑在其中四个层次中加密。在物理层和链路层中可以实现点对点配置或多点配置中的链路加密，即采用H帧中正文加密硬件实现。对于联网的分布式数据库，应在网络第四层传输层上开发并发控制的管理机构及保密机构。这是因为传输层负责确定网络结点和接收它上一层会话层的数据，将其完整、准确地送至另一端。在OSI第六层表示层，它负责数据的转换。进行标准格式、标准文件和标准设备的转换。而数据加密实质上也是一种数据变换，因此可放在表示层一起实现。

在OSI参考模式中，表示层的功能是通过表示上下文来实现的。即由一个抽象语法及其相应的传输语法所组成的偶对，前者是对数据的一般性结构描述；而后者则定义数据在开放系统间传送时，体现数据的具体表示的数据形式化规范说明所用的规则。这样，可将加密功能与传输语法进行松散耦合；当需要时，它们合为一体，不但对数据格式进行转换、压缩，而且对它们进行加密，并按要求对表示规程控制信息中的某些字段加密；否则，旁路加密功能，只完成传输语法。

5. **数据库安全保密保护层** 在计算机系统中，信息资源都是集中存放在数据库中的，除了硬件、操作系统和网络等保护机构外，最困难的是对存放信息的数据库 (DB) 的安全性和完整性技术措施的研究。归纳起来，DB安全性保护措施有以下几种：对用户的鉴定；“事务”访问的检查；授权规划的定义；请求的批准；语义完整性的检查；对访问进行记录；审计跟踪；操作系统检查；硬件检查；物理卷保护；密码技术；统计DB的安全性；DB的恢复；分布式DB的并发控制等。

我们可以通过四个方面来分析DB的安全性和保密性、存取控制、流向控制、推断控制及密码控制。

存取控制解决“哪些用户对什么数据有何种存取权的问题”。我们可以通过建DB的安全性基本模式来实现“禁止非法访问DB”。用一个四元组 (s, o, t, p) 来表示一个存取规则，它指定：“主语s用存取操作类型t去访问那些谓词p为真的对象o”。在此基础上建立

各种扩充的DB存取控制安全性模式，以适应不同环境及不同密级的要求。流向控制，防止合法用户向第三者复制数据，可在DB的四元组中再引入两个元素，给数据赋密级 $sc_i$  (security class) 和给运行程序分配许可证 $ps_j$  (pass)，仅当 $sc_i \leq ps_j$ 时，程序 $j$ 才可以写数据 $i$ 。推断控制，它是系统中“统计DB”的保密对策。通过控制查询集合的规模、重合度和舍入，同时采用划分、交换、随机化、监视等措施，防止推断统计数据造成个体数据泄漏。虽有一些措施，关于统计DB的安全性仍然需要深入研究。密码控制，将数据内容密码化。大多数数据库管理系统 (DBMS) 都不同程度的全部或部分应用了这些技术，如 Honeywell 公司的 DM6 DB, Sperry 公司的 i100 网状 DB, 王安公司的 VS PAGE DB 等，它们都具有三级保密措施：(1) 用户注册；(2) 文件/目录分级；(3) 存取权限制。

6. **系统管理制度安全保护层** 对用户、操作员和程序员等制定规章制度；建立全系统的保密员，监督制度的执行。

7. **法律及社会环境保护层** 数据立法；网上数据流通法；建立各级的信息系统安全组织，如数据政策委员会、立法委员会、保密委员会；规定必须用国产机的部门，必须用自已开发的软件，违则犯法；计算机及网络的引进法等。

### 三、计算机信息系统安全保密的算法

不论是硬件、操作系统、网络，还是DB的安全保密都是离不开密码技术，特别是网络保密问题。传统的密码技术有置换、位移、分块加密等方法。从本世纪以来，科学和技术高度发展，为现代密码学的发展奠定了雄厚的基础。当代密码学的两个主要成果是：

1. 1977年美国国家标准局 (NBS) 正式颁布的联邦数据加密标准 (DES)；
2. 1976年，W. Diffie和M.E. Hellman建立的公开钥密码系统的新概念，以及随之而产生的各种实施方案。它标志着现代密码学的诞生。

数据加密标准DES最初由IBM公司提出，而其核心内容则是美国的Horst Feistel 1967年所提出的理论，IBM公司于1972年以LUCIFER的名称推出了该算法的原型。1974年，美国联邦政府为制订数据加密算法的标准，从已有的一批算法中选中了它。1980年，DES成为美国标准化协会 (ANSI) 的标准，即DEA-1。1984年ISO/TC97 (信息处理系统委员会) 成立了分技术委员会SC20，专门负责信息处理系统内使用的加密技术的标准化，它也准备直接采用DEA-1作为国际标准。DES采用分组乘积密码，它用56位密钥 (密钥总长64位，其中8位是奇偶校验位) 将64位的明文转换为64位的密文。DES的加密算法 (同时可用于解密) 通过初始换位 $IP$ ，首先将输入的二进制明文块 $T$ 变换成 $T_0 = IP(T)$ 。然后 $T_0$ 经过16次函数 $f$ 的迭代，最后通过逆初始换位 $IP^{-1}$ 得到64位二进制密文输出。在接收端按逆过程得到明文。从密钥长度可知，理论上要破译密钥需尝试约 $2^{56}$ 次，而且可定期不定期地更换密钥，使破译几乎不可能。

所谓密钥就是经过选择的一个秘密词 (码)，例如：

BEAUTIFIER  
0 1 2 3 4 5 6 7 8 9

当想表示1289时，可用EAER。BEAUTIFIER在这里就是所谓密钥。象这样一种钥密也称为“单钥密码系统”，不但钥要保密，而且容易被破译，这就是传统的密码系统。而公

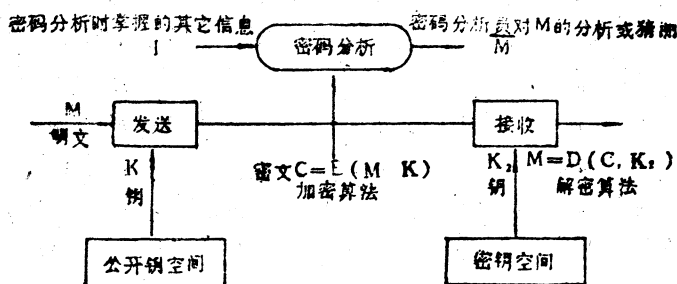


图3 公开钥密码系统信息流程图

开密钥为“双钥密码系统”，如图3所示。它的特点是不需要分发密钥的单独通道，也不能从一个键推导出另一个键，仅需要将解密密钥保密。在解密和加密交换上是容易计算的，即应属P；而在密码分析上是计算困难的，属于NP完全性问题。

但是，不管系统上开发了多么强的安全保密机制，若信息的保密问题不从法律，行政管理和技术系统工作全局通盘考虑，单纯依靠DB安全技术和网络密码技术那是靠不住的。因此，在保护层中划出了第一、六、七三个层次，见图1。

当今，已有一些厂家将上述的保护层中的第二、三、四、五层的部分或全部功能在单台或多台计算机中加以实现，并已投入实用。如美国 Honeywell 公司的安全保密通讯处理机。

#### 四、安全保密通讯处理机

美国 Honeywell 公司为解决计算机信息系统的安全保密问题推出了安全保密通讯处理机—SCOMP (secure communications processor)。美国国防部计算机安全保密中心在1984年9月批准SCOMP为A1级。美国国防部对计算机安全的等级分为四级：A1, C1, C2, D1。A1级是最高保密级。

SCOMP机是通过Honeywell公司的DPS6小型机的中档机改进而成的。SCOMP系统采用硬软件相结合的方式来解决计算机系统的安全保密问题，并形成多层次保密。在硬件方面，要对中央处理器（CPU）进行功能扩充，加上一个安全保护模块（SPM）和一个虚内存接口模块（VMIU）。

Honeywell SCOMP机既可以用在军方和政府来开发他们自己的应用软件，也可以用在其它需要数据安全保密的任何领域。SCOMP利用其软件对信息进行安全性划分，用户只可访问他们具有访问权并被充分验证后的信息部分。

##### 1. SCOMP安全保密特性

- 八个安全保密级和32个互相独立的安全保密类别，从而防止对数据的非特权访问。
- 8个数据完整性的级和32个互相独立的分类，从而防止对信息的非特权修改。
- 一个访问控制列表（ACL），定义谁可以读、写和执行的权限。
- 硬件形成多环保护机构，保护安全性软件。
- 段式虚内存。
- 扩展型安全保密审计。
- 一个可信任计算机系统，这是一个形式高级说明，通过使用当前可用的最先进技术来

完成验证。

## 2. SCOMP硬件

Honeywell的DPS6小型机的中档机是16位机，大总线的分布式结构，总体模块化，因此，将其改造成SCOMP机只需将原CPU用修改后的CPU替代，并增加一块SPM板，同时去掉科学指令处理器SIP和商用指令处理器CIP。

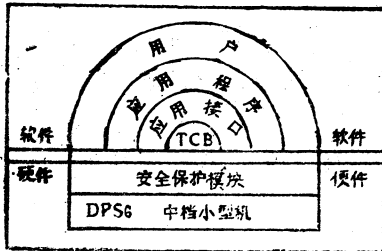


图4 SCOMP机安全保密特性

修改后的CPU变动了其微程序，同时用虚拟内存接口模块代替了原来的内存管理模块(MMU)。修改后的CPU插在DPS6的60英寸标准机柜的第一个槽上。

SPM包括一个大总线接口板和一个快速存取描述符存储板，它占大总线的第2个和第3个槽，并和CPU用内电缆互连。SPM驻留在DPS6上的CPU和其它功能模块之间，负责SCOMP的特性与原DPS6的特性功能之间的接口及调整工作。这样可使SPM捕获到所有

处理机请求，并在访问内存或I/O设备前形成对请求的调整工作。

SCOMP机硬件设计上仍支持全部DPS6机的外部设备，并通过专门的硬件提供安全保密机构。因此，很容易就可将DPS6转换成SCOMP机。

## 3. SCOMP软件

在SCOMP上运行STOP (SCOMP trusted operating program) 多用户操作系统。STOP是由一个TCB (保密算法及验证库) 和应用接口组成。TCB提供安全保密机制，而应用接口组成。TCB提供安全保密机制，而应用接口允许用户形成所希望的活动。

## 4. SCOMP的TCB

SCOMP的TCB是一个基本的操作系统，依据TCB库中的安全保密策略控制对目标和数据的访问。TCB利用参照监控的概念，用适当的语言写出，并且用最佳实施技术加以完成。

用户，系统管理员和操作员都通过保密安全接口与SCOMP系统对话。用户通过一条保密安全通讯路径访问TCB，它是由硬件和软件共同来确认一条路径的。安全保密路径用来防止用户由于要求得到应用软件 (象涉及到操作系统等内部资源一样的响应所引起的系统内外部的破坏及泄密。

TCB提供三种功能服务：安全保密用户服务、安全保密操作服务和安全保密维护服务。用户服务为用户提供到SCOMP系统的接口；操作服务为系统操作员提供运行该系统的必要能力；维护服务为系统管理员提供建造和维护SCOMP系统的功能。

对于系统中的每个目标，TCB包含两类数据——访问信息和状态数据。访问信息包含安全保密级和相应的分类，及完整性级和相应的分类。状态信息依目标类不同而变化。

## 5. SCOMP核心接口软件包

SCOMP核心接口软件包 (SKIP) 提供用户与应用程序的接口及系统与SCOMP安全保密机构的接口。SKIP提供三个基本的用户功能——一个层次文件系统；一个进程控制机

构和I/O外设的支持功能。SKIP允许用户通过形成类似TCB提供的某些功能来处理非文件系统段。为了保护文件系统的完整性，对从用户环直接调用文件的数目进行了限定。SKIP还允许用户创建和删除进程，置优先级，接收报文等等。这些功能为在一个应用环境中建造和管理进程提供了通讯能力。

## 6. SCOMP应用

SCOMP应用有三个方面：

- STAND ALONE C主机，它包括字处理，多级保密数据库和通用分时主系统。
- 安全保密前端机，它可以配置为单个主机安全保密监控器，也可以配置成多主机安全保密的转接器。
- 网络，在网络和通讯应用中，它可作为网络监控器，网络仿真控制器，网络接口及信息。

文献出处《系统工程与电子技术》1987年9期78—84页

# 关于运行程序的加密与反加密

上海计算机技术研究所 赵玉成

## 一、研究软件加密技术的意义

在计算机技术的发展史上，硬件和软件的发展历来是相互促进的。八十年代以来，微机、超级微机的迅速发展，使计算机的应用深入到社会生活、经济、科学各个领域，同时也为计算机软件的产品化创造了极为有利的条件。国外一些软件公司都陆续由原先为特定用户设计、开发专用系统为主，改而以开发通用的软件产品为主。这些软件公司出于其经济利益的考虑，对其开发的软件都进行加密处理，防止用户拷贝，以保护其公司的利益。

但是在国内，软件技术市场尚未能很好的形成。其主要原因之一是存放在软盘上的软件可经过拷贝而轻而易举地复制。因此，人们往往不愿花数百元，数千元买一个优质软件，而宁可等一年时间后无偿地拷贝、复制软件。这就造成这样一种情况，即一方面，软件开发是一种技术密集型，知识密集型的复杂劳动，开发一个具有一定规模，一定水平的软件需要付出艰巨的劳动，相当的代价，因此软件具有较高的价值。而且，随着硬件技术的不断发展，硬件生产效率的迅速提高，软件和硬件的价值比还在进一步上升。另一方面，由于前面提到的软件易复制性，一个软件开发单位研究制的软件，只要流出去一个付本，用户就有可能通过拷贝的途径，廉价地甚至无偿地取得这些软件。软件开发部门耗费大量资金，化了相当时间研究的软件，常常被轻易地复制走，因而软件产品往往无法很好地进入软件市场，不能真正成为商品，其价值得不到合理的体现。这种状况影响了软件开发部门再开发经费的来源，影响了软件研究人员的积极性。从全局来看，这种状况势必影响到我国计算机软件产业的形成和发展。

目前的状况是各软件开发部门无法得到软件产品开发所需的足够经费和必要设备，无法把主要力量放在能形成巨大经济效益的拳头产品的开发上，而是纷纷竞相为用户开发当前经济效益稍为明显的专用系统。这一方面大量地造成同一功能同一水平的重复开发，另一方面又由于专用系统的使用局限性较大，而我国当前软件开发人员的数量又较有限，因而不能满足广大用户的需求，再加上近年来我国进口了数万台微机，因而造成大量的计算机使用不足。就拿上海来说，大约三分之一的计算机买来后都未能得到很好的使用而处于闲置当中。

因此，为了保护软件开发部门的经济权益，促进软件开发人员的积极性，加速我国软件产业的形成和发展，使我国有限的软件人员发挥出更大作用，使有限的开发经费取得较好的经济效益，研究软件技术的保护迫在眉睫。国家现在正着手软件保护法的研究。在软件保护条例颁布之前，软件只能完全依靠软件加密技术来保护。而在软件保护法实行后，软件加密也仍然是软件保护的一个有效辅助手段。可以说，研究软件保护技术，保证软件开发者的权益，是一项必需而有相当意义的工作。

## 二、软件加密的技术基础

软件保护技术，也即通常所说的软件加密技术，可以分为两类。

(1) 数据加密。最典型的是网络通讯中的数据加密。此外，软件系统的源程序和其它文档资料都是字符，图形的某种集合，可以看成是某种形式的数据。因此，它们的加密也属此类。

(2) 运行程序的加密。这是本文讨论的主题。

目前，软件开发部门开发的软件，如果它能够作为产品的话，一般都是存放在软盘上销售给用户的。用户则往往通过文件复制命令或软盘复制程序复制出与原本盘完全一样的付本盘，而无限制地向外流散。我们把接受加密的软件称为目标软件（或目标系统），目标系统内各可运行文件则称为目标文件，目标文件内在运行时装入内存的程序代码称为该文件的装入模块（或目标程序）。运行程序加密的基本要求是，一方面，加过密的程序经过运行时译码后仍能如常运行；另一方面，对运行程序所加之密是不可复制的。这样即可使用户能正常使用其购买的目标系统原本，又能防止用户通过非法复制付本侵害软件开发部门的权益。

一般来说，要使运行程序不可复制，就要使拷贝程序不能读出该程序。那么命令解释程序又如何读出该程序呢？如果命令解释程序能正常读出该程序，那么为什么又能使拷贝程序无法复制该程序呢？

其实，软件加密的基础是在目标系统原本盘上产生这样一种信息，这种信息既是目标系统内各运行程序在运行过程中所必须引用的，又是各种文件复制命令或软盘复制程序所无法辨认的。这样，就使目标系统的运行完全依赖于载有目标系统的原本软盘，见图1。因而，用户也就无法用软盘复制程序复制目标系统的付本，或者说，用户无法复制出目标系统可正常运行的付本盘。

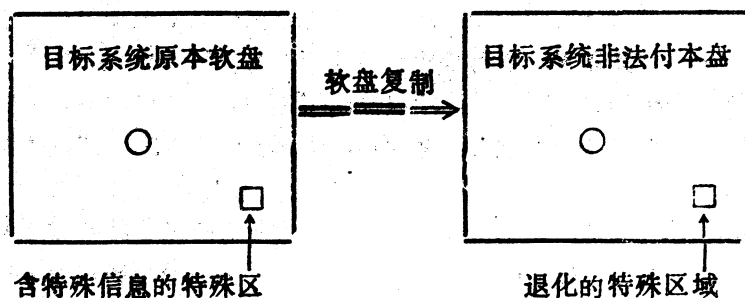


图 1

未经加密的运行程序的运行过程是：命令解释程序把可运行程序装入内存，然后即把控制转向该程序的入口处。而经过加密的程序，其代码前面配有一段译码程序，仅当译码程序在确认软盘上特殊区域中含有特殊信息并进行译码后才开始正常运行，若其发现特殊区域退化，则立即停止运行，见图2。

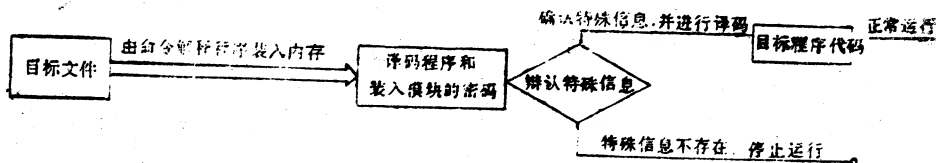


图 2

因此，软件加密最根本的基础，是对于软盘的较高的存取能力，在能够的软盘中写入和读出一般拷贝程序所无法辨认的信息的能力。

值得注意的是，为目标文件配置的译码程序必需保持原目标文件的结构，也即保证目标文件装入模块原先的可浮动性，并保持其原先的执行初始条件。这样才能保证目标程序的正常运行。

### 三、软件保护技术的保护技术

加密和解密是矛和盾的关系。有加密就有解密。任何一种加密方案都必须考虑反解密问题。

前面指出软件加密可以分为数据加密和运行程序加密两种。数据加密可以说与数据通讯中的密码系统基本上属于同一范畴。在密码系统中，数据的流程如图3。因此，对于数据加密来说，窃密者是无法直接窃得编码算法和译码算法的，他所能窃得的仅仅是密码，也就是说，窃密者只能根据他所窃得的密码来进行破译。

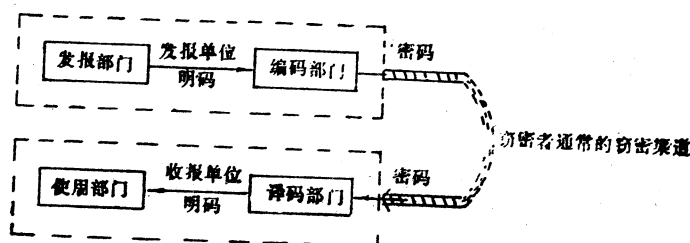


图3

运行程序的加密则不同。如前面所述，运行程序的基本加密要求是，它是不可复制的，又是可正常运行的。这就要求提交给用户的目标系统不仅要有用户所无法复制的密码信息，而且在运行程序内还必须配有译码程序，以便在运行时把密码转换成明码而正常运行。因此，对于运行程序来说，用户进行解密的依据不仅仅是密码，而且还包括了译码程序，也即译码算法。可以说，运行程序的加密与通常数据加密的不同之处，既在于加密对象的结构比较复杂，更在于它不仅要求其明码为外界所不可见，而且要求它的译码算法也必须为外界可用而不可见。译码算法的隐蔽问题，也即软件加密的反解密，反运行跟踪问题 实际上就是软件加密技术自身的保护问题，或者称为软件保护技术的保护技术。

一个有效的反解密方法，应该满足如下三个要求：

- (1) 译码程序是不可跳越的，不执行译码程序，程序就无法执行。
- (2) 译码程序是不可跟踪的，又是复杂的，隐蔽的，若硬件跟踪，程序就无法执行。
- (3) 不通过译码算法，密码是不可破译的。

要达到上述第一点，可以把运行程序代码本身全部通过某种编码算法转换成密码，只有当译码程序把运行程序的代码从密码形式转换为明码形式后，目标程序才能正常运行。这样，解密者就无法跳过译码程序来观察运行程序本身。

要达到上述第二点，则应采用这样一种译码算法，它的运行必然造成解密者跟踪环境的破坏。这样，解密者若不执行译码程序，他就无法观察和运行程序本身，但若他执行译码算法时却又破坏了跟踪环境，因而他仍然无法观察到运行程序本身。

此外，译码程序还可作进一步的判断，若判明自己在DEBUG控制下运行，它就马上停止执行，这就从根本上切断了解密者跟踪的可能性。



要达到上述第三个要求，就要求采用一种较完备的编码算法。满足上述前两点的加密方法，已保证解密者无法看到运行程序的明码，但他还是可以通过DEBUG的显示命令D看到放置于内存中的密码。对于破译专家来说，有了密码，他就可能通过摸索密码的编排规律解

出相应的明码。比如说，明码THERE通过映射

A	B	C	D	E	...	H	...	R	...	T	...	Z
S	T	U	V	W		Z		J		L		R

转换为密码

LZWJW。从编码学的角度来说，这个编码算法，它的密钥数为26。对于这种编码，窃密者窃得密码后可通过有限次的编排，总能摸索出编码规律，从而破坏译出明码。而且，当他截获的密码量越多，他破译的依据也越充分，破译的成功率也就越高。

而完备的编码算法则要求，不管窃密者窃得多少密码，他也无法通过有限次的编排摸索出编码规律。完备的编码算法，要求它的密钥数与编码量相对应。

采取了上述一系列措施，基本上排除了窃密者进行运行时跟踪的可能性，这样，我们就不仅防止了用户通过解密复制出其有效运行的付本，而且也防止了用户通过跟踪来窥探加密的原理和方法。通过软件保护技术的自保护，可以使得对目标软件的保护更切实有效。

文献出处：《上海微型计算机》1987年6期32—35页

# 推荐几种加密方法

福建师大物理系 吴庆祥

## 1. 使CATALOG失效。

做法是在您的HELLO程序的任意位置插入下列语句：POKE 42350, 96  
在HELLO程序中插有这个语句的磁盘启动后，CATALOG就失效了。

2. 恢复位时 (CTRL—RESET) 运行盘上的名为“H”的程序,做法很简单,只要在HELL程序中加上如下程序即可。

```
ILIST
```

```
100 POKE 40421, 28; POKE 40420, 76
```

```
120 POKE 40422, 03
```

```
130 FOR I=796 TO 006
```

```
140 READ A; POKE I, A
```

```
150 HEXT
```

```
160 DATA 32, 149, 160, 169, 200, 141, 117, 170, 76, 209, 164
```

```
170 END
```

运用这种方法可使您的程序不会被CTRL-RESET复置后而LIST出来，而盘上的文件名为“H”的程序可以进行一些加密处理，如使CATALOG失效等。

## 3. 防拷贝加密

以上所述的方法只是让使用者不能分析您的程序，而达不到不让别人复制的目的。我们通过分析DOS3.3的驱动器控制程序后，自己研制了一套控制磁头移动位置和磁轨的程序，用这程序对空盘进行加密处理，然后把要保密的资料存到这个加密盘上，通过实验发现这样加密的磁盘，用我们现有的十几个较高级的拷贝程序都不能拷贝。如果同时使用上述的方法，使CATALOG失效，使CTRL-RESET运行“H”程序。在文件名为“H”的程序中，设置一个按键作为出口（按其它任何键都不能中断，也不能退出），按下这个键后，将一切在机内的加密程序和加密痕迹都清除掉，返回正常的DOS状态，此后，对加密盘进行任何操作都不接受，只有重新启动才能运行加密盘。这样加密的磁盘，既不能拷贝，又不能调出盘上程序进行分析，从而提高了保密程度。由于程序很长不能在此加以详细的论述，有兴趣的读者可来信向本人索取。

# 软件加密及其实现的一种方法

中科院广州电子技术研究所 刁锐涛

由于计算机的发展迅速，价格也不断下降，因此凡有条件的单位都买了计算机或正准备买计算机。硬件是看得见摸得着的东西，一台机器不能为两个单位所有，要仿制也不是那么容易的，仿制出来的冒牌货也极易被发现，因此硬件的所有权问题比较好解决。但对于看不见摸不着的软件来说，其“版权所有”问题就很难做到了。人家辛辛苦苦研制出来的软件，几分钟的工夫就可被复制，复制出来的与原版的一模一样，不存在真假问题。正因为如此，软件加密的问题已越来越引起人们的重视，想了很多办法来解决这个问题。

## 一、一般的加密方法

给一台机器安装某个软件，一般来说有两种方式：一是将此软件的源模块全部给你，由用户自己去生成一个适合自己需要的系统，这种方式对用户来说是较好的，它比较灵活，但对软件加密问题就较难解决了。另一种方式是由软件研制者给用户安装，而不提供源模块，这样生成出来的系统是固定死的，不能再变，对用户来说较吃亏，但对软件的加密工作就提供了很有利的条件。

一般对前一种方式，加密的手段不是太多。给微机提供软件时，通常是在对盘的格式化方面做工作，使用操作系统提供的COPY程序不能使用，从而达到不让其它用户复制的目的。

对于后一种方式：采用的加密方法较多，如：①在安装软件时，要提供主机的number，安装好后运行这个软件时，它会与主机的number进行比较，若对不上，它就拒绝执行。这样，即使复制了这个软件，由于不同机的number不同，它也不能在其它的机上运行，这种办法是有效的，但不是在所有机上都能实现，在小型机上常采用此方法。②用隐文件名的办法：即是说提供给用户的软件在安装生成后产生一些隐形的文件，在列目录时用户看不见。这样在复制时，这些隐形文件没复过去，以致不能运行。③有的软件虽然可以复制过去，但不能运行。还有很多种方法，在此不能一一列出，下面只对其中的一种谈谈其实现办法。

## 二、一种加密的方法

在PDP-11小型机上研制了DTR的中文引导接口软件CGD是为了能推广应用中文。在转让CGD时，存在如何实现CGD的加密问题。对CGD的加密工作的前提是不向用户提供源模块，而到现场进行系统程序安装。即上面叙述的后一种方式的第③种办法。下面简要叙述一下这种办法的思路及其实现：

假如一个系统（比方说叫CCC）有若干个源模块。在安装CCC时，可设计成必须先运行一个初始化程序CCCINI·TSK，这个初始化程序对CCC是必要的，它必须且只需运行

一次即可，对于CCC的加密工作就可以放到CCCINI中去做。在系统的源模块中找一个可见的ASCII文件，假定文件名为A·TXT，在A·TXT中再选定一行（最好的注解行）。设此行为L。行L原先有n个字符。在CCCINI中，可以编一段加密程序，这段程序所做的工作是：将行L读出，然后在它末尾加上一个空格，再写回去，那么这行变成n+1个字符了（在CRT显示出来它仍然是个n字符）。当然完成这样的功能不仅是光对L行做工作，而必须将整个A·TXT作为顺序文件重新倒一遍，从而产生了另一个文件B·TXT，然后再将A·TXT删去，将B·TXT改名为·ATXT。做完这部分工作，用户并没有什么感觉，表面上好似没有任何变化，因为对于A·TXT的变化仅是行L末尾多加了一个空格，而这个空格在CRT显示时是根本看不出来的。如果系统CCC被复制到另一个新用户中去，他也一定要先运行CCCINI·TSK这个程序，则文件A·TXT的L行的末尾又加上了一个空格，L行变成n+2个字符了，这个新用户CRT上是看不出来的。但是在系统CCC的运行过程中，已加进了一段检查程序，看是否A·TXT文件中的L行的字符个数为n+1个，并且最后一个字符为空格。若是则为正常，CCC继续做下去，若不是，则进入死循环，并打印出信息，告诉用户这个CCC不是从研制单位复制来的。从而实现了CCC的加密工作。（即不允许用户再复制给他人）在PDP-11机上没有提供现成的反汇编程序，只要将加密程序及检查程序以机器码的文件形式提供（文件属性为·TSK），用户是不容易解密的。用这种方法来加密，并不太难，只要编两部分程序加到系统中去即可，而要解密却不太容易（对于系统有多个可见ASCII文件时更有效）。

### 三、结 语

这里要强调一下的是：对于精通软件的高级系统管理人员说来，任何软件的加密对于他们都是徒劳的，其作用是有局限性的。俗语说：“魔高一尺，道高一丈”。加密和解密都能通过一定的办法实现的。这只不过是时间问题和付出的代价问题罢了。但是对于一般软件人员及使用计算机最多的职能部门的人来说，它又是很有有效的，往往使人束手无策，必须去找软件研制所需要的系统。从另一方面看，即使人们能解密，如果付出的代价大于这个系统的价格，那么解密工作也就没有多少经济意义了，而自己的声誉方面还会受到影响，因此，探讨研制一些简便的软件加密措施对于保护研制者的利益是有一定意义的。也有利于软件的商品化。

# 软件加密保护方法

何建民

由于微电子技术的迅猛发展，微计算机的应用日趋普及，软件的商品化程度越来越高，软件交换的方式已由早期的以物换物形式蜕变成计价商品、公开出售。因此，对贮存在软磁盘上的软件加密，便成为软件工作者和电脑爱好者谈论的热门话题。

计算机所使用的软磁盘信息记录方式与录音带和录像带不同。录音带和录像带的信息是以ANALOG记录方式贮存的。因此，若对录音带和录像带一再复制的话，音响、音色及图象的效果会一次比一次差，最终将成为无法使用的复制品，而软磁盘就不同了，它是用DIGITAL记录方式存贮，复制品与源盘质量相同，使用过程中也不辨谁是原件，谁是复制品，这是计算机用户众所周知的。正因如此，软件研制者尤感软件加密的重要性、迫切性。

下面就目前国内外在微机上对软件加密常用的方法做一介绍，以飨广大读者。

## 一、位移动技术

这是一种比较早的软件加密方法，比如，将一数84848484...以二进制来写的话，就成为：

10000100100001001000010010000100.....序列。此时，再将此数后移三位来读就变为：24242424.....，若数据信息进来的话，移动若干位来读数据，对于不知挪位的软件复制者来说，读出的数据是未移动时的数据，从而实现了对软件的加密。

## 二、改变磁道转速

微机上的软磁盘在被驱动器的轮毂夹住，激活驱动器马达时，盘片即以300转/分的速度旋转。因此磁盘每转一周需200ms，故可贮存50K bit的信息，若将磁盘转一周的时间改变为需201ms，那么，就可以比原来多记忆250bit信息，软件加密便是利用这点差异实现的。一些磁盘机列出了ISV (Instantaneous Speed Variation)与LSV (Long Speed Variation)来表示瞬间与长期转速的最大变化量。一般为1.5%，Apple机驱动器约有1~2%的转速变化范围，故用盘前若不确认一下，在写信息后可能不清楚信息存放在哪一位磁道，加密就是利用这点预先在磁带上记忆有多少个字节，在读盘时测试一下，据此改变磁盘的转速。

## 三、磁道间距的不规则变化

在磁盘上存取信息是由软件控制步进电机使磁盘驱动器的读写头在磁道上来回移动完成的，此加密方法即为：将一个磁道以每 $\frac{1}{2}$ 个移动，但两道磁道间距须保持 $\frac{3}{2}$ 个磁道的间隔，加密便是根据磁道间距的变化来完成的，也即以 $1\frac{1}{2}$ 、 $2\frac{1}{2}$ 、 $3\frac{1}{2}$ 、 $4\frac{1}{2}$ .....等不规则的磁道间距，使复制工具不清楚是在使用哪一个磁道。经此加密后复制困难、能够有效的阻止

软件的快速无偿扩散，达到加密目的。

#### 四、对磁道的某些扇区不格式化

众所周知，微机上使用的软盘需要经过格式化后才能正常工作，存贮信息、复制备份，否则，就无法使用。加密的措施就是在格式化的盘中，对某些扇区作不格式化处理，使加密软件既能在系统下运行，又使拷贝工具无法正常工作。

#### 五、改变磁道密度

即改变软磁盘片上的某些磁道密度，使之在同一张盘上出现不同的密度，造成拷贝工具读写错误、使其不能正常工作完成复制，而使磁盘的软件受到保护。

#### 六、改变磁道格式

这种方法加密的软件一般是分离式的，自带系统支持该软件运行。即驱动器工作时直接引导盘上的系统支持自己的软件。这种盘经处理后其磁道格式与机器操作系统认可的磁道格式不尽相同，因此无法用通常的拷贝工具复制。此加密措施行之有效，若没有解锁盘只好望之作罢了。

#### 七、调换磁道信息

软磁盘上信息存放的物理位置与用户无关，计算机如何存入就如何原样取出。但若对存有数据的磁盘，把磁道上存贮的信息互相调换，计算机再取时就无法找到所要的数据信息了。比如有一目录文件的内容存贮在磁盘上第1、5、9、17、23、27、…道等等，若将这些道号上的内容同其道号加二的道号上的内容互换，即1、3道内容互换；5、7道内容互换，…等等，复制者在不知磁道信息内容被调换的情况下，即使复制成功，在没有解锁盘的帮助下也无法运行，从而被加了密。

#### 八、利用非用户使用道存贮信息

此加密措施现被广泛应用，在IBM-PC微机上0~39道为用户使用道，除此以外的道次均被认为是非用户使用道，若将一些重要的信息存放在这些道上，就能有效地阻止一些比较通行的拷贝工具非法复制。dBASE II就是在第40道上加的密。目前拷贝工具大多复制到【0~41道】和【0~43道】，有复制到更高的非用户使用道的拷贝工具持有人不多。因此，大多数加密也就是在这上面做做文章。

#### 九、激光孔加密

这是利用激光技术对软磁盘进行加密的一种物理方法。它主要利用激光的方向性好，亮

度高及光子简并度高的特点，把贮存在软磁盘磁道上的一些数据地址“冲掉”，形成激光孔（Laser hole）。使拷贝工具无法识别，不能正常工作。如dBASEⅡ、Lotus 1—2—3及一些游戏软件都是采用此方法加密的。

## 十、掩膜加密技术

这也是一种物理加密方法。国外近年来才发展起来的。广义地说，它与激光孔加密实现的目的是一样的，所不一样的是实施途径不同。此方法是通过膜“遮掉”数据地址，使拷贝工具无法识别，工作失败。

软件的加密方法和实现的途径很多，千变万化，共同的目的就是要阻止非法复制，故不能尽表详述。有些密可以被破解，但也有些密破解工作复杂，甚至无从下手。更有趣的是有些密自己能加却不能解，这都是有待探讨和研究的技术。

文献出处：《计算机世界》月刊1987年3期18—19页

# 一个反动态跟踪程序的破译方法

沈空通信处 褚玉清

## 一、引言

反动态跟踪是软件加密的一种技术。它是阻止技术人员破译软件的一个手段。一般来说，只要突破了加密软件的反动态跟踪程序，那么这个软件所加的密就基本解决了。本文介绍一个在IBM PC机上实现的反动态跟踪程序（简称为X）的破译方法，以期对希望获得这方面知识的人有所帮助。

## 二、X的反动态跟踪方法

通常，当我们开始分析一个新软件时，都是先用DEBUG的U命令将其程序清单打印出来，然后读这个程序（这叫作静态分析）。当读程序遇到一些中间数据或其它问题时，就会上机用DEBUG来了解程序的运行情况。这就是动态跟踪。

### 1. DEBUG的T命令和G命令

DEBUG的跟踪命令有T命令和G命令，其用法如下：

T命令

T [=address] [value]

T命令从address地址（缺省时从当前IP）开始执行程序，共执行value条指令（缺省时为1），然后停下来。每条指令执行后便显示一次运行状态。

G命令

G [=address] [address [address]]

G命令在每一个address处设置断点，然后从当前IP（或由=address指出的地址）开始执行，当遇到一个断点时，就停下来，并显示当前的状态。

T命令和G命令是跟踪程序运行的基本命令。

### 2. BIOS的0号中断和3号中断处理程序

BIOS的0号中断程序完成单步中断处理；3号中断程序完成断点中断处理。DEBUG的T命令和G命令就是通过调用这两个中断处理程序实现的。

### 3. X的反动态跟踪方法之一

由于用DEBUG进行动态跟踪的基本命令只有T命令和G命令，而这两条命令又是通过调用BIOS的0号和3号中断处理程序实现的，因此，只要破坏BIOS的0号和3号中断处理，使DEBUG的T命令和G命令不能正常运行，这样就可以起到一定的反动态跟踪作用。X的反动态跟踪方法之一正是采用了这种措施。它在程序的一开始就将BIOS的0号和3号中断处理



程序地址(该地址在 0 : 0 和 0 : E 单元)改为 X 内部的一段程序的地址。当跟踪者使用 G 命令的 T 命令进行跟踪时,就会自动地通过 BIOS 掉入这个事先设好的陷阱。

#### 4. X 的反动态跟踪方法之二

上述方法一旦被识破,跟踪者就可以通过一些方法越过这一段修改 BIOS 中断地址的程序,使跟踪继续下去。可以想象,仅仅用一种方法来阻挠破译者跟踪是不够的。

X 所采用的反动态跟踪方法之二是对反动态跟踪程序本身进行软加密。所谓软加密,即将运行的程序由明文变为密文,这样,破译者用 U 命令反汇编出来的东西就是一堆难以看懂的东西了。

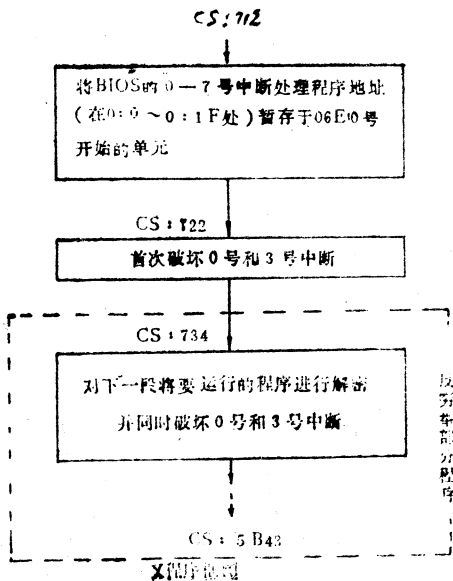
X 所采用的软加密是变形法,它用 XOR 指令对程序的每一个字节进行两次操作。这样,所形成的代码根本无法辨认,也不能直接运行。因此,在这段程序运行之前,必须对其解密。

#### 5. X 的反动态跟踪方法之三

软件加密使破译者难以将程序全部打印出来进行静态分析,因此,给破译者造成一定的困难。但是如前所述,软加密程序运行之前,必须对其进行解密。这样一来,就等于向破译者告了密。破译者俱此就可以了解到下一段的结果,并能够继续跟踪下去。

由于这个原因,所以 X 又采用了第三种反动态跟踪措施:反穷举。这一段程序的长度为 20 多 K,它以 30 个字节为单位,不断地用上一程序单位生成下一程序单位(也即解软加密),破译者只有一个程序单位一个程序单位地跟踪,才能继续下去。但是这样做是很困难的,因为此间需要跟踪 700 多步,这对于一个人的精力来说是很难胜任的。更何况,跟踪过程中稍有差错,就会前功尽弃。

### 三、X 的程序分析及破译方法



#### 1. 程序框图

从程序实现的步骤来看, X 主要由两部分组成。一部分是破坏 BIOS 的 0 号和 3 号中断处理程序;另一部分则是构成反穷举的大规模解软加密过程。

需要指出的是,程序中除了破坏 0 号和 3 号中断外,同时也破坏了 1、2、4~7 号中断。这样做有两个作用,一是程序中需要借用这些单元;二是起到迷惑破译者的作用。实际上,在整个反动态跟踪程序运行过程中,没有涉及到 1、2、4~7 号中断,因此破坏了它们对程序无任何影响。

## 2. 破译方法

根据上述程序分析,破译时也应从两个方面着手,一是想法避开程序破坏0号和3号中断;二是寻找反穷举部分程序的规律,以便找到它的终止点。

1) 怎样避免破坏0号和3号中断

先来看X的开始部分程序。

```
u cs:712 733
0913:0712 33C0      XOR    AX, AX; AX寄存器清0
0913:0714 8ED8      MOV    DS, AX; 置数据段地址为0
0913:0716 0E          PUSH   CS
0913:0717 07          POP    ES; 将附加段地址置成与程序段相同
0913:0718 BFE006     MOV    DI, 06E0; 使目的变址寄存器指向06E0
0913:071B 8BF0      MOV    SI, AX; 使源变址寄存器指向0
0913:071D B92000     MOV    CX, 0020; 传送20(16进制)次
0913:0720 F3        REPZ; 将0:0~0:F之间的内容保存
0913:0721 A4        MOVSB; 到CS:06E0开始单元
0913:0721 33C0      XOR    AX, AX
0913:0724 8EC0      MOV    ES, AX; 附加段寄存器为0
0913:0726 8BF8      MOV    DI, AX; 目的变址寄存器为0
0913:0728 8CC8      MOV    AX, CS
0913:072A 8ED8      MOV    DS, AX 数据段等于程序段
0913:072C BE0007     MOV    SI, 0700; 源变址寄存器为700
0913:072F B91000     MOV    CX, 0010; 传送10次(16进制)
0913:0732 F3        REPZ; 将700~710单元的内容传送
0913:0732 A4        MOVSB; 到0:0~0:10单元
```

### 程序1.1

0722后面的一段程序即是破坏0号~3号中断的程序。为了避免0号和3号中断被破坏且又能使下一段软解密用上0:0~0:1D号单元的数据,可以采用这样的方法,即将这段数据转存到另一些单元中(比方说06E0开始的单元中)。这只要修改后一部分程序就可做到。修改后的程序如下:

```
-u 722 733
0913: 0722 0E      PUSH   CS
0913: 0723 07      POP    ES; 附加段指向程序段
0913: 0724 BFE006     MOV    DI, 06E0; 目的变址寄存器指向06E0
0913: 0727 90      NOP
0913: 0728 8CC8      MOV    AX, CS;
0913: 072A 8ED8      MOV    DS, AX; 数据段等于程序段
0913: 072C BE0007     MOV    SI, 0700; 从700号单元开始传送
0913: 072F B91000     MOV    CX, 0010; 传送10次。下一段程序将使用1D个字
```

节,其余D个字节已在上面0722前的那部分程序传送过。

```
0913: 0732 F3      REPZ
0913 2 0733 A4      MOVSB
```

### 程序1.2

经这么一改,就可以用DEBUG的T命令和G令跟踪一段程序了

#### 2] 怎样解决反穷举部份程序

从程序地址CS:0734至CS:5B43是反穷举部分程序。它由700多个程序单位组成,每个程序单位通过解软加密来生成下一程序单位的程序。

下面我们通过列出前2个程序单位来看一看它的基本面貌:

```
-u 734 770
```

```
0913: 0734 1E      PUSH  DS
0913: 0735 33C0     XOR   AX,AX
0913: 0737 8ED8     MOV   DS,AX      : 数据段等于0
0913: 0739 33DB     XOR   BX,BX
0913: 073B B91D00    MOV   CX,001D    : 循破1D次
0913: 073E BD5207    MOV   BP,0752    : 软加密程序地址
0913: 0741 2E      CS,
0913: 0742 A05107    MOV   AL,[0751]  : 解密参数
0913: 0745 3007     XOR   [BX],AL    : 第一个求异或,存入0:00:1D
0913: 0747 2E      CS,
0913: 0748 304600    XOR   [BP+00],AL:第二次异或操作,解出一个字节
0913: 074B 43      INC   BX
0913: 074C 45      INC   BP          : 为下一个字节解密作准备
0913: 074D E2F6     LOOP  0745       : 判解密是否完毕,未完继续,反之转入下一程序单位。

0913: 074F EB01     JMP   0752
0913: 0751 50      PUSH  AX
0913: 0752 63      DB    63
0913: 0753 90      NOP
0913: 0754 DE88638B FIMUL WORD PTR [BX+SI+8B63]
0913: 0758 E94D50    JMP   57AB
0913: 075B ED      IN   AX,DX
0913: 075C 3F      AAS
0913: 075D 57      PUSH  DI
0913: 075E 7EF6     JLE  0750
0913: 0760 3E      DS,
0913: 0761 57      PUSH  DI
0913: 0762 60      DB    60
```

```

0913 : 0763 57          PUSH    DI
0913 : 0764 7E60        JLE     07C6
0913 : 0766 16          PUSH    SS
0913 : 0767 50          PUSH    AX
0913 : 0768 1315        ADC     DX, {DI}
0913 : 076A B2A6        MOV     DL, A6
0913 : 076C BB51B3      MOV     BX, B351
0913 : 076F D023      SHL     BYTE PTR [BP+DI], 1

```

#### 程序2.1

0751前面的一段程序是一个程序单位，它是0751后面那一段软加密程序的解密程序。需要注意的是，解密程序借用了0:0~0:1D单元的数据，这样做既起到生成下一程序单位的作用，又同时破坏了0~7号中断程序。

在上面“怎样避免破坏0号和3号中断”里曾经用06E0开始的单元代替：0:0~0:1D单元。这里为了使追踪进行下去，我们仍需要这样做，下面我们来改写0751前面的那一段程序，使之变为：

```

-u 735 750
0913 : 0735 0E          PUSH    CS
0913 : 0736 1F          POP     DS
0913 : 0737 BBE006        MOV     BX, 06E0
0913 : 073A 90          NOP
0913 : 073B B91D00        MOV     CX, 001D
0913 : 073E BD5207        MOV     BP, 0752
0913 : 0741 2E          CS,
0913 : 0742 A05107        MOV     AL, {0751}
0913 : 0745 3007          XOR     [BX], AL
0913 : 0747 2E          CS,
0913 : 0748 304600        XOR     [BP+00], AL
0913 : 047B 43          INC     BX
0913 : 074C 45          INC     BP
0913 : 074D E2F6          LOOP   0745
0913 : 074F EB01          JMP     0752

```

#### 程序2.2

经修改后，即可用G命令和T命令将追踪进行到地址CS:74F处。当程序运行到74F后，就可以清楚地看出，从752~76F这一段程序，除转移地址不同外，其它与735~74F完全一样：

```

u 752 76c
0913 : 0752 33C0          XOR     AX, AX
0913 : 0754 8ED8          MOV     DS, AX
0913 : 0756 33DB          XOR     BX, BX

```

```

0913 : 0758 B91D00      MOV    CX, 001D
0913 : 075B BD6F07      MOV    BP, 076F
0913 : 075E 2E          CS;
0913 : 075F A06E07      MOV    AL, [076E]
0913 : 0762 3007      XOR    [BX], AL
0913 : 0764 2E          CS;
0913 : 0765 304600      XOR    [BP+00], AL
0913 : 0768 43          INC    BX
0913 : 0769 45          INC    BP
0913 : 076A E2F6      LOOP  0762
0913 : 076C EB01      JMP   076F

```

### 程序2.3

如果按上述方法再往下走几个程序单位，将会得到同样的结果。可想而知，若按照上述方法一步一步来追踪，即便是几天几夜不休息，也很难走完这段反穷举过程。

要破反穷过程 关键是要找到它的终止点。一旦找到它的终止点，就可以越过这段反穷举过程，从而达到破译的目的。

通过对反穷举程序（软加密）的静态分析，可以发现，它是有规律可循的。比方说，下面一串数据：

```
79 2F C4 2C 4E EA F7 4A
```

在内存CS：751~CS：FFFF中出现了多次，用DEBUG的搜索命令S，

```
-S CS:751 FFFF 79 2F C4 2C 4E EA F7 4A
```

搜索，找到它们的地址在：

```
07AB, 0D38, 157D, 1E53, 23E0, 2C25, 34FB, 3A88, 42CD, 4FB7, 58E4,
```

由此可以估计反动态跟踪程序大约在CS：58E4至CS：6000之间结束。

由于其数据相同，因此我们就可以假想这些相同数据之间的程序是重复循环的，这样，我们就可以越过前面的穷举过程，从最后一个循环开始穷举。如果这样能进行下去，就可以大大缩短穷举过程。

事实上，这样是进行不下去的。这是因为反穷举过程的软加密是由两个参数决定的，一个参数在反穷举程序内，另一个参数则是在0：0~0：1D号单元中。由于第一个参数已在程序中，所以我们不必考虑它。而第二个参数对于所假想的循环则是不同的，因此不能简单地跳到最后一个循环，必须找出最后一次循环运行开始时：0：0~0：1D单元的内容。

找0：0~0：1D单元的内容可以这样解决：编写一段显示0：0~0：1D单元内容的程序，编好后，将其二进制代码进行异或操作，再用DEBUG的命令存入最后一个循环所在的存贮区，然后，从程序的开头运行，这样，当程序进行到最后一个循环时，便显示出当时的0：0~0：1D单元的内容。

一旦知道了最后一个循环执行前0：0~0：1D单元的内容，就可以用JMP指令从第一个循环之前跳到最后一个循环。在最后一个循环执行之前，需要做两件事：一是将最后一个循环的程序改成与第一个循环的程序一样（但需要改动破坏0~7号中断部分程序），二是将找出的0：0~0：1D单元的内容写入06E0单元。

如果最后一个循环能进行下去，那么反动态跟踪程序X就可以说基本破译了。

#### 四、结 束 语

反动态跟踪技术是当今软件加密的一个潮流，目前，市面上所有的加密软件基本上都有反动态跟踪措施。因此，要想破译加密软件，首先必须破译反动态跟踪程序。

不同的加密软件，其反动态跟踪措施多有不同，要想找到普遍的规律是比较困难的。因此，软件人员只有充分利用自己的聪明才智，才能将加密软件破译。

文献出处：《小型微型计算机系统》1988年2期41—45转27页

# 微型计算机的序列加密方法

工业自动化系 王志良 李 纪

**摘要** 根据微机的特点,本文提出了一种产生伪随机序列的新方法,用此法产生的伪随机序列可消除 $m$ 序列间存在的线性递推关系,从而可有效地用于信息加密,用这种序列在微机上开发了信息加密功能,收到了很好效果。

## 一、概 述

微型计算机目前在我国的工业、农业、商业、军事、银行、企业管理、科研和通信等领域获得了日益广泛的应用。信息加密保护是微机在这些应用中不可缺少的功能。从微机发展的水平来看,信息加密处理中采用序列加密方法是比较合适的。信息的安全保密技术非常丰富,有各种加密算法可供采用。但因微机的运算速度有限,很多加密方法因其算法复杂而难以在微机中获得实际有效的应用。序列加密算法相对说来较为简单,易于在微机上有效地实现。因此,对在微机中的序列发生方法及序列加密技术进行深入研究是有实际意义的。

早在四十年代末期,Shanon就证明了只要用与信息量相同数量的随机序列作密钥对信息进行加密处理,则加了密的信息就不可能被破译。但是,真正随机的序列实际上是不能用来对信息进行加密处理的,因为随机序列是不可能按要求重复产生的。所以只能用伪随机序列对信息进行加密处理。近几十年学术界对伪随机序列进行了充分的研究,特别是关于线性反馈移位寄存器(LFSR)发生序列的方法、性质等有了较完备的理论。然而,关于非线性反馈移位寄存器和在一个或几个LFSR上加一个非线性前馈逻辑网络产生序列的方法,迄今仍未取得令人满意的结果。LFSR序列直接用于信息加密处理还遇到一个很大的难题,即它的各位(bit)之间存在线性递推关系,破密者只要截取 $2n$ ( $n$ 为LFSR寄存器的级数)位的连续的明、密对照的报文,或者 $2n$ 位的连续的LFSR序列,就能很容易地推导出整个LFSR序列,从而破译全部密文。由此可见,如何设法改善LFSR序列这个致命的弱点,是序列加密方法的关键问题之一。

为克服LFSR序列的这个不足之处,学术界过去总是试图在非线性反馈移存器、移存器的前馈以及反馈逻辑结构的研究上取得突破,但迄今未获得一般性的结论。现在我们用微机来产生序列,进行信息加密处理,可以充分利用微机灵活的算术逻辑运算功能,以克服LFSR序列的不足,产生符合信息加密需要的序列。为此,首先分析一下,对二元随机序列作一些算术逻辑运算将会对它的随机性产生什么影响。

## 二、二元随机序列的定义及其运算

定义一: 设  $\{a_i\}$ ,  $i = 1, 2, \dots, n, \dots$ , 为一个取值为  $(0, 1)$  的二元序

列, 若每一位的取值满足  $p(a_i=1) = p(a_i=0) = \frac{1}{2}$ , 则称该序列  $\{a_i\}$  为二元随机序列。

根据微机的算术逻辑运算, 定义两种二元序列的运算。

定义二: 两个取值为  $(0, 1)$  的二元序列  $\{a_i\}$ ,  $\{b_i\}$ ,  $i=1, 2, \dots, n, \dots$  的异或运算是指它们的每一对应位都作异或运算, 即  $S_i = a_i \oplus b_i$ ,  $i=1, 2, \dots$ , 记为

$$\{S_i\} = \{a_i\} \oplus \{b_i\}.$$

定义三: 两个取值为  $(0, 1)$  的二元序列  $\{a_i\}$ ,  $\{b_i\}$ ,  $i=1, 2, \dots$  的加法运算是指二序列从低位到高位作带进位的二进制加法, 即  $S_i = a_i + b_i + c_{i-1}$ , 记为

$$\{S_i\} = \{a_i\} + \{b_i\}$$

其中  $C_{i-1}$  是前一位加法运算产生的进位。

关于二元随机序列作上面定义的两运算, 有下述二条定理:

定理一: 两个二元随机序列  $\{a_i\}$ ,  $\{b_i\}$ ,  $i=1, 2, \dots$  作异或运算后, 得到的  $\{S_i\} = \{a_i\} \oplus \{b_i\}$  也是随机序列。

证明: 根据定义一, 若对于  $\{S_i\}$ ,  $i=1, 2, \dots$ , 存在  $p(S_i=1) = p(S_i=0) = \frac{1}{2}$ 。

则定理得证。表 1 为该运算的真值表。根据真值表计算  $p(S_i=1)$  如下:

$$p(S_i=1) = p(a_i=0)p(b_i=1) + p(a_i=1)p(b_i=0) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}.$$

表 1  $S_i = a_i \oplus b_i$  的真值表

$a_i$	$b_i$	$S_i = a_i \oplus b_i$
0	0	0
0	1	1
1	0	1
1	1	0

$$\text{则 } p(S_i=0) = 1 - p(S_i=1) = \frac{1}{2} = p(S_i=1)$$

由于  $i$  可以取任意的正整数, 故  $\{S_i\}$  是随机序列。证毕。

定理二: 两个二元随机序列  $\{a_i\}$  和  $\{b_i\}$ ,  $i=1, 2, \dots$ , 作加法运算  $\{S_i\} = \{a_i\} + \{b_i\}$  后, 得到的序列也是随机序列。

证明: 设加法运算是对第  $i$  位进行的, 即  $S_i = a_i + b_i +$

$C_{i-1} \pmod{2}$ , 其中  $C_{i-1}$  是前一位运算结果的进位。

表 2  $S_i = a_i + b_i + C_{i-1}$  的真值表

$a_i$	$b_i$	$C_{i-1}$	$S_i = a_i + b_i + C_{i-1}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

为不失一般性, 设

$$p(C_{i-1}=1) = \frac{r}{m}$$

$$\text{则 } p(C_{i-1}=0) = \frac{m-r}{m}.$$

由  $S_i$  真值表 (表 2) 可以算出  $p(S_i=1)$  如下:

$$F(S_i=1) = p(a_i=0)p(b_i=0)p(C_{i-1}=1) + p(a_i=0)p(b_i=1)p(C_{i-1}=0) + p(a_i=$$

$$1)p(b_i=0)p(C_{i-1}=0) + p(a_i=1)p(b_i=1)p(C_{i-1}=1)$$



$$= \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{r}{m} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{m-r}{m} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{r}{m} = \frac{2m}{4m} = \frac{1}{2}$$

由此可见,  $p(S_i = 1) = p(S_i = 0) = \frac{1}{2}$ ,  $\{S_i\}$  序列是一个随机序列。证毕。

上述定理虽然是从随机序列的运算中得到的, 但对于最大  $m$  伪随机序列 (以下简称  $m$  序列, 可用 LFSR 移存器产生。) 也是基本适用的。这是因为  $m$  序列在其一个周期中 1 元出现的次数比 0 元出现的次数多 1 次, 只要  $m$  序列的周期取得足够长, 就可以认为 1 和 0 出现的次数是近乎相等的, 因而在  $m$  序列中每一位的值取 1 还是取 0 的机会是均等的, 上述定理也是基本适用的。此外,  $m$  序列经过移位后仍是  $m$  序列, 故在  $m$  序列移位后进行运算并不影响所得结果的随机性。

以上分析为我们提出的产生加密序列的新方法提供了数学基础。

### 三、用微机产生加密序列的方法

为了用微机产生加密序列, 首先在内存里建立几个级数互不相同的线性反馈移位寄存器, 用以产生几个周期各异的最大  $m$  序列。微机是以八位或十六位甚至更大的字长为单位进行运算的。为了提高效率, 我们每次使各移存器仅反馈移位一位, 且其中至少应有一个移存器的移位方向与其余移存器的移位方向相反。每次移位使各移存器内容更新后, 就以微机的运算字长为单位, 将各移存器的全部或某一部分内容取出, 在这些被取出的部分之间, 作加法和异或运算。这样将每次运算的结果排列起来, 就构成了一个新的序列。这个新序列完全克服了  $m$  序列作为加密序列的缺点。下面举例说明。

设在某微机的内存中已建立了两个分别为九级的移存器 A: 

$a_8$	$a_7$	.....	$a_1$	$a_0$
-------	-------	-------	-------	-------

 和十级的移存器 B: 

$b_9$	$b_8$	.....	$b_1$	$b_0$
-------	-------	-------	-------	-------

。为了使它们向彼此相反的方向移位, 可取两个

反馈系数为  $a_0 = a_8 \oplus a_4$  和  $b_0 = b_9 \oplus b_3$ 。若在某次移位后移存器 A 和 B 的内容分别为:

A: 1 0 0 1 0 1 1 1 0      B: 0 1 0 0 1 1 1 0 1 0

则在此后的几次移位过程中 A 的内容变化为:

100101110 → 001011101 → 010111011 → 101110111 → 011101110

而 B 的内容变化为:

0100111010 → 1010011101 → 0101001110 → 1010100111 → 1101010011

显然, 单从移存器 A 或 B 的每次移位结果观察, 移存器内容的变化很有规律。尽管采用增加移动位数的方法使每次移动后移存器的内容变化看起来很大, 然而并不能从实质上消除移存器内容之间的线性反馈移位关系。因此, A 或 B 的内容均不能直接作为加密密钥使用。

若每次从移位后的 A、B 移存器内容中取出右边八位作二进制加法运算, 并假定第一次运算之前进位位为零, 则可得到如下的运算结果:

01101000 → 11111010 → 00001001 → 00011111 → 01000001

这是一个新的序列, 它已不再象  $m$  序列那样变化有规律了, 而是变化无常, 令人捉摸不透, 仅从外观上观察, 每八位的变化具有很大的随机性。从整个序列的角度看, 由于它是从

两个 $m$ 序列经移位后作加法运算产生的, 根据前面提到的定理和 $m$ 序列的随机性质, 可以肯定, 这个新序列具有良好的整体随机性。由于该序列是两个周期序列间作加法运算后产生的, 它必然也是一个周期序列。此外,  $m$ 序列的可重复性也保证了这个新序列可重复产生。由此可见, 这个序列能满足加/解密对序列的要求。

为了增加序列的复杂度, 一般要由三个以上的 $m$ 序列经加法和异或混合运算以产生加密序列。由于要求各个 $m$ 序列的周期应各不相同, 产生出的加密序列的周期为参与运算的各 $m$ 序列的周期的公倍数。因此, 只要适当地选取各 $m$ 序列的周期, 就能很容易地使它们的周期公倍数足够大, 从而使加密序列的长度足以复盖所要加密的信息量。

#### 四、密钥破译可能性分析

就序列密码方法而论, 密码的安全保密性完全取决于密钥序列的保密程度。密钥序列一旦被破译, 就能轻而易举地获取明文信息。反之, 没有密钥序列, 就无法破译密码。所以, 密钥序列是否经得起破译分析, 是设计序列密码至关重要的课题。一个序列要经得起破译分析, 它除了应具有良好的随机性和足够长的周期之外, 还应具有很高的复杂性。 $m$ 序列由于它具有线性递推关系, 降低了它的复杂性, 故不能直接充当加密序列。前面已经论述过, 用 $m$ 序列经加法运算产生的序列具有良好的随机性, 且其周期较长。下面着重分析它的复杂性。

某课题或算法的复杂性研究已构成专门学科——复杂性理论。求解一类课题时所需要的最小时空耗费, 称之为该类课题的“计算复杂性”, 或称为“课题复杂性”, 简称复杂性。

序列的复杂性系指破密者为破译该序列必须付出的时空耗费。本文首次提出的新方法所产生的序列的复杂性分析, 可以分为三种情况进行。

##### 1) 破译者仅截取了全部密文

密文是由明文和密钥序列作运算(如异或运算)产生的。密钥序列具有的随机性使密文白噪声化了, 破译者未掌握明文或密钥序列的结构, 是无法通过密文用数学方法求得密钥序列或者明文的。

##### 2) 破译者截取到部分明文和全部密文

这是所谓“明、密文联合攻击”的情况, 也是序列密码所要考虑的最坏情况。破译者据此能获得密钥序列的某一连续部分、诸如:

$$S_i, S_{i+1}, \dots, S_{i+m}$$

并且知道该序列是由两个 $m$ 序列的对应部分

$$a_i, a_{i+1}, \dots, a_{i+m} \text{ 和 } b_i, b_{i+1}, \dots, b_{i+m}$$

作二进制加法操作之后产生出来的。破译者若能通过 $\{S_k\}$  求出 $\{a_k\}$  和 $\{b_k\}$ ,  $k=i, i+1, \dots, i+m$ , 进而求出 $m$ 序列 $\{a_i\}$ ,  $\{b_i\}$ ,  $i=1, 2, \dots$ , 他就能获得整个密钥序列 $\{S_i\}$ , 从而破译全部密文。为此, 破译者必须列出如下方程组:

$$\begin{cases} S_i = a_i + b_i + C_{i-1} \\ S_{i+1} = a_{i+1} + b_{i+1} + C_i \\ \vdots \\ S_{i+m} = a_{i+m} + b_{i+m} + C_{i+m-1} \end{cases} \quad (\text{mod } 2) \quad (4-1)$$

并求解其中全部变量，在方程组(4-1)中，序列  $\{a_k\}$ ， $\{b_k\}$ ， $k=i, i+1, \dots, i+m$ ，是有线性递推关系的，因此，它们的独立变量是有限的，其余的都可用独立变量的某种线性组合来表示。然而， $C_i$ 与 $a_i$ ， $b_i$ ， $C_{i-1}$ 的关系是不能用线性方程来表示的。如果将(4-1)当作线性方程组处理，则只得将 $C_{i-1}$ ， $C_i$ ， $\dots$ ， $C_{i+m}$ 等看作未知变量。其结果，方程增多，未知变量随之也增多，且未知变量始终比方程式的数量多。由此可见，线性方程组(4-1)是无解的。

根据二进制加法规则，可以求出 $C_i$ 与 $a_i$ ， $b_i$ ， $C_{i-1}$ 之间存在如下关系：

$$C_i = (a_i \wedge b_i) \vee (a_i \vee b_i) (C_{i-1}) \quad (4-2)$$

方程式(4-2)是一个非线性方程。若将式(4-2)代入方程组(4-1)，也不能完全消去式(4-1)中的所有 $\{C_k\}$ ，反而会使(4-1)变成一个非常复杂的非线性方程组。试图求解这个方程组，实际上是解非线性方程 $f(X_1, X_2, \dots, X_n) = b$ 的课题，也即布尔方程一般解的课题。在复杂性理论中称它为NP完全问题。

以上分析表明，在明、密文联合攻击的条件下，也无法采用数学分析方式解开我们前面提出的加密密钥序列的，亦即用这种序列对信息加密处理是十分安全的。

### 3) 破译者被迫采用穷举法

破获上述序列的唯一办法是猜测 $n$ 个 $m$ 序列的初始值(状态)。由于产生加密序列的 $m$ 序列数量和它们的级数不同，加密序列会千差万别，致使猜测时没有规律可循。只得采用逐一测试的穷举法。只要若干个产生 $m$ 序列的移存器级数的公倍数足够大，就能使这种破译法实际上成为不可能。设破译者拥有强大的计算能力，很高的计算速度，例如他能以每秒 $10^6$ 次的速度进行搜索试探，在 $2^n$ ( $n$ 为若干个移存器级数的公倍数)个可能的状态中找到唯一解的时间耗费如表3所示：

表3 设定 $n$ 值时，时间耗费估计(计算一步/ $\mu s$ )

$n$	20	40	50	64	100	200
耗费时间	1s	13年	35年	$4 \times 10^8$ 年	$4 \times 10^{14}$ 年	$4 \times 10^{44}$ 年

表3说明，当公倍数 $n \geq 64$ 时(在微机中这点很容易做到)，用穷举法破译不仅需付出巨额时空耗费，即使破译得逞，获得的信息已经成为历史，没有价值了。

综上所述，本文提供的设计序列密钥的方法能使产生的加密序列具有足够的复杂性( $n \geq 64$ 时)；使用这种序列对信息进行加密处理(变换)，能使密码具有足够的安全保密性。

## 五、在微机中实现新序列加密的效果

设新序列为 $X_i = (x_{i1}, x_{i2}, \dots, x_{ik})$ ，又设待加密的信息为 $Y_i = (y_{i1}, y_{i2}, \dots,$

$y_k$ )，其中 $k$ 为微机中运算时采用的字长； $i=1, 2, \dots, n$ ，则为了和微机中运算字长相匹配，可以采用如下的微机加密算法：

加密运算： $S_i = Y_i \oplus X_i$  其中 $S_i$ 为密码

解密运算： $W_i = S_i \oplus X_i = Y_i \oplus X_i \oplus X_i = Y_i$

我们在APPLE—II和IBM—PC微机上分别用它们的汇编语言研制了相应软件，实现了上述加密/解密方法。通过运行，取得了如下效果：

- 1) 对输入密钥的反应灵敏。密钥稍有不符，就不能解开密文。
- 2) 信息加密处理的效率很高。在IBM—PC微机上，每秒可加密50K以上的信息量。
- 3) 密钥的更改和密钥长度的选择均很方便。

由此可见，本文中提出的产生加密序列的新方法，在微机中用以加密信息是非常理想的。此外，作为一种新的序列产生的结构和方法，本文提供的方法也可以在其它需要产生序列的场合中加以应用，此处从略。

#### 参 考 文 献 ( 略 )

文献出处：《上海工业大学学报》1987年3期307—312页

# 扇区交错保密法

暨南大学计算机系87级 宋海路

在一个磁道内，不将扇区按 \$0~\$F 编号存放而是按任意顺序存放，这叫做扇区交错法，采用此方法，提高了DOS向磁盘存取文件时速度。

APPLE 的 DOS3.3操作系统中，扇区交错是由软件完成的，用DOS3.3格式化的磁盘上，16个扇区全按 \$0~\$F 顺序排列，并无交错，但在内存 \$BFB8~\$BFC7处存放着另一个转换表，此表将磁盘上的实际扇区编号一一对应到另一组假的扇区编号来供 DOS3.3使用（见下表）。

真实编号 0 1 2 3 4 5 6 7 8 9 A B C D E F

对应编号 0 D B 9 7 5 3 1 E C A 8 6 4 2 F

假设我们修改了 \$BFB8—\$BFC7转换表后，再用修改过的DOS3.3来 INIT 新盘，可以想象，如果我们在这个盘上存入程序，用标准的 DOS3.3是无法正常存取这些程序，因为它们之间扇区转换表不相同。

“扇区交错”程序保密法正是利用上述办法来制作出能保密程序的保密磁盘。

制作保密磁盘对硬件要求是：APPLE机的内存为48K，6号槽口接上磁盘机，磁盘机的编号为1。

制作方法：

①制出一个不同于DOS3.3的扇区转换表，你将16进制数字0至F间除去0剩余的15个数随意填入下表横线上（尽量不与标准DOS3.3的转换表一致）

00\_\_\_\_\_

假设你填入的是：

00 07 0D 0B 04 05 09 0E 01 06 0A 02 0C 08 0F 03

在纸上记下这组数字，然后根据APPLE机的显示ASCII码表，把每个数字加上C0后所得的值转化成字母，本例应为：

@ G M K D E I N A F J B L H O C

这就是以后使用保密盘时所必须给出的通行口令，有了这个口令，你才能往保密盘上存入或取出程序。记住这个密码。

②将标准DOS3.3引入内存，用CALL—151进入监控后，把刚才记在纸上的16个数输入从 \$BFB8开始的单元中。

\* BFB8, 00 07 0D 0B 04 05

09 0E 01 06 0A 02

0C 08 0F 03

\* (敲入CTRL-C)

I NEW

]

请你将一个空盘插入接在 6 号槽口的 1 号驱动器中。

] I N I T H E L L O

] C A L L - 1 5 1

• 6000: 01 60 01 00 00 00 11 60 00 40 00 00 01 00 00 60 01 00 01 EF D8

• 7000: A0 00 A9 60 4C D9 03 N 7 0 0 0 G

• 404E < 4 04 D. 4 05 B M

• 40E8: A2 01 20 1B FD 29 0F 9D 4C 08 E8 E0 11 D0 F3 A5 2B 4A 60

• 4007: 20 E8 08 N 4 0 4 A; 4C 69 BA N 60 0 C; 02 N 7 0 0 0 G

• 600C: 01 N 60 05; 09 N 7 0 0 0 G

• 40B9 < 40B8, 40 C6 M N 600C; 02 N 7000G

• 600C: 01 N 6005; 04 N 7 0 0 0 G

• 4069: 84 FF A0 10 B9 4C 08 99 B7 BF 88 D0 F7 A4 FF 6C FD 08 N 60  
0C; 02 N 7 0 0 0 G

大功告成！保密盘已制好。若你想使用保密盘，可以用 PR# 命令来引导保密盘上的 DOS，这时驱动器上的小红灯变亮，等约过了 10 秒，将你以前设定的密码打入，打完后无需按回车：@G M K D E I N A F J B L H O C L（注意：打入密码时，磁盘机上的小红灯仍是发亮的）。只有输入了正确密码，才能正常进入 DOS 状态，否则将发生预料不到的事。

这样制出的保密盘有一小缺陷，就是别人可以将保密盘中的 DOS 一个个扇区分别读出来后再与标准 DOS 盘中 DOS 所在扇区互相比较，分析出保密盘的扇区与标准 DOS 盘的扇区之间对应关系，进而解出保密盘采用的扇区转换表内容。要弥补这个缺陷较费工夫。这里我不准备详述而只是简单介绍一个弥补方法：

①先设定一个 IOB (Input/Output Control Block) 表及 DCT (Device Characteristics Table) 表，并将 IOB 表命令码设定为 02。然后调用 RWTS 将保密盘 0 磁道至 2 磁道所有扇区写入 0。

②这样处理的保密盘不能自举，要使用它，必须先在内存在装入正常的 DOS，然后把要使用的保密盘所采用的扇区转换表送入内存 \$BFB8~\$BFC7 中，方可使用各种 DOS 命令来向保密盘存入程序或从中取出程序。

文献出处：《电脑》1988 年 3 期 22—23 页

# 用数码形成密文的方法

陈 乐 庚

在计算机中，数字和符号都是以数码（如ASCII码）形式贮存的，因此要对一个文件的内容加密，就是要对这些ASCII码进行变换。下面就是一种简单的变换方法，从维吉尼亚加密法转变而来。

	0		明文						
	0	1	2	.....	7E	7F			
	0	0	1	2	.....	7E	7F		
	1	1	2	3	.....	7F	0		
密	2	2	3	4	.....	0	1		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
密	7E	7E	7F	0	.....	7C	7D		
7F	7F	0	1	.....	7D	7E			

上表中第一行为明文数码表，第一列为密钥数码表，加密过程如下：

1. 先约定密钥，例如我们以01, 2E, 6E, 6D为密钥。
2. 整齐地在一行上写明文，密钥整齐地写在明文的下面一行，而且密钥是循环地使用。
3. 查得交叉处的字即为密码数字。

例如：

明文	3E	3D	5F	41	45	6D	6E	6F	50
密钥	01	2E	6E	6D	01	2E	6E	6D	01
	↓	↓	↓	↓	↓	↓	↓	↓	↓
密码	3F	6B	4D	2E	46	1B	5C	5C	51

根据上述方法，可以找出一种数学方法如下：

加密： 密码 = 明文 + 密钥 ( 密码 \* ≤ 7F )

密码 = 明文 + 密钥 - 80H ( 密码 \* > 7F )

解密： 明文 = 密码 - 密钥 ( 密码 \* ≥ 密钥 )

明文 = 80H + 密码 - 密钥 ( 密码 \* < 密钥 )

利用这种数学方法，不需要查表，可以方便地计算出来，更适合于编制程序，如果要扩充上表的话，在公式中只要改变80H就行，例如80H改为100H，则明文和密钥的数码范围就是0—FFH。

注：密码 \* = 明文 + 密钥

文献出处：《计算机世界》月刊 1987年10期48页

# 加密程序的数据输入方法

合肥 汪明宽

我们知道，一个BASIC源程序存盘时如果用了P可选项：

```
SAVE ".....", P
```

则此程序只能装入和运行，而不能显示和修改，这称为加密（或保护）。如果此程序运行时需要输入数据，那么只能用回答 INPUT 提问的形式输入（即在源程序中必须事先设计好 INPUT 语句）。

然而，在很多场合下，我们更乐于用 DATA 语句来罗列数据，而在程序中用 READ 语句去读取，因为这样便于数据键入、检查和修改。可不可以在加密程序装入内存后，再加上若干 DATA 语句，或者先把 DATA 语句单独存盘成为一个程序，然后用归并的方法（用 MERGE 命令）让它与内存中的加密程序合并在一起呢？回答是否定的。因为加密程序不能被修改，而上面的办法实质上都对程序进行了修改。

如何在加密程序中用 DATA 语句输入数据呢？笔者采用了遇回作战的方法，使问题得到了较为圆满的解决。其基本思路是：

一、将数据用 DATA 语句列出，存在 ASCII 码文件。

二、在源程序的开头加上一段程序，这段程序对上述 DATA 语句组成的程序文件进行操作，将其改成为一个顺序数据文件。

三、源程序需要的数据从这个顺序数据文件中去读取，即用 INPUT# 语句取代 READ 语句。

四、源程序调试完毕后，用 P 可选项存盘进行保护。

下面我们用一个具体例子来说明操作过程和使用方法。

```
200 S=0:C=0
210 READ X
220 IF X=-9999 THEN 260 (-9999为结束标志)
230 PRINT X,
240 S=S+X:C=C+1
250 GOTO 210
260 PRINT
270 PRINT "P=" ; S/C
280 END
```

设欲求一批数据的平均值，则程序可以写成：

这个程序可用来求任一批数据的平均值。用户在使用这个程序时，可根据具体情况在其后面编上 DATA 语句，列出数据（以 -9999 为结束标志），再运行即可。

然而，如果这个程序用了 P 可选项存盘，那么就无法用上面的方法来合并 DATA 语句了。这时，我们可采用以下步骤：



一、将数据用DATA语句列出，注意不需要结束标志。

二、将这些DATA语句组成的程序以ASCII码存盘（设文件名取为）DATA.DAT）：

```
10 OPEN "I" , #1, "DATA.DAT"
20 OPEN "O" , #2, "TEMP.DAT"
30 WHILE NOT EOF ( 1 )
40 LINE INPUT #1, X$
50 K=INSTR ( 1, X$, "DATA" )
55 IF K=0 THEN PRINT"
    DATA XB$OK" :END
60 X$=MID$ ( X$, K+ 4 )
70 IF ASC ( X$ )=32 THEN
    X$=MID$ ( X$, 2 ) :GOTO 70
80 PRINT #2, X$; "." ;
90 WEND
100 CLOSE
110 KILL "DATA.DAT"
120 NAME "TEMP.DAT" AS
    "DATA.DAT"
    SAVE "DATA.DAT" ,
```

三、在程序（1）的前面加上如下的程序段：

这里，60句除去行号和“DATA”字符串，70句除去数据前面的空格。

这个程序段的功能，是将DATA语句组成的程序转变成一个顺序数据文件。

```
200 OPEN "I" , #1,
    "DATA.DAT"
210 S=0:C=0
220 WHILE NOT EOF ( 1 )
230 INPUT #1, X
240 PRINT X,
250 S=S+X:C=C+1
260 WEND
270 PRINT
280 PRINT "P=" ; S/C
290 CLOSE
300 END
```

四、将程序（1）本身修改为：

将这个程序（行号从10到300）以P可选项存盘，则成为一个可使用DATA语句的加密程序。用户在使用时，只要将DATA语句作为程序以ASCII码存盘，再调用加密程序运行即可。

文献出处：《软件报》1988年6月4日

# IBM PC 软件的加密

广州市计算机软件开发公司 何榕生

随着计算机的普及和广泛的应用，作为单位和个人开发的软件，对它进行加密以保护其权益，已成为当前人们研究的一个重要课题。下面笔者以在 IBMPC 机上研究的有关软件加密技术为例，谈一下软件加密的方法。

要研究软件加密技术，必须熟悉磁盘的格式，磁盘信息的分布，DOS操作系统和 ROM 监控程序的功能。对 IBMPC 上的软盘控制器 ( $\mu$ PD765) 的功能和寄存器编程方法也应有相当深的了解。

先介绍一下 IBMPC 软盘的标准格式。IBMPC 软盘的记录密度经常是用双面倍密度，也可用单面倍密度。一个软盘的每个面分为 0—39 磁道。而每个磁道又分成 8 或 9 个扇区，每个扇区为 512 字节/扇区长度。

IBMPC 规定 0 磁道为操作系统 DOS 所占用，其中 0 面 0 磁道 1 扇区是存放系统的引导程序。0 磁道两个面的其余扇区分别放文件分配表，文件目录表。而 1—39 道用户可以用来存放文件信息。

对于磁道上每个扇区的格式是这样的：每个扇区分成两大部分，即标识部分 ID 和数据部分 DATA。而 ID 和 DATA 之间都用间隙区 GAP 隔开。在 ID 和 DATA 区上都有 12 个字节的同步符号 SYNC。扇区的开头是 84 字节的 GAP，然后是同步号 SYNC 占 12 字节以及 1 字节的扇区标志 AMI。接下去才是 ID 部分和 DATA 部分。其格式如图一所示。

图一、IBMPC 软盘扇区格式

名称	GAP	SNYC	AMI	ID		CAP	SYNC	DATA			GAP
				ID	CRC			AM2	DATA	CRC	
字节	84	12	1	4	2	20	12	1	512	2	84
内容	4E	00	FE			4E	00	FB			4E

1. ID 部分为六个字节，其中磁道号  $C=00\sim 27H$ ，磁头号  $H=00\sim 01$ ，扇区号为  $R=01\sim 09$ ，数据长度  $N=00\sim 05$ 。IBMPC 取  $N=02$ ，表示扇区长度为  $128 \times 2^2$ ，即 512 字节/扇区。最后是 2 个字节的校验码 CRC。

2. DATA 部分由数据区标志 AM2，数据区和校验码 CRC 组成。当 AM2 为 FBH 时，表示该数据扇区为正常，如为 F8H，则该扇区作废。

扇区与扇区之间用 84 个字节的扇区间隙隔开。

磁盘的格式化和读写操作都调用 IBMPC 的监控程序中 ROMBIOS 的 13H 中断来实现的。其调用参数如下：

(1) 读盘

### 入口参数

AH=2 AL=要读的扇区数(1-9)  
CH=磁道号(0-39) CL=起始扇区号(1-9)  
DH=磁头号(0或1) DL=驱动器号(0-3)  
ES:BX=输入缓冲区首址

### 出口参数

AL=实际读出的扇区数  
CF=0, AH=0 读盘成功  
CF=1, AH≠0 读盘失败

### (2) 写盘。

AL=写入的扇区数, 其余与读盘相同。

### (3) 格式化

#### 入口参数

AH=5  
ES:BX=格式化信息起始地址  
格式化的信息应由C=磁道号, H=磁头号, R=扇区号, N=每扇区字节的指数组成。  
其中N参数可为00, 01, 02, 03  
分别代表每扇区128, 256, 512, 1024字节。  
CH, DH, DL寄存器规定参数与上述相同。

#### 出口参数

AL无意义, 其余与(1), (2)相同

我们格式化及读写磁盘时就是这个ROMBIOS中的13H中断完成的。

在调用ROM的13H中断功能时, 还需用到一系列磁盘参数, 这些参数放在ROMBIOS的位置是从F000:F0C7起11个单元中。而这些磁盘参数在系统启动时, 通过IBMBIO.COM文件把其入口地址放到1EH中断向量地址中。也就是0000:0078H~007BH单元中。

这11个磁盘参数作用如下:

字节1: 步进电机加速和去载时间。

DFH

字节2: 磁头加载时间和DMA操作。02

字节3: 操作后电机关闭时间。25H

字节4: 扇区长度指数。02

字节5: 磁道最大扇区号。09

字节6: 间隔长2AH

字节7: 数据长, 指数为0时才起作用。FFH

字节8: 格式化时的间隔宽。50H

字节9: 格式化用填充数据。F6H

字节10: 磁头稳定时间。25H

字节11: 主轴启动时间。04

只要对上面的磁盘参数进行修改, 格式化出来的磁道则为非IBM PC标准格式。这样我

们进行复制磁盘时，这个非标准的磁道则不能复制，从而达到加密的目的。由于格式化了一条非标准磁道，在读写这条磁道时，磁道参数也应与格式化时磁盘参数一致，否则不能对这条磁道进行读写操作。

了解以上有关IBMPC磁盘格式和INT 31H调用方法以及磁盘参数以后，我们就可以利用以上知识对软盘文件进行加密了。

首先分析一下著名的软件LOTUS 1—2—3，该软件的磁道格式还是按IBMPC的标准的。但在00道和01道的08扇区以后加了十个没有数据的扇区，只有ID部分。当执行123-EXE时，该程序先检查软盘00道和01道有没有这样的扇区，如有则把此扇区数据读入，进入123-EXE控制状态。否则返回操作系统。而这样特殊的扇区，DOS的拷贝，命令是无法拷贝的，这就达到对LOTUS1-2-3软件加密的目的了。

我们对自己研制的软盘文件也可以用类似的方法进行加密。比如，我们要对一个汉字打印驱动程序进行加密。其步骤如下：

1. 编写一个格式化某一磁道为非标准格式的程序。例如把某一磁道格式化为18个扇区，每个扇区长256字节，格式化时间间隔为20H，步进电机加速和去载时间为AHF。这就需要修改ROM中的磁盘参数区第1字节，第4字节和第8字节。然后才调用13H中断功能去格式化。为此应该重写一个11字节磁盘参数区，并把1EH中断向量地址内容改为指向这个参数区的首地址。

2. 编写一个写盘程序。也应按上述磁盘参数格式重写磁盘参数，并修改1EH中断指针指向这个参数区。然后调13H中断的写盘功能，把某些重要的数据写到由步骤1已格式化的那条非标准磁道上的某些扇区中。

3. 编写用户应用程序时，也应按上述格式设定磁盘参数，修改1EH中断指针。调用13H中断读盘功能把由步骤2写入的重要数据从非标准磁道的指定扇区读出。这些数据是用户应用程序执行时一定要用到，如果缺少这些数据程序无法执行，因此如果复盘时，复不到这条特殊格式磁道，则在这条磁道上的数据将在拷贝副本中丢失，副本的用户程序因找不到这些数据将无法执行，返回操作系统。

现以加密一个24点阵汉字打印驱动程序为例，说明这三个程序是如何编制的。

1. 格式化某磁道程序FORMAT24-ASM如图二所示。

```
stack segment para stack' stack'
    db 128 dup(0)
stack ends
; .....disk base date.....
code segment
disk ba db 0afh, 02, 25h, 01, 12h, 2ah
    db 0ffh, 20h, 0f6h, 25h, 04h
indicat db "39th# thrac format 1 $"
error db "error $"
old 1EHdw 2dup(0)
id buff db 39, 0, 1, 1, 39, 0, 2, 1, 39, 0, 3, 1
    db 39, 0, 4, 1, 39, 0, 5, 1, 39, 0, 6, 1
```

```

    db 39, 0, 7, 1, 39, 0, 8, 1, 39, 0, 9, 1
    db 39, 0, 10, 1, 39, 0, 11, 1, 39, 0, 12, 1
    db 39, 0, 13, 1, 39, 0, 14, 1, 39, 0, 15, 1
    db 39, 0, 16, 1, 39, 0, 17, 1, 39, 0, 18, 1
; .....disk CHRN data.....
    assume ce; code, ds;          id set; mov    ax, 0501h
    code es; code                 int    13h
format: push    ds                jc     rend
    push    cs                    lea    dx, indicat
    pop     ds                    mov    ah, 09
    push    cs                    int    21h
    pop     es                    G1;   qush   cs
; .....set 1EH INT and sayc old   pop    ds
    1EH INT.....                 ; .....reset old 1EH int.....
    mov    ax, 351eh             mov    dx, old 1EH
    int    21h                   mov    ds, old 1EH+ 2
    mov    old 1EH, bx          mov    ax, 251eh
    mov    bx, es               int    21h
    mov    old 1EH+ 2, bx      pop    ds
    lea    dx, disk ba         mov    ax, 4c00h
    mov    ax, 251eh          int    21h
    int    21h                 rend;  mov    dx, offset error
    .....format a track.....    mov    ah, 09
    lea    bx, id buff        int    21h
    push    cs                 jmp    G1
    pop     es                 code   ends
    mov    dx, 00              end    format
    mov    cx, 2701h

```

图二、格式化某磁道程序FORMAT24.ASM

这个程序对39磁道进行特殊格式化。磁道磁盘参数按disk-ba起的11个字节所规定的那样设置。这里把步进电机加速和去载时间改为AFH(字节1)。把磁道扇区长度改为01,即256字节/扇区(字节4)。把磁道扇区最大号数改为18(字节5)。把格式化用的间隔宽改为20H(字节8)。设计者还可以再改一些其它磁盘参数,使这条磁道与标准格式相差更大。在格式化39道前先修改1EH中断向量指针指向disk-ba。退出前再恢复系统设置的1EH中断指针。注意的是此磁盘应先用DOS的FORMAT.COM命令进行格式化后才用此命令对39道进行改写格式。

2. 写盘程序WRITE24.ASM如图三所示。

```

stack segment para stack 'stack'
    db 128 dup(0)

```

```

stack ends
code segment
disk ba db 0afh, 02, 25h, 01, 12h, 2ah
        'b 0ffh, 20h, 0f6h, 25h, 4
; .....disk base data.....
; .....save data buff.....
olib   db 03, "CLIB24", 20h, 20h, 20h, 20h, 20h
        db 25 dup( 0 )
        db 03, "CLIB241", 20h, 20h, 20h, 20h, 20h
        db 25 dup( 0 )
        db 03, "CLIB242", 20h, 20h,
        20h, 20h, 20h,
        db 25 dup( 0 )
        db 150 dup( 0 )
informa db "write complete! $"
old 1EH dw 2 dup( 0 )
indicat db "error $"
        assume cs; code, es; code,
            ds; code
write:  push ds
        push cs
        pop ds
        push cs
        pop es
; .....set 1EH int and save 1EH int
        .....
        mov ax, 351eh
        int 21h
        mov old 1EH, bx
        mov bx, es
        mov old 1EH+2, bx
        lea dx, disk ba
        int 21h
; .....write a sector.....
        lea bx, clib
        push cs
; .....reset old leh int.....
G1:     push cs
        pop ds
        mov dx, old 1EH
        mov ds, old 1EH+2
        mov ax, 251eh
        int 21h
        pop ds
        mov ax, 4c00h
        int 21h
rend:   mov dx, offset indicat
        mov ah, 09
        int 21h
        jmp G1
code ends
end write

```

图三、写盘程序WRITE24、ASM

这个程序用于把打印机驱动程序的汉字库文件的参数写入到39磁道1扇区。磁盘参数与FORMAT24.ASM一样。先修改1EH中断指针，然后调用13H中断写盘功能，把clib开始

的256个单元的数据写到该扇区，退出之前要恢复1EH中断指针。

3. 打印机驱动程序PRINT24.ASM如图四所示。

B) type print24, asm

```

stack segment para stack 'stack'
    db 256 dup(0)
stack ends
; .....dick base.....
code segment
disk ba db 0afh, 02, 25h, 01, 12h, 2ah
    db 0ffh, 20h, 0f6h, 25h, 4
cib24 db 258 dup(0)
error db 'load printer program error ! $"
tr24 db 0ah, 0ah, "CANNOT OPEN
    FILE C:CLIB24 $" ; .....read 39th a sector
str241 db 0ah,0dh,"CANNOT OPEN    lea    bx, cib24
    FILE C:CLIB241 $"           push   cs
str242 db 0ah 0dh "CANNOT OPEN    pop    es
    FILE C:CLIB242 $"           mov    dx, 00
old 1EH dw 2 dup(0)             mov    cx, 2701h
;                               mov    ax, 0201h
;                               int    13h
;                               jc     rend
;                               ; .....reset old 1EH INT.....
;                               mov    dx, old 1EH
start: push   ds                mov    ds, old 1EH+ 2
    push  cs                    mov    ax, 251eh
    pup   ds                     int    21h
    push  cs                    mov    dx, offset clib24
    pop   es                    mov    ah, 0fh
; .....set 1EH int and save old 1EH
    int.....                   int    21h
    mov   ax, 351eh             or     a1, a1
    int   21h                  jnc   err024
    mov   old 1EH, bx          mov    dx, offset clib24+37
    mov   bx, es              mov    ah, 0fh
    mov   old 1EH+2, bx       int    21h
    lea   dx, disk ba         or     al, a1
    mov   ax, 251eh          jnc   crro241
    int   21h                mov    dx, offset clib24+64
                                mov    ah, 0fh

```

```

int      21h
of       al, al
jnc      erro242
;
;
;
int      27h
rend     mov    dx, offset error
kk       mov    ah, 09
int      21h
mov      dx, old 1EH
mov      ds, old 1EH+2
mov      ax, 251eh

int      21h
mov      4c00h
int      21h
crro24:  mov    dx, offset str24
jmp      kk
crro241: mov    dx, offset str24
jmp      kk
corr242: mov    dx, offset str242
jmp      kk
;
;
;
code    edns
end

```

图四、打印机驱动程序PRINT24.ASM

这个程序是24点阵汉字打印的驱动程序中与加密技术有关的部分。由于打印时要打开的三个汉字库文件的参数都由WRITE24.ASM放到39磁道1扇区中。因而要执行此打印机驱动程序也应按上述的格式修改磁盘参数，修改1EH中断指针，然后调13H中断的读盘功能从规定的扇区取出汉字库的文件参数，使程序能打开打印的汉字库文件，执行下去。如果磁盘中没有这个扇区则显示：打印机驱动程序装入错误！并且返回DOS系统。

由于加密软件由三部分组成，而第一和第二个程序与用户无关，研制单位只把第三个用户程序存放到由第一、第二个程序作过特殊处理的软盘交给用户使用。用户如对此磁盘作拷贝，则不能把39磁道中的数据拷贝过去。这样拷贝的付本由于缺少39磁道中的汉字库文件的参数而不能执行。这就达到了对研制软件进行加密，防止拷贝的目的了。

当然，IBMPC的软盘加密技术是很多的，例如改变扇区标志FEH，使复制时找不到扇区开始地址。把作废数据标志F8H写进AM2，使复制时把这个扇区作为无用扇区放弃掉。

也可利用激光穿孔的方法，在软盘上随机打孔作为程序判别密码，在用户程序中增加一段读密码程序，这样经拷贝过的付本由于没有拷贝到穿孔之处的密码，执行时读不到密码而退出。

总之随着计算机应用水平的提高，为了对付破密技术，需作大量研制工作，以提高软件加密技术。本文介绍方法只是加密IBMPC软件的常用方法，作为抛砖引玉，不一定最可靠，有不当之处望赐教。



# APPLE- II 微机磁盘防止复制的一种方法

复旦大学 王宪初

本文介绍的“增加-变换磁道”的方法是防止复制、保证软件安全的一种十分有效的方法。

## 一、概 述

COPY文件一般有两种方法。一种是一个一个文件单独复制，比如用FID软件，或者用LOAD-SAVE命令。第二种是一个磁道一个磁道地复制，如CRAZY COPY，COPYA等软件。另外还有半磁道COPY法，如LOCKSMITH 5.0，NIBBLES AWAY I等软件。要保证软盘不能复制，就必须避免采用上述几种人们熟悉的方法。

## 二、原 理

“增加-变换磁道法”的原理很简单，最主要的思想就是将目录磁道从原来的\$11移到新增加的\$24中去，这样一来由于目录区找不到，就不可能用FID或LOAD-SAVE方法复制文件。而目前所有的复制软件都是从\$00道一直COPY到\$22道（也有可复到\$23道的，但是没有COPY到\$24的，因为APPLE II所有的资料都说得很明白，磁盘分区是从\$00到\$22共35个磁道）。所以放在\$24中的目录区不会被复制。为了防止在程序运行的过程中被Ctrl-C、Ctrl-Rcset中断，或者一工作程序运行完毕之后，使用者能打开目录区看到文件名，并能列出文件清单，或者能用CALL-151进入监控等，还需要在工作文件中加进控制语句。最后，特别要指出的是，我们发现在微机刚启动时就按下Ctrl-C，能够在工作程序（如Hello程序）运行之前停机，尽管此工作程序中已经加了控制语句。所以这个问题也是要以解决。顺便说一句，目前国内一些文献介绍的保密方法是可以利用Ctrl-C来破译的，方法就是一开始启动微机即按下Ctrl-C键，强迫中断后再去查目录，或者LIST等。

## 三、方 法

1) 用主盘将DOS·3.3引导好。

2) 换一空白磁盘放入驱动器，用CALL-151进入监控，然后将\$AC01，\$AE9F，\$B293，\$B397中的\$11换成\$24，再将\$BEFE中的\$23换成\$25。用Ctrl-C回到BASIC，再用命令“INIT SECRET”来初始化磁盘。这样所得的就是共有\$24道，且目录道在\$24的特殊盘片了。

3) 再次进入监控，将本文程序清单中的子程序从键盘输入机器，同时在\$88F7，\$88F8，\$88F9，\$88FA，\$88FB，\$88FC中分别输入一组你选定的密码的ASCII码。（比如你选定的密码是123456，那么你输入的ASCII码是\$B6，\$B5，\$B4，\$B3，\$B2，\$B1。请注意顺序相反）。同样在\$89F7，\$89F8，\$89F9，\$89FA，\$89FB，\$89FC中送进一样的ASCII码（上例中为\$B6~\$B1）回到BASIC。

4) 先用DELETE SECRET清除原先的SECRET, 然后用BSAVE SECRET, A \$ 8890, L \$ 180的命令将刚刚从键盘输入的程序存进磁盘, 这一程序的用处是防止在磁盘刚启动时按Ctrl-C来强迫中断。它的原理和使用方法下面再详述。

5) 使用DOS手册中的RWTS子程序将第0磁道第D扇区中的\$42位置中原来的数\$06改成\$34, 再写进\$0道\$D区。

6) 最后编一简单的BASIC程序, 以Hello为名存入磁盘。这个Hello程序最好建立如下:

```
10 POKE 1012, 1
20 ONERR GOTO 10
30 PRINT CHR$(4): "RUN H"
```

此程序中的第10句是控制Ctrl-Reset的, 20句是控制Ctrl-C的, 最后一句磁盘操作命令中的H是使用者真正有用的工作程序, 当然这个H中也应该有类似Hello中的第10、第20这两句以便控制Ctrl-Reset和Ctrl-C。而且H运行到最后应该自动进入另一程序或进入一个死循环以防此时被别人打开磁盘。

#### 四、子程序的使用

该程序(见程序清单)有三个作用: 1) 首先设了一个过滤网, 检查键盘输入的字符, 如果不是Ctrl-C就让它过去, 如果是Ctrl-C, 则进入本子程序处理。2) 本程序中预先放好了密码, 遇到Ctrl-C后, 立即停止运行等待用户连续输入6个字符, 如果输入的是正确的密码, 本程序即自行结束, 同时打开磁盘, 用户可根据需要进行磁盘操作。如果密码有误, 子程序则输出一个信息, 同时进入死循环, 此时非关机不能退出。3) 子程序还告诉微机第一个要执行的BASIC程序叫做Hello, 当然也可改名。\$88B0~88B4中就是Hello的ASCII码。你可将其改成所希望的名称。\$8890~\$8899将过滤网的入口地址建立起来, 这里是\$899A。

当微机启动后立即按下Ctrl-C键时, 微机在将DOS全部取入内存后即“嘟”一声, 然后屏幕光标闪烁, 没有提示符“)”你输入密码(比如123456)后, 微机又“嘟”一声, 提示符出现, 同时打出一个“SYNTAX ERROR”信息, 不用去管它, 此时磁盘已经被打开了。

#### 五、几点说明

1) 上述方法保密性能好, 主要是很少有人会想到目录道被放在\$24道上, 尤其是我们的保密盘中如果你用DOS 3.3的CATALOG去检查的话, 会看到Hello确实存在, 也可调出, 但其中真正有用的“H”却怎么也找不到。另外即使你想到是\$24的磁盘, 也无法用目前所有的COPY程序去复制。

2) 密码如果泄密, 可以任意修改, 方法是先将SECRET取进内存, 改掉\$88F7等位置的密码, 换上新的密码(不仅是数字, 也可是大、小写的字母)。再将新的SECRET存进磁盘即可。

3) 注意内存的使用。请勿超过\$8890, 以免冲掉SECRET。

4) 某些驱动器不能用\$24, 可以将前面第2步中的\$24改成\$23, 将\$25改成\$24, 也就

程序清單

8890 - A9 BA LDA # \$9A  
 8892 - 8D 53 AA STA \$AA53  
 8893 - A9 89 LDA # \$89  
 8897 - 8D 54 AA STA \$AA34  
 889A - 8D F4 03 STA \$03F4  
 889D - A2 00 LDX # \$00  
 889F - BD AB 88 LDA \$88AB,X  
 88A2 - F0 06 BEQ \$88AA  
 88A4 - 20 ED FD JSR \$FDED  
 88A7 - E8 INX  
 88A8 - D0 F5 BNE \$889F  
 88AA - 60 RTS  
 88AB - 84 D2 STY \$D2  
 88AD - D5 CE CMP \$CE, X  
 88AE - A0 C8 LDY # \$C8  
 88B1 - C5 CC CMP \$CC  
 88B3 - CC CF 8D CPY \$8DCF  
 88B6 - 00 BRK  
 8920 - 20 1B FD JSR \$FD1B  
 8923 - 20 4A FF JSR \$FF4A  
 8926 - A5 45 LDA \$45  
 8928 - C9 8D CMP # \$BD  
 892A - D0 08 BNE \$8934  
 892C - A9 70 LDA # \$70  
 892E - 85 38 STA \$38  
 8930 - 4C 73 89 JMP \$8973  
 8933 - EA NOP  
 8934 - AE 1F 89 LDX \$891F(a)  
 8937 - 30 E7 BMI \$8920  
 8939 - 9D F7 89 STA \$89F7,X  
 893C - CA DEX  
 893D - 8E 1F 89 STX \$891F  
 8940 - E8 INX  
 8941 - D0 DD BNE \$8920  
 8943 - A2 06 LDX # \$06  
 8945 - BD F6 89 LDA \$89F6,X  
 8948 - DD F6 88 CMP \$88F6,X  
 894B - D0 D3 BNE \$8920

894D - CA DEX  
 894E - D0 F5 BNE \$8945  
 8950 - 2C 51 C0 BIT \$C051  
 8953 - 2C 52 C0 BIT \$C052  
 8956 - 2C 54 C0 BIT \$C054  
 8959 - A9 FD LDA # \$FD  
 895B - 85 37 STA \$37  
 895D - 85 39 STA \$39  
 895F - A9 F0 LDA # \$F0  
 8961 - 85 36 STA \$36  
 8963 - A9 1B LDA # \$1B  
 8965 - 85 38 STA \$38  
 8967 - A9 8D LDA # \$8D  
 8969 - 85 45 STA \$45  
 896B - 20 3F FF JSR \$FF3F  
 896E - 60 RIS  
 8970 - 20 4A FF JSR \$FF4A  
 8973 - AE 10 89 LDX \$8910  
 8976 - BD 18 89 LDA \$8918, X  
 8979 - CE 10 89 DEC \$8910  
 897C - D0 14 BNE \$8992  
 897E - A2 06 LDX # \$06  
 8980 - 8E 10 89 STX \$8910  
 8983 - A0 04 LDY # \$04  
 8985 - BE 14 89 LDY \$8914, Y  
 8988 - 96 35 STX \$35, Y  
 898A - 88 DEY  
 898B - D0 F8 BNE \$8985  
 898D - A2 18 LDX # \$18  
 898F - 8E 1F 89 STX \$891F  
 8992 - 85 45 STA \$45  
 8994 - 20 3F FF JSR \$FF3F  
 8997 - 60 RTS  
 8998 - 00 BRK  
 8999 - 77 ウワワ  
 899A - C9 9A CMP # \$9A  
 899C - D0 06 BNE \$89A4  
 899E - A9 20 LDA # \$20  
 89A0 - 8D 99 89 STA \$8999

89A3 = 60	RTS	89C7 - A2 04	LDX # \$04
89A4 - C9 96	CMP # \$96	89C9 - BD 10 89	LDA \$8910, X
89A6 - D0 06	BNE \$89AE	89CC - 95 35	STA \$35, X
89A8 - A9 77	LDA # \$77	89CE - CA	DEX
89AA - 8D 99 89	STA \$8999	89CF - D0 F8	BNE \$89C9
89AD - 60	RTS	89D1 - A9 05	LDA # \$05
89AE - C9 C2	CMP # \$C2	89D3 - 8D 1F 89	STA \$891F
89B0 - D0 0D	BNE \$89BF	89D6 - A9 06	LDA # \$06
89B2 - 48	PHA	89D8 - 8D 10 89	STA \$8910
89B3 - AD 99 89	LDA \$8999	89DB - A9 87	LDA # \$87
89B6 - C9 20	CMP # \$20	89DD - 4C F0 FD	JMP \$FDF0
89B8 - F0 0C	BEQ \$89C6	89E0 - 20 E6 89	JSR \$89E6
89BA - C9 77	CMP # \$77	89E3 - 4C 00 E0	JMP \$E000
89BC - D0 F5	BNE \$89B3	89E6 - A2 04	LDX # \$04
89BE - 68	PLA	89E8 - BD 14 89	LDA \$8914, X
89BF - C9 87	CMP # \$87	89EB - 9D 52 AA	STA \$AA52, X
89C1 - F0 04	BEQ \$89C7	89EE - CA	DEX
89C3 - 4C F0 FD	JMP \$FDF0	89EF - D0 F7	BNE \$89E8
89C6 - 68	PLA	89F1 - 60	RTS

是说将目录磁道放在 \$23 中, 其余完全一样。

5) 此方法做成的盘片还有一个附带的好处: 安全性好, 别人不可能擦去你盘中的任何信息, 除非初始化或机械损伤。

以上过程及程序全部运行正确无误。

参考资料(略)

文献出处: 《微型计算机》1988年3期71—73页

## 也谈APPLE—II 机程序的加密

复旦大学 王宪初

**提要:** 本刊86年第3期发表了一篇文章, 介绍了一种简单易明的加密法, 本文是在其基础上, 进一步讨论了加密的方法, 提出了使用密码和改变DOS的加密方法, 这种方法的实施, 大大提高了程序的保密性能, 增加了破译的难度。

### 一、问题的提出

APPLE—II 机上要防正对程序列清单, 一是要对LOAD命令进行屏蔽, 二是要对CTRL-C和CTRL-Reset进行屏蔽。对LOAD命令屏蔽的最方便的方法是在SAVE程序时,

在程序名中加入控制字符,对CTRL-C和CTRL-Reset行屏蔽的方法是在程序开头之处加上ON ERR GOTO×××和POKE10120,这样在运行程序之后便不能用CTRL-C和CTRL-Reset中断程序。但是实际上只要输入一个小程序便可将目录中程序名称里的控制字符以闪烁态显出。这一小程序在任何一台APPLE-Ⅰ的随机资料中均可找到。另外在程序从磁盘取入内存时即磁盘驱动器在转时,就先按下CTRL-C键,程序会在执行之前就中断,然后很方便地可LIST程序清单。所以有必要进一步讨论对LOAD、CTRL-C的屏蔽问题。下面附的程序1就是前面提到的反控制字符的小程序。输入机器后即RUN,然后CATLOG后的目录便将全部程序的真名显示出。

#### 程序 1

```

10 DATA 201, 141, 240, 21, 201, 136
20 DATA 240, 17, 201, 128, 144, 13
30 DATA 201, 160, 176, 9, 72, 132
40 DATA 53, 56, 233, 64, 76, 249
50 DATA 253, 76, 240, 253
60 FOR I=768 TO 768+27
70 READ V:POKE I, V:NEXT I
80 POKE 54, 0:POKE 55, 3
90 CALL 1002

```

## 二、对LOAD的屏蔽

解决这一问题有两种方法:一是使用RWTS子程序读出目录区,在保密的程序名称后加上一个控制字符CTRL-U,或ESC,或CTRL-J、CTRL-H等的ASCII码。这样处理好的程序名即无法用前面提到的程序一解密,键盘上根本就打不出这几个作为程序名中的控制字符,自己运行前则可用RWTS将此控制字符去掉后再正常运行,当然这一过程可以编成三、四句话的一个BASIC程序,连同RWTS一起放在同一盘中,使用起来比较方便。第二种方法更为彻底,在初始化磁盘前,预先修改以下几个地址的内容:\$AC01、\$AE9F、\$AEFC、\$B293、\$B397。这五个地址中的数原为\$11,即目录磁道号,可将之改为\$4~\$22的任一数。一般改为\$4为好;同时将\$A8D2到\$A8D8中的7个ASCII码即CATLOG改成你自己知道的其它7个字母的ASCII码;再将\$A888到\$A88B的4个ASCII码即LOAD也换掉,最后INIT磁盘,这样做成的磁盘,别人用CATALOG和LOAD命令后将会出现语法错误的信息,只有你自己知道该用什么命令来代替CATALOG和LOAD。即使有人用自己的盘片引导好DOS后,再放进你的保密盘,用CATALOG也不能显示目录,因为磁道已经变了,显示的信息将是I/O ERROR。这样就彻底解决了LOAD的屏蔽问题。程序2就是RWTS子程序,使用的方法在APPLE-ⅠDOS手册中有具体说明。

#### 程序 2

```

0C00- A9 0C      LDA  # $0C
0C02- A9 0A      LDY  # $0A
0C04- 29 D9 03   JSR  $03D9

```

0C07 - 60	RTS
0C08 - 00	BRK
0C90 - 00	BRK
0C0A- 01 60	ORA (\$ 60, X)
0C0C- 01 00	ORA (\$ 00, X)
0C0E- 11 0F	ORA (\$ 0F), Y
0C10- 20 0C0C	JSR \$ 000C
0C13- 20 00 00	JSR \$ 0000
0C16- 01 00	ORA \$ 00, X
0C18- FE 60 01	INC \$ 0160, X
0C1B- 00	BRK
0C1C- 00	BRK
0C1D- BA	TSX
0C1E- FE 60 00	INC \$ 0060, X
0C21 - 01 EF	ORA (\$ EF, X)
0C23 - D8	CLD
0C24 - 22	???

### 三、对 CTRL—C 的屏蔽

这个问题很困难，我编制三段程序来解决这一问题，主要思想是对键盘输入的字符进行过滤，非CTRL—C则正常通过，是CTRL—C则跳进我为之编好的一段程序中进行处理，不使其有效地中断，而让其成为密码检查的前奏。就是说CTRL—C键按下之后，不管是程序运行中还是运行前，立即停下，等待输入密码，正确的话则正常中断，否则永远出错，除非重新启动机器。这就解决了CTRL—C键用ON ERR GOTO ×××不能完全屏蔽的问题。程序和说明如下：

#### 程序 3

8890-	A9 9A	LDA # \$9A
8892-	8D 53 AA	STA \$AA53
8895-	A9 89	LDA # \$89
8897-	8D 54 AA	STA \$AA54
889A-	8D F4 03	STA \$03F4
889D-	A2 00	LDX # \$00
889F-	DB AB 88	LDA \$88AB, X
88A2-	F3 06	BEQ \$88AA
88A4-	20 ED FD	JSR \$FDED
88A7-	E8	INX
88A 8-	D0 F5	BNE \$889F
88AA-	60	RTS

程序 4

```

88AB- 84 D2 D5 CE A0
88B 0- C8 C5 CC CC CF A0 A0 A0
88B 8- A0 A0 A0 A0 A0 A0 A0 A0
88C 0- A0 A0 A0 A0 A0 A0 A0 A0
88C 8- A0 A0 A0 A0 A0 8D 00
88D 0- 00 00 00 00 00 00 00 00
88D 8- 00 00 00 00 00 00 00 00
88E 0- 00 00 00 00 00 00 00 00
88E 8- 00 00 00 00 00 00 00 00
88F 0- 00 00 00 00 00 00 00 B6
88F8 - B5 B4 B3 B2 B1 B8 B7 B3
8900 - B4 B0 B0 B7 B4 B8 B2 B3
8908 - B5 B7 B6 B9 B4 D3 C4 C3
8910 - 06 A3 89 20 89 9A 89 1B
8918 - FD 8D D4 CE CF C3 98 FF
89F2- 00 8D F3 03 4C B6
89F8- B5 B4 B3 B2 B1 00 00 00
    
```

程序3是将字符处理的地址改到我编的程序入口处,即“过滤器”的入口处\$899A,然后  
再RUN HELLO程序。程序4是数据库,可以看出从\$88AB到\$88B5是RUN HELLO  
的命令,\$88F7到\$88FC以及\$89F7到\$89FC都是密码保存地址,这里的密码是123456,  
密码可以是任意六位字母、数字组成。程序5就是过滤器,在磁盘引导之后任何时候按键都会  
跳转这里来“过滤”。如果是CTRL-C则程序跳转到\$8920去执行;如果不是CTRL-C则  
一切照旧,正常工作。\$8920开始的程序6是一“密码核对器”,如果连续输入的六个键符  
合原先放好的密码,则程序正常中断,此时可以CATALOG,如果不符合,则永远显示出  
错信息。

程序 5

```

8899 - 77   ? ? ?
899A- C9 9A   CMP # $9A
899C- D0 06   BNE $89A4
899F- A9 20   LDA # $20
89A0- 8D 99 89 STA $8999
89A3- 60     RTS
89A4- C9 96   CMP # $96
89A6- D0 06   BNE $89AE
89A8- A9 77   LDA # $77
89AA- 8D 99 89 STA $8999
89AD- 60     RTS
89AE- C9 C2   CMP # $C2
89B0- D0 OD   BNE $89BF
89B2- 48     PHA
89B3- AD 99 89 LDA $8999
89B6- C9 20   CMP # $20
89B8- F0 OC   BEQ $89C6
89BC- C9 77   CMP # $77
89BE- 68     PLA
89BF- C9 87   CMP # $87
89C1- F0 04   BEQ $89C7
89C3- 4C F0 FD JMP $FDF0
89C6- 68     PLA
    
```

89C7-	A2	04		LDX	# \$04	8945-	BD	F6	89	LDA	\$89F6,X
89C9-	BD	10	89	LDA	\$8910, X	8948-	DD	F6	88	CMP	P88F6,X
89CC-	95	35		STA	\$35, X	894B-	D0	D3		BNE	\$8920
89CE-	CA			DEX		894D-	CA			DEX	
89CF-	D0	F8		BNE	\$89C9	894E-	DO	F5		BNE	\$8945
89D1-	A9	05		LDA	# \$05	8950-	2C	51	C0	BIT	\$C051
89D3-	8D	1F	89	STA	\$891F	8953-	2C	52	C0	BIT	\$C052
89D6-	A9	06		LAD	# \$06	8956-	2C	54	C0	BIT	\$C054
89D8-	8D	10	89	STA	\$8910	8959-	A9	FD		LDA	# \$FD
89DB-	A9	87		LAD	# \$87	895B-	85	37		STA	\$37
89DD-	4C	F0	FD	JMP	\$FDF0	895D-	85	39		STA	\$39
89EO-	20	E6	89	JSR	\$89E6	895F-	A9	F0		LDA	# \$F0
89E3-	4C	00	E0	JMP	\$E000	8961-	85	36		STA	\$36
89E6-	A2	04		LDX	# \$04	8963-	A9	1B		LDA	# \$1B
89E8-	BD	14	89	LDA	\$8914,X	8965-	85	38		STA	\$38
89EB-	9D	52	AA	STA	\$AA52,X	8967-	A9	8D		LDA	# \$8D
89EE-	CA			DEX		8969-	85	45		STA	\$45
89EF-	D0	F7		BNE	\$89E3	896B-	20	3F	FF	JSR	\$FF3F
89F1-	60			RTS		896E-	60			RTS	
				程序 6		896F-	00			BRK	
8920-	20	1B	FD	JSR	\$FD1B	8970-	20	4A	FF	JSR	\$FF4A
8923-	20	4A	FF	JSR	\$FF4 A	8973-	AE	10	89	LDX	\$8910
8926-	A5	45		LDA	\$45	8976-	BD	18	89	LDA	\$8918,X
8928-	C9	8D		CMP	# \$8D	8979-	CE	10	89	DEC	\$8910
892A-	D0	08		BNE	\$8934	897C-	D0	14		BNE	\$8992
892C-	A9	70		LDA	# \$70	897E-	A2	06		LDX	# \$06
892E-	85	38		STA	\$38	8980-	8E	10	89	STX	\$8910
8930-	4C	73	89	JMP	\$8973	8983-	A0	04		LDY	# \$04
8933-	EA			NOP		8985-	BE	14	89	LDX	\$8914,Y
8934-	AE	1F	89	LDX	\$891F	8988-	96	35		STX	\$35, Y
8937-	30	E7		BMI	\$8920	898A-	88			DEY	
8939-	9D	F7	89	STA	\$89F7,X	898B-	D0	F8		BNE	\$8985
893C-	CA			DEX		898D-	A2	18		LDX	# \$18
893D-	8E	1F	89	STX	\$891F	898F-	8E	1F	89	STX	\$89F
8910-	E8			INX		8992-	85	45		STA	\$45
8941-	D0	DD		BNE	\$8920	8994-	20	3F	FF	JSR	\$FF3F
8943-	A2	06		LDX	# \$06	8997-	60			RTS	

使用这组程序(程序3到程序6)的具体方法是:首先键入程序3到程序6,注意地址不能有误,同时将密码表的六个数字改成只有自己知道的内容(ASCII码)。然后存入磁



盘; A\$是\$8890, L\$是180。名称自定。第二步是修改磁盘中1道9区的内容,使用RWTS子程序读出后,将\$2075开始的程序名称改掉,原来是HELLO,改成你刚刚存进的名称,再用RWTS写进去即可。这样做成的磁盘在引导完DOS之后立即成为不能用CTRL-C中断的保密盘,这就解决了CTRL-C的屏蔽问题。

最后还要说明两点:一用前面对LOAD和CTRL-C屏蔽的方法结合起来做成的磁盘中放进的需保密的程序体中要注意不让其有结束的地方,防止程序执行完毕后即停下来。必须将结束的地方转到执行另一程序,比方Hello等。当然CTRL-Rest的屏蔽也要在Hello中预先解决,方法就是POEK1012, 0,二是保密和解密是一对发展中的矛盾,世上决无绝对的保密方法,只有破译难易的程度不同。上述方法就是一种比较难破译的方法。

文献出处:《电脑与微电子技术》1987年2期20—23页

## 也谈APPLQ BASIC程序的加密

广西柳州师专 肖孟虎

本刊1986年三期武进军、李应刚介绍的一种程序加密方法,容易破解。例如,先启动另一块系统盘或文件盘,即可恢复LIST命令,因为这时没执行POKE 214, 255语句。又例如,运行下列程序,再用CATALOG命令,便可得到含调动控制字符的文件目录。

```
10 FOR I=768 TO795:READ N: POKE I, N: NEXT
20 POKE54, 0: POKE55, 3:CALL 1002
30 DAT A201, 141, 240, 21, 201, 136, 240:17, 201, 128, 144, 13, 201, 160,
176, 9, 72, 132, 53, 56, 233, 64, 76, 249, 253, 76, 240, 253
```

若要恢复正常显示方式,键入PR# 0即可。

显示执行文件就更容易了。只需运行以下程序(假设文件名为F1)

```
10 D$=CHR$(4):PRINTD$; "MON I"
20 PRINT D$; "EXEC F1"。
30 PRINTD$; "NOMON I":END
```

以下介绍一种保密度更高一些的加密方法,它不仅能保密程序,而且一般的COPY程序不能拷它,该方法的指导思想是构造一个非标准的文件盘,使标准的DOS与COPY不能对其进行操作,步骤如下:

1. 启动标准的DOS3.3,并进入监控
2. 键入下列内容,其中“\*”为电脑自动给出,“↓”表示回车键。

* AC01:13 ↓	* AE9E:13
* AEDC:13 ↓	* B293:13 ↓
* B3BC:13 ↓	* B4BC:13
* B5FA:13 ↓	* AEC5:4C ↓

- AEC9:50 ↓
- AEB5:90 ↓
- B3EF:23 ↓
- BEFE:23 ↓

3. 键入CTRL-C ↓
4. 把系统盘换成文件盘，然后键入NEW命令。
5. 键入HELLO程序，其中第一句为POE E214, 255
6. 键入命令INIT HELLO ↓便完成了。

文献出处：《电脑与微电子技术》1987年2期

## 简单可靠的APPLESOFT程序保密法

湖北 瞿波

在DOS3.3, 运行下列程序

该程序的作用是将机器内存有关目录磁道的单元的数据加以改变,使之不再是关于17(\$11)磁道的数据,而是由20语句重新设置的目录磁道T的数据。(在3~16、18~34范围内任选)。

随后键入一个HELLO程序,由HELLO程序建立一个菜单,并在HELLO程序中加入如下语句:

```
1 POKE 1011, 0
2 ONERR GOTO 1000
:
1000 PRINT CHR$(4); "PR# 6"
```

POKE1011, 0的作用是屏蔽CTRL-RESET,使按CTRL-RESET后,磁盘自举。

2语句和1000语句的作用是在按了CTRL-C后或者出了错误后,重新启动磁盘。

再键入:

```
INIT HELLO ✓
```

这样就得到了一张目录磁道为T的磁盘,该磁盘与标准DOS3.3盘除了目录磁道的位置不同外,其他完全一样。

然后键入应用程序,存入磁盘,使应用程序和菜单建成一个转动锁——由菜单选择应用程序,程序运行完后又返回菜单。注意,在应用程序中也要加入:

```
2 ONERR GOTO 1000
:
1000 PRINT CHR$(4); "PR# 6"
```

如此,一张加了密的盘便制作完毕。该加密盘启动后,程序运行自如,可就是无法列出程序清单,用其它DOS3.3盘开机引导后,也无法看到该加密盘的目录。

```
10 HOME
```

```

20 INPUT "CATALOG TRACKE ( 3—16 OR 18—34 ):" ; T
30 POKE 44033, T:POKE 44703, T:POKE 44741, 4 * T:POKE 44745, 4 *
    T+4 :POKE 44764, T:POKE 45715, T:POKE 46012, T: POKE 46268,
    T:POKE 46536, T
40 NEW

```

文献出处:《软件报》1988年6月4日

## APPLESOFT程序的加密与解密

沈阳 修广荣

笔者编制了一个小程序(见程序)运行后可在磁盘上产生一个名为UNBINAR Y的T型文件

解密时,插入含有T型文件UNBIARY的磁盘,打入: EXEC UNBINAR Y/屏幕出现提示: FILENAME? 时,再插入用户盘,键入欲解密之文件名,即刻列出该程序清单。利用SAVE命令存入磁盘,即可得到一个解密的APPLESOFT程序。并且,如欲再剖析其它同类加密程序时,也不必重新执行此T型文件,而只须直接打入: BLOAD (加密文件名) /然后LIST即可。

程序所产生的T型文件,均可在汉卡支持下的中文APPLESOFT中执行,进行程序加密和解密。

```

10 D$=CHR$(4):PRINT D$ "OPEN UNBINAR Y":PRINT D$
    "WRITE UNBINAR Y"
20 PRINT "HOME"
30 PRINT "POKE 115, 0:POKE 47699, C:INPUT" CHR$(34)"FILE
    NAME P" CHR$(24) "; IN$:POKE 118, 255;POKE 43699, 212"
40 PRINT "PRINT CHR$(4)" CHR(34)" BLOAD "CHR$(34)" IN$"
50 PRINT "CALL 151"
60 PRINT "81A:60 N 300G N E0036"
70 PRINT "LIST"
80 PRINT DE CLOSE
HOME
POKE 118, 0:POKE 43699, 0:IHPUT "FILE NAME ?" ; N$:POKE 118,
    255:POKE 43699, 212
PRINT CHR$(4) "BLOAD" ; N$
CALL 151
B1A; 60 N 300G N E0036
LIST

```

文献出处:《软件报》1988年6月4日

# APPLE II 应用软件的加密方法

梁炯基 中山眼科中心

在设计应用软件时，设计者往往希望自己所设计的软件具有一定的保密功能，使用户或其他人不能随意列出程序，为此，本文将讨论APPLE II应用软件的加密方法。

**一、列程序清单的方法** 通常情况下，用户可采用下列方法之一来列出程序清单。

1. 在DOS状态下，使用LOAD命令，将程序调入内存，再用LIST命令列出程序清单。

2. 对于存在文本中的程序，先用MON命令，再用EXEC命令运行文本程序。利用MON功能列出程序清单。

上述两种方法，程序文件名是向用户公开的，也可用CATALOG命令得到文件名。

3. 程序运行结束后，利用程序仍留在内存的特点，用LIST命令列出程序清单。

4. 程序运行中用CTRL-C命令将运行中断，然后用LIST命令列出程序清单。

5. 有意制造不能容许的错误，如令除数为零或负数开平方等，使系统产生出错中断，然后用LIST命令列出程序清单。

6. 程序运行中用RESET命令强迫系统复位，执行暖启动，然后用LIST命令列出程序清单。

**二、软件的加密方法** 要禁止用户列出清单，就必须针对上述的各种列程序清单的手段，在软件上采取相应的加密措施，具体如下：

1. 程序是以A文件的形式存在磁盘上，为了防止用户使用LOAD命令，我们可以将文件名进行加密处理。所谓文件名加密，就是在文件名的适当地方加入隐字（即按住CTRL键后打入字母）。例如：有一程序存盘，我们键入SAVE PUR CTRL-T CTRL-U CTRL-F，以加密的文件名存盘后用CATALOG命令列文件目录，屏幕上所看到的文件名为PUR，若键入LOAD PUR，系统将显示FILE NOT FOUND，只有使用和输入时相同的文件名，才能调用该程序。

2. 用户使用RESET命令时，系统转入监控复位程序。系统完成初始化后将校验\$03F2 \$03F3 \$03F4三个单元的值，以决定转移的入口地址，从监控程序可知（有关部分的程序如下），只要适当修改这三个单元的内容，就可以使系统复位时转移到我们编制的机器语言程序，完成预想的工作。

```
FA82- 20 3A FF JSR $FF3A
FA85- AD F3 03 LDA $03F3
FA88- 49 A5     EOR # $A5
FA8A- CD F4 03 CMP $03F4
FA8D- DO 17     BNE $FAA6
FA8F- AD F2 03 LDA $03F2
FA92- DO 0F     BNE $FAA3
```

```

FA94 - A9 E0      LDA # $E0
FA96 - CD F3 03   CMP $03F3
FA99 - DO 08     BNE $FAA3
FA9B - A0 03     LOY # $03
FA9D - 8C F2 03  STY $03F2
FAA0 - 4C 00 E0   JMP $E000
FAA3 - 6C F2 03  JMP ($03F2)

```

通常，磁盘自举启动DOS后，这三个单元的内容分别为BF 9D 38，\$9DBF为DOS的暖启动入口地址，\$9D84为冷启动入口地址。设CECR程序入口地址为\$0300，则预先将\$03F2等三个单元的内容分别为00 03 A6，复位时系统初始化将会自动转向以\$0300为入口的ECR程序。为了实现这一转移，可在程序一开始用POKE语句将指针及CECR程序写入相应的内存。其程序如下：

```

1 POKE 1010, 0; POKE 1011, 3; POKE 1012, 166
2 FOR I=768 TO 768+17; READ A; POKE I, A; NEXT
3 DATA 169, 191, 141, 242, 3, 169, 157, 141, 243, 3, 169, 56, 141, 244,
   3, 76, 132, 157

```

CECR程序如下：

```

0300 - A9 BF      LDA # $BF
0302 - 8D F2 03  STA $03F2
0305 - A9 9D     LDA # $9D
0307 - 8D F3 03  STA $03F3
030A - A9 38     LDA # $38
030C - 8D F4 03  STA $03F4
030F - 4C 84 9D  JMP $9D84

```

3. 程序运行结束时，为了清除内存中的程序，最简单的方法是在END语句的 前面加入NEM语句，这样，程序结束时就可将程序从内存清掉。但如采用上述的方法修改了\$03F3三个单元的内容时，为了使程序运行结束后系统复原，可将END语句改为CALL 768，这样程序结束时将自动清除内存的程序，并使系统复原。

4. 在APLPESOFRT里，当使用ONERR GOTO语句后，若按CTRL-C键，系统将作为错误处理，错误代码为255。因此，只要在错误处理程序里采取加密措施，就可以防止用户用CTRL-C键或制造错误产生中断。错误处理程序如下：

```

0 ONERR 6OTO 63992
1 POKE 1010, 0; POKE 1011, 3; POKE 1012, 166
2 FOR I=768 TO 768+17; READ A; POKE I, A; NEXT
3 DATA 169, 191, 141, 242, 3, 169, 157, 141, 243, 3, 169, 56, 141, 244,
   3, 76, 132, 157
63990 6OTO 63999
63992 IF PEEK(222) = 254 THEN RESUME
63994 PRINT CHR$(7); "ERROE MODE;" PEEK(222)

```

```
63996 PRINT "LINE NUMBER; " , 256 * PEEK ( 219 ) + PEEK ( 218 )
63999 CALL 768
```

上述程序中，63990行是为了防止有些程序未使用END语词而加入。63992行是使用户在输入语句中错误按键时能自动返回程序而不致于清除程序。输入错误的代码为254。

5. 为了进一步加强程序的保密性能，可将程序以文本的形式存盘，使用EXEC命令使计算机在文本控制下工作。这样程序是以合法的T文件出现在磁盘目录上，无法用LOAD命令调出。为了防止用户用MON命令来查阅程序，可在文本的开头写入以下命令：

```
PRINT CHR $ ( 4 ) ; "NOMON O, I, C"
POKE 1010, 0; POKE 1011, 3; POKE 1012, 166
POKE 768, 169; POKE 769, 191; POKE 770, 141
POKE 771, 242; POKE 772, 3; POKE 773, 169
POKE 774, 157; POKE 775, 141; POKE 776, 243
POKE 777, 3; POKE 778, 169; POKE 779, 56
POKE 780, 141; POKE 781, 242; POKE 782, 3
POKE 783, 76; POKE 784, 132; POKE 785, 157
```

6. 上述的几种加密措施，在执行RUN或EXEC命令时，仍需输入文件名，这样，使用加密的文件名作不大。为此，可借助于磁盘中的引导程序，使我们设计的程序自运行。

当磁盘只存一个软件时，可在磁盘初始化时，指定一个加了密的文件名作为该盘的引导程序，然后将调试好的程序自动存入此盘。以后该盘的引导程序就是我们设计的程序。另一种办法是将我们调试好的程序以一加了密的文件名存入磁盘，然后运行系统中的MASTER CREATE程序，将该磁盘转变成主磁盘，同时将该文件名作为引导程序名。这样，磁盘自举时就能自动引导而转入程序运行。

当盘中有几个应用软件时，可编一段程序作为引导程序。通过运行该程序，使用户可以选择所需的软件，然后自动运行。假设磁盘中有五个程序，文件名分别为A CTRL-P T1、B CTRL-Q T2、A CTRL-K R1、B CTRL-U G2、H CTRL-S L3、引导程序名为HEL CTRL-H LO引导程序如下：

```
10 F $ ( 1 ) = "A CTRL-P T1"
20 F $ ( 2 ) = "B CTRL-Q T2"
30 F $ ( 3 ) = "A CTRL-K R1"
40 F $ ( 4 ) = "B CTRL-U 62"
50 F $ ( 5 ) = "H CTRL-S L3"
60 FOR I= 1 TO 5
70 PRINT I; " , F $ ( I ) ; PRINT
80 NEXT
90 INPUT "TYPE THE NUMBER UP" ; N
100 IF N < 1 OR N > 5 THEN 90
110 PRINT CHR $ ( 4 ) ; "RUN" ; F $ ( N )
```

三、应用举例 下面介绍一个名为CECRET的程序。运行该程序，可将普通的没有保密功能的程序自动生成为具有保密功能的新程序，并以新的文件名存入磁盘。CECRET对

欲加密的程序有如下的限制:

1. 在程序中应未使用ONERR GOTO语句。若已使用了语句, 则需对错误讯息处理子程序作相应的修改, 使未作处理的错误讯息转到63991语句行。例如下列错误处理子程序中, 对INPUT错误, 对DOS操作的部分错误作处理后返回。而对其它类型的错误, 则采取清除ONERR GOTO标志并返回, 使其再次发生错误而产生出错中断。

```
10 ONERR GOTO 6000
6000 A=PEEK(222)
6010 IF A=254 THEN PRINT "REENTER"; RESUME
6020 IF A<4 OR A>10 THEN POKE 216, 0; RESUME
6030 PRINT "ERROR MODE: "; A; PRINT "CHECK DISK! "; CHR$(7)
6040 INPUT "READY TYPE 'RETURN' "; PQ$; RESUME
```

为了适当CECRET程序的要求, 需对6020行进行修改, 修改后的程序如下:

```
10 ONERR GOTO 6000
6000 A=PEEK(222)
6010 IF A=254 THEN PRINT "REENTER"; RESUME
6020 IF A<4 OR A>10 THEN 639 91
6030 PRINT "ERROR MODE: "; A; PRINT "CHECK DISK! "; CHR$(7)
6040 INPUT "READY TYPE 'RETURN' "; PQ$; RESUME
```

2. 将程序中所有的END语句都改成GOTO63998或CALL768。

3. 程序中所使用的行号须在2—63989的范围以内。

4. 程序需以A文件的形式存在磁盘上。

CECRET程序的简要说明:

1000—1050行进行人机对话, 用户可在键盘上回答程序的提问。1060—1220行用来在磁盘上建立一个名为F\*-2\$的文本文件(该文件名和磁盘上的其它文件同名的可能性极小)。该文本只是一个过渡性文件, CECRET在文本里写入一系列语句, 通过EXEC命令运行该文本, 计算机将在文本的控制下, 进行如下的工作:

1. 将欲加密的程序从磁盘调入内存。

2. 给程序增加0行和1行语句。在这两个语句, 设置了RESET指针和出错转移语句, 在内存预先写入以\$300(768)为入口的机器语言程序, 其内容如下:

```
0300 - A9 BF      LDA    # $BF
0302 - 8D F2 03  STA    $03F2
0305 - A9 9D      LDA    # $9D
0307 - 8D F3 03  STA    $03F3
030A- A9 38      LDA    # $38
030C- 8D F4 03  STA    $03F4
030F- A9 00      LDA    # $00
0311 - A2 10      LDX    # $10
```

```

0313 - A8          TAY
0314 - 91 67      STA  ($67), Y
0316 - C8          INY
0317 - D0 F8      BNE  $0314
0319 - E6 68      INC  $68
031B- CA          DEX
031C- 10 F6      BPL  $0314
031E- 4C B4 9D   JMP  $9D84

```

该程序和前述的程序相比，新增加了\$30—\$31D，目的将程序全部清为零。为了在两行语句写完上述命令，1090—1110行在输入时要特别注意，不要随意颠倒位置及增加空格，否则，有些内容将会被系统自动删掉。

3. 给程序增加63990—63999语句行，以便对错误信息进行处理。

4. 将已具有保密功能的程序以新的文件名存入磁盘，该文件名为用户根据程序的提示输入的，建议使用加密文件名，并登记存档。

5. 将过渡文件F\* \*-2\$从磁盘删除。用户如认为必要，也可增加命令，将原有程序删除。

CECRET程序清单如下：

```

1000 HOME
1010 INPUT "FILE NAME FOR KEEP IN CECRECY" ; F1$ ; PRINT
1020 INPUT "CECRET FILE NAME" F2$ ; PRINT
1030 PRINT "THE LINE NUMBERS ARE BEWTEEN" ; INPUT " 2 TO
      63989? (Y/N)" ; P$
1040 IF P$ = "N" THEN PRINT "MODIFY LIEN NUMBER!" ; END
1050 IF P$ <> "Y" THEN 1030
1060 PRINT CHR$(4) ; "OPEN F* *-2"
1070 PRINT CHR$(4) ; "WRITE F* *-2$"
1080 PRINT "PRINT CHR$(4) ; " ; CHR$(34) ; "LOAD" ; F1$ ;
      CHR$(34)
1090 PRINT " 0 POKE 1010, 0 ; POKE 1011, 3 ; POKE 1012, 166 ; POKE
      768, 169 ; POKE 769, 191 ; POKE 770, 141 ; POKE 771, 242 ; POKE
      772, 3 ; POKE 773, 169 ; POKE 784, 0 ; POKE 775, 141 ; POKE 786,
      16 ; POKE 777, 3 ; POKE 796, 16 ; POKE 779, 56 ; POKE 798, 76 ; POKE
      781, 244 ; POKE 782, 3 ; " ;
1100 PRINT "POKE 783, 169"
1110 PRINT " 1 POKE 774, 157 ; POKE 785, 162 ; POKE 776, 243 ; POKE
      787, 168 ; POKE 788, 145 ; POKE 789, 103 ; POKE 790, 200 ; POKE 791,
      208 ; POKE 792, 251 ; POKE 793, 230 ; POKE 794, 104 ; POKE 795, 202 ;
      POKE 778, 169 ; POKE 797, 246 ; POKE 780, 141 ; POKE 799, 132 ;
      POKE 800, 157 ; " ;

```



```

1120 PRINT "ONERR GOTO 63991"
1130 PRINT "63990 GOTO 63998"
1140 PRINT "63991 IF PEEK ( 222 ) =254 TBEN RESUME"
1150 PRINT "63992 IF PEEK ( 222 ) =255 THEN 63998"
1160 PRINT "63994 PRINT; PRINT"; CHR $ ( 34 ); "ERROR MODE; ";
      CHR $ ( 34 ); "PEEK ( 222 ); PRINT"
1170 PRINT "63996 PRINT; PRINT"; CHR $ ( 34 ); "LINE NUMBER; ";
      CHR $ ( 34 ); "256 * PEEK ( 219 ) + PEEK ( 218 ); PRINT CHR $
      ( 7 )"
1180 PRINT "63998 PRINT"; CHR $ ( 34 ); "GOOD BY! "; CHR $ ( 34 )
1190 PRINT "63999 CALL 768"
1200 PRINT "PRINT CHR $ ( 4 ); "; CHR $ ( 34 ); "SAVE"; F2 $;
      CHR $ ( 34 )
1210 PRINT "PRINT CHR $ ( 4 ); "; CHR $ ( 34 ); "DELETE F • • -2 $";
      CHR $ ( 34 )
1220 PRINT CHR $ ( 4 ); "CLOSE"
1230 PRINT CHR $ ( 4 ); "EXEC F • • -2 $"
1240 END

```

文献出处：《微型电脑》1987年5期48—51页

## COMX—PC1机上程序加密

广西 罗家仁

在COMX-PC1机上键入几个简单的命令，能对你的程序起到加密的作用。

一、一个可运行的BASIC程序，只要键入 POKE ( 17420, 225 ); POKE ( 17421, 255 ) ✓，则该程序只能运行，而不能列表。将程序录进磁带，下次调用依然具有加密功能。

二、如果只键入 POKE ( 17025, 255 ) ✓。则程序既不能列出也不能运行。录进磁带后，依然保持这一功能。

三、假定程序第0行是空着不用的，这时，键入：

```

0 REM JM ✓
POKE ( 17423, 0 ) ✓

```

这样，原程序也无法列表和运行。再录进磁带，同样具有加密之功能。

解密方法：

一、键入 POKE ( 17420, 0 ) ✓

POKE ( 27421,0 ) ✓即可。

二、键入POKE ( 17025,68 ) ✓即可。

三、键入POKE ( 17423,128 ) ✓即可。

#### 四、加密原理

原来，程序用户区是从十六进制的4400开始，程序存放都从440C开始存放，当我们用POKE ( 17420,255 ) 和POKE ( 17421,255 ) 这两个命令时，改变了十六进制的440C和440D单元的内容，程序无法列表，但仍可运行。

进一步加密：

结合改变用户空间入口地址指令DEFUS，可以加一把或多把更保险的“锁”，程序转录到磁带上和从磁带把程序装入内存，都不会影响这把“锁”的功能。

文献出处：《软件报》1988年3月12日

## R1机BASIC程序的加密

王 用 明

设有程序

10 A=8

20 A\$="ABC"

30 PRINT A\*\*\*3

40 PRINT A\$

加密方法之一：

键盘输入

( 1 ) 1 PEM1 ✓

( 2 ) POKE 17307,118 ✓

这时按列表命令：LIST ✓，屏幕只显示：

1 REM

如果要看原来的程序，可按“LIST2 ✓”，原来的程序就会显示出来。

加密后运行的结果是一样的：

512

ABC

解密方法：

POKE17307,0 ✓。

加密方法之二：

键盘输入：

( 1 ) 1 REM1 ✓

( 2 ) POKE 17307, 118✓

( 3 ) POKE 173002, 39✓

此时“LIST✓”，屏幕显示：

9985 REM

此时“LIST2✓”，还是一样。就这样，把9985句以前的程序都被封锁了。当然运行的结果终始如一。

当然，解铃还需系铃人，解密方法

键盘输入：

OPKE 17302, 0✓

POKE 17307, 0✓

LIST✓，全部的程序又会出现，当然运行的结果还是终始如一。

方法二保密后，虽然程序不能被窥视，但是程序照样能用“SAVE”命令转录到磁带上。

“LOAD”后照样运行，并得出相同的结果，能否找到一种加密的方法，即使程序是被复制到磁带上，其复制品也不能如期运行呢？回答是肯定的。下一期将介绍这种方法。

文献出处：《电子与电脑》1987年9期32页

## 浅谈BASIC程序的加密与解密

张 氏

一个较好的程序。往往要花费程序设计者大量的脑力与精力。为了不让别人窃看你的程序，就需要对程序进行加密。加密步骤如下：

### 1. 程序在内存中的加密

当BASIC程序在内存中经调试好后，在存盘前，需要对其进行加密。加密方法有许多种，这里仅介绍三种常见的：

( 1 ) 直接键入“POKE214,128”，把128存入内存214单元，这样LIST命令就失效了。若要恢复的话，必须运行一个含有POKE214,0的程序。注意，不能直接把POKE214, 0作为键盘命令。

( 2 ) 改变\$ 801单元以后的内容。我们知道，BASIC程序在苹果机中是\$ 801单元开始存放的，保留字占一个字节，数字要拆开来算字节长度。现有一程序：

```
10 FOR I = 1 TO 100
```

现在我们来算这一行共占几个字节（行号除外）。由于FOR与TO是保留字，它们各占一个字节，100拆开算占三个字节，因此该行程序共占8个字节。其余类推。在程序调试结束后，把某单元的数值修改一番，可达到保密效果。例如，我们只让别人列出第一行程序，这时要先算出第一行程序所占的字节数（N），然后把N加上\$ 806，得\$S，再把\$S单元值置1。上例的N值为8，则POKE2062, 0（因为\$ 806 + 8 = \$ 80E，\$ 80E = 8 × 265 + 0

$\times 16 + 14 = 2062$  )。

(3) 为了防止用户按下CTRL-RESET键,使程序中断。我们可以将1010、1011单元置入某种特定的值,使计算机在按下RESET键后,转到某个单元去执行一个子程序。例如,按下CTRL-RESET键后,仍跳回BASIC程序执行的方法是在程序开头加上:

```
1 POKE 1010, 102:  
POKE1011, 213; CALL-1169
```

这样别人按了CTRL-RESET键,也不能停下你的程序,就达到了加密的目的。

## 2. 在程序存盘时的加密法

在程序存盘时,只要在文件名中加入一些控制字符即可,这样使用 CATALOG命令时,显示屏上不会显示出控制字符,从而起到保密作用。

这种加密法可用下面方法解密:

```
10 FOR K=768 TO 795  
20 READ A: POKE K, A: NEXT  
30 POKE 54, 0: POKE 55, 3  
40 DATA 201,141,240,21,201,136,240,17,201  
50 DATA 128,144,13,201,160,176,9,72,132  
60 DATA 53, 56, 233,64,76,249,253, 76, 240, 253  
70 CALL 1002  
80 END
```

文献出处:《电子与电脑》1987年2期39页

# 用LIST执行DOS命令程序保密又一法

山东农机学院 宋吉和

**摘要:**本刊介绍了多种BASIC程序加密的方法,这些方法各有千秋,但都有一致命弱点,即必运行程序以后才起保密作用,所以容易破密。本文提出一新的加密方法。该方法的特点是利用LIST命令来执行象FP、PR# 6等有关DOS命令,从而不必运行程序即可起保密作用。

本刊86年第三期《APPLE II 机程序的加密》及87年第一期《再谈APPLE II 机的加密和解密》两篇文章,介绍了APPLE II 机BASIC源程序加密和解密的几种常用方法,对于用户来说的确简单、实用,但这几种方法都有致命弱点,即必须运行程序后方起加密作用。因此均可用一种方法解密,即用一个DOS主盘启动系统,然后把存有加密源程序的软盘放入软盘驱动器,用LOAD命令调出源程序,这样再用LIST命令就可列出全部源程序清单。所以在使用中一般和其它保密方法一起使用,本文提出一种新的保密方法,供大家参考。

## 一、BASIC程序在内存中的存放格式

BASIC程序在内存中是以语句为基本单位用链状结构存放的，每个语句先用二个单元存放链指针，接着二个单元存放该语句的语句标号，随后相继单元存放语句的内容。语句内容存放时，每个BASIC保留字（以其特征值表示）均占一个字节，其它字符存放ASCII码，每个语句用\$00作为结束标志，程序存放的首地址由\$67和\$68单元指出，一般从\$801开始，例如语句0 REM \_\_\_FP\_\_（注：文中“—”字符表示空格符，下同）在内存中的存放代码为（从\$801开始）：

```
0C 08 00 00 B2 20 20 46 50 20
```

其中，\$08和\$0C为链针，指出一个语句存放的地址为\$80C

\$00和\$00为本语句的语句标号

\$B2为保留字REM的特征值

\$20为空格ASCII码

\$46为字母F的ASCII码

\$50为字母P的ASCII码

## 二、基本原理

我们知道，当打入LIST命令后，计算机会将内存中现有BASIC源程序列出。实际上，计算机执行LIST命令时，也具有PRINT命令的功能，否则，源程序不会在屏幕上显示。换言之，计算机在解释PRINT命令时调用了PRINT命令的解释部分。LIST命令在执行DOS命令时，若其后的第一个字符是Ctrl-D，则以后的字符按DOS命令解释；另外，控制字符\$0D会产生回车换行。因此，若某段连续单元的内容为：

```
0D 04 46 50 0D
```

当用LIST命令列源程序时，就相当于键入了FP←命令（符号“←”表示回车，下同），从而会将系统指针复位，则不但列不出源程序清单，而且会清除内存中的源程序，相当于打入了NEW命令，从而起保密作用。

## 三、具体实现

我们既要为源程序加密，又不能影响程序的执行，必须在BASIC语言的非执行语句作文章。例如REM语句，计算机并不执行该语句，但是REM语句中可包含任何可键入字符。因此，我们在程序中插入如下语句（我们且称之为保密句）：

```
n REM Ctrl-D FP←
```

其中n为语句标号

Ctrl-D表示同时按Ctrl键和字母D键。

该语句的语句内容在内存中存放代码为：

```
20 04 46 50 20
```

对照前例，只要我们能将两个存放\$20的单元内容转换成\$0D即可。

举例说明，假设我们在程序中加入如下保密句：

```
100 REM Ctr1 - D FP←
```

该保密句在内存中放的首地址为\$801。

现在按以下步骤操作：

1 打入CALL-151←进入监控系统。

2 根据BASIC程序存放的链状结构找到该保密句中两个空格所在的内存单元地址（本例为\$806和\$80A）。

3 将这两个单元的内容改写为0D即：

• 806, 0D←

• 80A, 0D←

4 用Ctrl-C←返回。

最后，用SAVE命令存盘，那么以后任何时候用LIST命令列到该保密句时，计算机将执行FP命令，从而将清除该程序，起保密作用。

## 四、其它

1 利用该方法可执行任何一个DOS命令，只要将FP两个字符改为相应的DOS命令即可。

例如，我们将保密句中FP字符改为FR#6，当用LIST命令列到保密句时，会使磁盘自举，重新调入HELLO程序运行，从而会冲掉加密了的程序。我们也可在程序开始（或程序末）加入以下两个保密句：

```
0 REM Ctr1-D DELETE加密程序的文件名
```

```
1 REM Ctr1-D FP
```

则当用LIST命令列表时，不但可清除内存中的程序，而且还可将磁盘删除磁盘中的文件。所以掌握了这一方法，一定会使你的程序更富有变化，技巧性更高。

2 为了增强破译难度，可在程序中加入多个保密句。

保密句越多，破译越困难，即使对于水平较高的破译者，往往也是一件较复杂的工作，从而使其失掉破译信心。

3 将保密句作为程序的第一个语句。

对于不大了解BASIC程序在机器内存放格式的同志，不妨按以下步骤操作，可在你的程序开头加入标号为0的保密句，即：

(1) 打入：

```
) 0 REM Ctr1-D FP←
```

(2) 打入：

```
) POKE 2054, 13: POKE 2058, 13←
```

或者：

(1) 打入：

```
) 0 REM Ctr1-D PR#6 ←
```

(2) 打入:

) POKE 2054, 13; POKE 2060, 13; ←

文献出处: 《电脑与微电子技术》1987年2期25—27页

## DOS3.3下文件的简易加密与解密

江苏 王才宝

这里介绍一种简便的磁盘文件加密法。将下面的程序键入计算机,将需要加密的磁盘插入1号驱动器,运行该程序,计算机便对驱动器内的磁盘进行加密。运行结束后,倘要CATALOG,则只能在屏幕上出现:DISK VOLUME 254

其余什么也不再显示,仿佛所有的文件名“消失”了一般(其实文件和文件名全都完好地存在,只是文件名被“隐藏”起来了)。经过这样处理的磁盘即便使用“FID”程序,也无法列出目录,但在列磁盘空间时,可以看到已用和未用的扇区。要使得这样的磁盘恢复正常,只要将程序二中的POKE24587, 0一句改为24587, 18后再执行,磁盘就又恢复到常态,这时列目录(CATALOG),就可以列出所有的文件名了。当然,还可以有“隐去”部分文件名的办法。

```
5  REM HIDE CATALOG
10  FOR A=833 TO 872: READ V: POKE A, V: NEXT A
20  DATA 169, 3, 160, 79, 32, 217, 3, 169, 0, 133, 72, 96, 0, 0, 1,
      96, 1, 0, 17, 15, 101, 3, 0, 96, 0, 0, 1, 0, 0, 96, 1, 0,
      0, 0, 0, 0, 0, 1, 239, 216
30  CALL 833
38  POKE 24587, 0: REM HIDE CATALOG & IF WANT RESTORE
      THEN CHANGE TO: POKE 24587, 18
40  POKE 859, 2
45  CALL 833
50  END
```

文献出处: 《软件报》1988年6月4日

## 修改DOS命令达到加密目的

邹 庆

所有的DOS命令都是在系统启动时,由磁盘载入的,我们可以改CATALOG,使非法用户看不到磁盘上的文件名。DOS内有一个DOS命令名称主文表。每一DOS命令都是以ASCII码形式存放在内存中,范围在(\$A884, \$A907)内。而CATALOG这一命令的七个字母在(\$A8D2, \$A8D8)内。修改这七个字母就算大功告成。为说明问题起见,先将英文字母表的ASCII码列表于下:

字母A B C D E F G H I J K L M

ASCII C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD

字母N O P Q R S T U V W X Y Z

ACHICE CF DO D1 D2 D3 D4 D5 D6 D7 D8 D9 DA

标准DOS的CATALOG命令的ASCII码是:43 41 54 41 4C 4F CT。这里,聪明的读者有一定能看出DOS命令的ASCII码除最后一个外,其余全部都加上了80(十六进制),溢出位舍去。值得注意的是,代替CATALOG命令的字母不宜超过七个,但若少于七个,应将后面命令依次靠拢。否则,会引起命令的混乱。下面的实例读者可试一试。

以字串ZHOU为加密磁盘的口令为例。取一INIT后的空磁盘,键入:

ICALL-151

• A8D2: 5A 48 4F D5

• A8D6<A8D9 · A907M

• FP

J INIT HELLO

经过这样的处理,可得到一个加密的空白磁盘。同理,读者可举一反三,修改所有的DOS命令。

文献出处:《苹果园》1987年6期28页

## APPLE II CP/M软盘加密方法

内蒙古乌兰察布师范专科学校 田 智 杨永竹

本文所介绍的加密方法是靠修改磁道来实现的。5<sup>1</sup>/<sub>4</sub>英寸软盘共有70个磁道,APPLE机原使用偶数号磁道,共有35个。为了防止拷贝,我们将其改为奇数号磁道。因此固化在驱动卡上的引导程序只能读0号磁道,所以0号磁道仍然保留,而跳过1号、2号磁道,实际使用0, 3, 5, ..., 69号磁道。为了达到这个目的,需重修改FORMAT与COPY程序,同时还需要修改系统,以下介绍具体方法。



## 1. 制备一个小工具ZAP

搞软件加密必须首先能够直接对软盘上指定的磁道、扇区进行读写。已有的ZAP软件就是用来执行这一任务的。但一般用户手头往往并不具备这一软件，且文献上所介绍的ZAP是多用6502汇编语言写成，程序较长，使用起来也颇为不便。不妨按下列方法动手自制一个ZAP，该程序仅有十四行，稍加应用你就会觉得它确是一个得心应手的小工具。以下以56KB的CP/M为例介绍制作方法。

进入DEBUG后用A命令从100H开始键入下列程序：

```
0100 LD IX,0200          0117 LD (IY+01),A
0104 LD IY,F3E0        011A LD A,(IX+03)
0108 LD A,(IX+00)     011D LD (IY+0B),A
010B LD (IY+04),A     0120 LD HL,0E10
010E LD A,(IX+01)     0123 CALL DB3B
0111 LD (IY+00),A     0126 JP AB99
0114 LD A,(IX+02)
```

退出DEBUG状态进入CP/M状态，再键入下述命令存盘，至此，工具制作完毕。

A>SAVE 1 ZAP ↵

使用方法如下：运行DEBUG，同时调入ZAP。（以下有下划线“—”的为计算机自动显示的内容）

```
-S200 ↵
0200 00 1 ↵ (指定1号驱动器) 0203 00 1 ↵ (命令码,01读,02写)
0201 00 0 ↵ (0磁道)          0204 00 ↵ (退回DEBUG)
0202 00 2 ↵ (2扇区)          -G100 ↵ (运行ZAP)
```

待屏幕重新出现“—”符时就表明1号驱动器0道02扇区内容已读入F800H—F8FFH。这段程序之所以能如此简练，其原因之一是因为它在DEBUG状态下运行，因此可以方便地借用DEBUG的命令来显示、改变、反汇编读入内存的内容；原因之二在于它巧妙地调用操作系统写程序，该程序用6502指令写成，它在CP/M引导过程中被写入内存。

## 2. FORMAT.COM的修改

FORMAT程序由两部分组成，第一部分用Z80指令编写（100H—3FFH），其功能是给出提示信息，请求键盘输入及命令处理；第二部分用6502指令编写（400H—96CH，对应的6502的地址编码为1400H—196CH），这一部分是真正的格式化程序，它实际负责对磁盘的每一个磁道格式化工作，其中有

```
1801-      A5 41          LDA      $41
1803-      0A           ASL
```

这两个语句负责将逻辑磁道转化为物理磁道（偶数号磁道），现将其（三个字节）改为转子语句。在CP/M状态下，调用DEBUG，同时将FORMAT.COM调入内存，键入以下命令：

```

A>DEBUG FORMAT.COM ↵    0801 A5 20 6D 19. ↵ JSR $196D
DEBUG Version 00.17      -S96D ↵
NEXT =0A00               096D 00 A5 41 0A C9 00 F0 03 18 69
                           01 60 ↵
NEXTM=0A00               -Go                退出DEBUG
-S801 ↵                  Z80编址    A>SAVE 9 FMT.COM    存盘

```

这样，格式化程序就修改好了，其中从96DH开始键入的码为6502指令码，其汇编语言为

```

196D- LDA $41 ; 目的磁道送A寄存器      1974- CLC ; 否则清进位
196F- ASL ; 逻辑磁道号乘2              1975- ADC# $01; 磁道号+1
1970 - CMP # $00 ; 是0号磁道吗?        1977- RTS ; 返回
1972 - BEQ $1977 ; 是则转RTS

```

现在运行FMT，以格式化一张新盘，该新盘便是一张使用奇数号的磁道的盘。（注意6502与Z80的编址不同，键入与调用的地址差1000H）

### 3. COPY.COM的修改

COPY.COM程序对磁盘的读写要调用CP/M系统的读写程序，逻辑磁道与物理磁道的转化也在CP/M系统的读写程序中，该转化方式需要修改，以保持与上述FMT.COM“同步”，来达到对上述格式化新盘的读写操作。我们要将需保护的软件拷贝到上述格式化的新盘上，以达到加密的目的，这就要求在读主盘时，仍按原来的逻辑磁道与物理磁道转化方式进行，而在写入新盘时，则按新的转化方式进行。为此对COPY.COM作如下修改，键入：

```

A>DEBUG COPY.COM ↵      -A208 ↵
DBBUG version 00.17     0208 LD DE,045E CALL 335 ↵
NEXT =0500              (改转子语句)
NEXT=0500               020B CALL 025C
-S332 ↵                (结束输入)
0332 20 0D 0A 24. ↵

```

然后用A命令从355H和4DDH开始分别键入以下两个程序。

```

0335 PUSH HL              033F INC HL
0336 LD HL,FF52          0340 LD (HL),0F
0339 LD (HL),0A         0342 POP HL
033B INC HL              0343 LD DE,045E
033C INC HL              0346 RET
033D LD (HL),5A         -
0226 CALL 04DD          04E7 INC HL
04DD PUSH HL            04E8 LD (HL),14
04DE LD HL,FF52        04EA POF HL

```

```

04E1 LD (HL,EA)          04EB LD DE,0487
04E3 INC HL              04EE RET
04E4 INC HL
04E5 LD (HL),EF

```

接着再输入下列命令

```

-S4EF
04EF 19 0A C9 00 F0 03 18 69 01 4C 5A 0F. ↵
-G0 ↵
A>SAVE 4 CP.COM ↵

```

退出

这样，COPY.COM就修改好了，其中第一部分是將COPY.COM读取子程序的入口处改換为一条转子语句，转向从355H开始键入的子程序，使得在读取主盘前还原逻辑磁道与物理磁道的转换方式。第二部分是將COPY.COM写入子程序的入口处改換为一条转子语句，转向从4DDH开始键入的子程序，该子程序將原来逻辑与物理磁道的转换语句改为一條转子语句（因新的转换方式语句较之原来的转换方式为多，在原来的地方容纳不下，因此只能作为子程序放在别的地方）它所调用的子程序便是后采用S命令键入的那段6502指令码，而这段指令码与FMT.COM中的指令码是一致的。

加密拷贝程序CP.COM与普通的COPY.COM使用方法完全一样。运行CP.COM即可加密拷贝，拷贝完毕新的磁盘便是一张加密的软盘，但请注意，这张盘现在还不能运行，这是因为盘上的内容是写在奇数号磁道上的，但拷贝在这张盘上系统中的逻辑磁道与物理磁道转换方式仍然没有改变。要使这张加密软盘能够使用，须使FMT、CP、系统三者“同步”，故对系统还须作修改。

#### 4. 加密软盘系统的修改

CP/M操作系统的读写程序位于FA00H—FFFFH，逻辑磁道与物理磁道的转换程序在FF52H（仅一个字节），现将该处改为转子语句，转向新的转换方式子程序，经查在FC70处有一段空区，可以放置该子程序，这两段区域与磁盘的磁道扇区对应关系为：

FC00H—FCFFH 0 磁道 3 扇区      FF00H—FFFFH 0 磁道 6 扇区

用ZAP对这两个扇区进行修改。用含有ZAP的软盘进行启动，新加密的软盘插入B：

键入：

```

A>DEBUG ZAP ↵          F853 20 ↵
DEBUG version 00.17    F854 5A 70 ↵
NEXT =0200             F855 0F 0C ↵
NEXTM=0200            F856 4E .
-S200 ↵               -S203 ↵
0200 00 2 ↵           2 号盘 0203 01 2 ↵ 改为写
0201 00 ↵             0 道 0204 00 .

```

0202 00 6 ↵	6 扇区	-G100 ↵	运行ZAP, 写入修改
0203 00 1 ↵	读		好的内容
0204 00 . ↵	退回DEBUG	-S202 ↵	
-G100 ↵	运行ZAP	<u>0202 06</u> 3	3 扇区
-SF852 ↵		<u>0203 02</u> 1	读
<u>F852 0A EA</u> ↵		<u>0204 00</u> .	
-G100 ↵	读入另一扇区	<u>0203 01</u> 2 ↵	
-SF870 ↵		<u>0204 00</u> ↵	
<u>F870 FF 0A C9 0F0 03 18 69 1 4C 5A</u> -G100 ↵			写入
0F. ↵		-G0 ↵	退出
-S203 ↵			

这样，加密软盘的系统便修改好了。这张盘的使用与未加密的CP/M盘完全一样。（当然，用该盘进行冷启动后，欲在B:或其它驱动器上使用别的软盘，则别的软盘事先须用FMT格式化）。

上述方法加密的软盘，LOCKSMITH, NIBBLE AWAY, BACK IT UP等强拷贝都对它无能为力。这里需要指出的一点是：若加密盘上本身带有COPY.COM程序，则用这个拷贝程序可以拷贝加密盘，这是由于这个拷贝程序调用系统本身的读写程序，而该程序已被修改，要避免这一点，只要将加密盘上的拷贝程序删除即可。

根据以上原理和方法，读者完全可以搞出具有自己特色的加密软件，使得即使知道这一方法的人也不易破译。另外，上面各步工作可以编入拷贝软件中，使加密拷贝过程自动化。

文献出处：《微型计算机应用》1988年2期46—49页

## 用BASIC在DBASE III 程序中加密

成都 温希涛

目前有很多对DBASE III 程序进行加密的方法。我对这些加密程序作过分析，几乎都是在文件名上作文章，即改变文件名属性，起到加密作用。但是，如果我们把这些文件名改为正常属性后，程序就不保密了。

我在这里介绍一种不改变文件名属性，而深入到程序内部进行加密的方法。

大家知道DOS系统的文件用十六进制的“1A”作为文件结束符号，在DOS系统下，用TYPE或编辑等命令时，一旦遇到“1A”就认为文件结束，从而不能继续打印或继续装入。我利用这一特点，把程序每一行结束的“OD”、“OA”两个字符中的“OA”改为“1A”，使之在TYPE和编辑等命令中不能进行，但在DBASE III 中能运行，起到程序保密的作用。

加密程序用BASIC语言编成(见程序清单①)。用此程序加密时, DBASE III程序不得超过500行, 如超过则把加密程序20句中的“500”改为所需要的值, 本程序经编译后使用非常方便。以下是使用举例: ①为加密程序。②为未经加密的DBASE III程序。③加密过程(不输入扩展名时, 加密程序隐含扩展名为PRG), ④为加密后DBASE III程序所显示的情况。

```
10 A=0
20 DIM F$(500)
30 INPUT "输入文件名: "; N$
40 IF LEN(N$) < 1 THEN GOTO 190
50 IF INSTR(N$, ".") < 1 THEN N$=N$+".PRG"
60 PRINT N$; "正在加密....."
70 OPEN N$ FOR INPUT AS# 1
80 IF EOF(1) THEN 110
90 LINE INPUT # 1, F$(A)
100 A=A+1; GOTO 80
110 CLOSE 1
120 OPEN N$ FOR OUTPUT AS# 1
130 PRINT # 1, ".....dBASE III Program" DATE$; "....."
    CHR$(13); CHR$(26);
140 FOR I=0 TO A
150 PRINT # 1, F$(I); CHR$(13); CHR$(26)
160 NEXT I
170 CLOSE 1
180 PRINT N$; "加密完成 1"
190 END
```

加密后程序运行情况与加密前完全一致, 本人所使用的机型为长城0520C—H。

文献出处: 《软件报》1987年11月2日

## 一种较为可靠的磁盘加密系统

### 电磁软盘加密系统介绍

李红川

目前, 国内外以磁盘为介质的软件加密方法众多, 其中以激光加密系统最为安全可靠。但激光加密却有着两个致命的缺点, 一是由于目前激光设备价格较贵, 一般的使用者很难达到目的。另一个是由于采用光刻法在一定程度上总要破坏盘表面的光洁度, 降低系统的可靠

性，一些使用者反映，用激光加密的软盘使用稳定性较差。据调查，目前国内外并没有很好地解决这个问题。本文将介绍一种基于电磁方法设计的磁盘加密系统，由于电磁的获得及强弱控制比起激光方便经济得多，因而，基于电磁法实现的软盘加密系统给使用者带来的好处是不言而喻的。

电磁软盘加密系统（简称电磁磁盘）主要由两部分构成：密钥产生和反跟踪密码系统。密钥产生是借助专用的电磁机构在所加密的数据盘上随机地生成一系列密钥信息，供数据盘上专门的密钥搜索程序识别。由于生成的密钥在盘上的位置是随机变动的，每张盘的密钥不可能相同，这就给解密分析增加了困难。特别要说明的是，采用上述方法在磁盘上生成的密钥信息与一般的信息在编码方式（FM或MFM）上有着很大的区别，从理论上讲，单纯采用软件方法的拷贝程序是不可能进行原样复制的。这一点正好和激光加密达到的效果不谋而合。磁盘上信息的编码，就是在信息中间以一定的格式插入同步脉冲，主要目的是为了改善当信号为全“0”或全“1”时磁头的频率特性，而这些个同步脉冲对信息本身来说是无用的，从盘上读出后必须与有用信息分离开来，这就要求在读写信号过程中系统必须保持严格同步。当磁盘上的信息是采用一种奇异的编码格式写入，一般的系统对其进行读、写操作时就会产生同步混乱，从而使读出的信息出现随机错误。电磁磁盘正是利用这一特性来阻止复制程序正常工作的。

反跟踪密码系统主要是由软件构成，其中包括了反跟踪系统、密钥搜索程序和密文的解密程序。密文程序是采用多次换位、代替复合运算即所谓乘积密码的方式对明文程序处理得来的，加密后生成的密文程序可靠性较高。解密程序是加密程序的逆过程，其工作原理简要说明如下：

设密钥空间为  $K = \{k_1, k_2, \dots, k_n\}$ ， $T = \{t_1, t_2, \dots\}$  为输出的密文， $R = \{r_1, r_2, \dots\}$  为输出的明文， $m$  为运算的次数， $T_i$ 、 $K_i$  为第  $i$  次运算后的结果，于是由：

$$\begin{cases} T_i = \beta_i(T_{i-1}, K_i) & (1 \leq i \leq m) \\ K_i = \varphi_i(K_{i-1}) \end{cases}$$

得到：

$$\begin{aligned} R &= f(\beta_m(T_{m-1}, K_m), \varphi_m(K_{m-1})) \\ &= \{r_1, r_2, r_3, \dots\} \end{aligned}$$

可以看出，最后输出的明文是通过多次的复合运算结果，其中密钥  $K$  本身也是中间变量，在程序的运算过程中不断变化。反跟踪系统是利用被调试的程序与动态调试程序（DEBUG），在一些区域发生重迭这一特点，于程序中的多处设置反跟踪陷阱，来阻止动态跟踪分析。

把上述的两部分有机地结合起来，就构成了我们的电磁磁盘。因密码系统和反跟踪系统一般由软件来实现，这就使其灵活性很大，相信读者还会有更好的方法，这里不多述。

前面介绍过，激光加密就是在磁盘上的某个道上用激光刻上一些印记，这种印记是永久性无法抹掉的，一旦这种盘要作为它用（虽然这种情况很少会出现），则这些印记区域在重新格式化时将被当作坏块处理，也就是说激光加密是以局部地破坏磁盘记录作为代价来阻止他人复制的，虽然在适当调整激光强度情况下其破坏的面积可以控制得很小，但多少要影响盘面的光洁度，增加磁头的磨损。而用电磁法在盘上做的密钥可随意抹去，不会在盘上留下任何痕迹及损坏盘面的光洁度，提高了磁盘子系统正常工作的可靠性。另外要说明一点，用本文介绍的方法生成的电磁磁盘，由于没有改变原磁盘的物理特性，因而其稳定性高于激光

加密,但它又和传统的格式加密有着本质的区别,其可靠性远远高于它们,不可并论。

总之,我们认为电磁加密实为一种较安全可靠的磁盘加密方法,和激光加密比较有着很多后者没有的优点。利用上述方法,笔者对本单位的几个软件进行了加密处理,通过一年多的使用证明,效果很好。现已用各种流行的COPY程序作过实验,结果都很满意。

文献出处:《计算机世界》月刊1987年10期40—42页

## APPLE—II 操作系统技术与磁盘加密方法

李联敏

APPLE DOS是以磁盘读写操作和磁盘文件管理为基本任务的系统管理程序。DOS提供了基本存取,文件操作,文件处理辅助命令等近三十条命令。DOS系统程序主要由三个部分组成:

### 1. DOS命令区(\$9D00—\$AAC9)

这部分包括DOS命令解释程序,DOS启动程序入口,BASIC解释程序接口等。

### 2. 档案管理区(\$AA9C—\$B600)

### 3. RWTS磁盘读写驱动程序(\$D00—\$C000)

这是磁盘读写执行和磁盘初始格式化工作程序。

用户是通过命令解释DCI和DOS实现信息交换的,而档案管理器FMS和磁盘驱动程序RWTS对用户是透明的,不需要用户直接干涉。RWTS程序是DOS系统程序中核心部分,直接管理磁盘的读写操作,它和硬件有紧密的联系。DOS系统也提供了调用RWTS程序的入口和方法。

## 一、部分程序段落分析

### 1. DOS命令扫描分析程序:

\$9FCD—\$A197是DOS系统程序中的命令扫描程序段,DOS命令由键盘输入后,进入命令扫描分析,完成DOS命令的判断,识别和有关参数的处理,最后转入命令的执行。命令处理程序执行后跳回调用程序。

### 2. 与DOS扫描分析有关的程序段及参数是:

- (1) DOS命令名称表(\$A884—\$A908);
- (2) 命令处理程序入口表(\$9D1E—\$9D55);
- (3) 命令有效键语表(\$A909—\$A940);
- (4) 键语值有效范围表(\$A955—\$A970);
- (5) 错误信息表(\$A971—\$AA3E);
- (6) 重要参数及其地址。

\$AA4F—\$AA65    DOS主要变量表;  
 \$ZAA66—\$AA74    命令分析取得键语值;  
 \$AA75—\$AA92    主档名缓冲区;  
 \$AA93—\$AAB0    次档名缓冲区;  
 \$AAB1—\$AAC0    DOS主要常数:  
     \$AAB2    \$84 (CTRL—D);  
     \$AAB6    BASIC类型标志;  
     \$00      INT BASIC;  
     \$40      ROM BASIC;  
     \$80      RAM BASIC; 等等。

3. DOS命令扫描分析程序注释: (略)

4. DOS命令处理程序:

\$A229—\$A60D是DOS命令处理程序区, 共28个DOS命令。下面以INIT命令为例, 进行分析注释, 并将与INIT命令直接关联的程序段同时列出分析。INIT命令处理程序 (\$A54F—\$A5D6)

```

LDA   # $40      ; $40   A
AND   $AA65     ; 查键语表 (V 册号)
BEQ   $A553     ; 结果为零转 $A55B (册号取254)
LDA   $AA66     ; $AA66内容为送A (册号)
BNE   $A560     ; 结果不为零转 $A560
LDA   # $FE     ; 册号   A
STA   $AA66     ; A     $AA66
LDA   $9D0D     ; DOS参数  A
STA   $B5BC     ; A     档案参数表
LDA   # $0B     ; $0B   A
JSR   $A2AA     ; 转 $A2AA (档案管理公用程序)
JMP   $A397     ; 转 $A39 (SAVE命令处理)
  
```

这一段程序主要完成册号的设置, DOS第一页参数的传送, 转入档案管理公用程序, 完成抄录档案名称, 参数的写入 (\$A71A), 并设定命令的运算码, 转入 \$A6A8档案处理程序。

INIT动能处理程序注释: (略)

RWTS驱动程序 (\$B052—\$B0B5)。

由RWTS驱动程序转子进入 \$B7B5停中断, 并调用RWTS, 转入 \$BD00的RWTS主进入点。

在执行INIT命令的过程中, 较为重要的几个程序段是:

\$BEAF—\$BF0C    FORMDSK程序;  
 \$BF0D—\$BF61    TRACBWRITE程序  
 \$BC56—\$BCC3    初始化写入位址;  
 \$B82A—\$B8B7    WRITE程序;



\$B8DC—\$B94B READ;

通过这些程序段，INIT命令完成了：

- (1) INIT命令分析；
- (2) 寻找相应的功能处理程序入口；
- (3) 格式化磁道 (FORMDSK)；
- (4) 写位址区参数 (初始化设定)；
- (5) 写资料区参数 (TRSTRK—WRITE, WRITE)；
- (6) VTOC (规映表) 写入；
- (7) 写入DOS；
- (8) 写入HELLO (引导程序)；
- (9) 返回调用。

## 二、磁盘的保护和解密

### 1. 改变DOS磁盘格式参数加密方法

标准的DOS磁盘在进行初始化工作时就完成了规道格式及参数的设置。这一工作的完成是由DOS系统程序中，初始设定子程序 (\$BC56—\$BCC3)；WRITE子程序 (\$B82A—\$B8B7)；READ子程序 (\$B8DC—\$B943) 和RDADR子程序 (\$B944—\$B99F)等程序完成的。有关DOS磁盘格式以及参数形式这些程序中的相应参数决定。改变这些参数值可以制作出非标准格式的磁盘。如同步字节值，位址区起始标记，结束标记等参数的改变都能使正常的拷贝程序无法成功的复制，而达到保护的目的。

例如将位址栏和资料栏结束标记 \$D5/\$AA/\$EB中的 \$AA改变为 \$92，可作如下操作：

```
] CALL—151
```

- B8A3: 92
- B99B: 92
- B93F: 92
- BCB3: 92
- AB

```
] INIT HELLO
```

这样格式化后的磁盘可成为非标准格式磁盘。对于DOS磁盘格式有关的参数，其取值不是任意的，\$D5，\$AA是磁盘数据的保留字节，不允许在位址区内或资料区内出现，以确保某些标志“唯一性”免受破坏，造成磁盘数据的混乱，以至完全不能使用。

### 2. 修改DOS命令制做非标准磁盘

如前所述DOS命令的键入到执行经过键盘接收程序，命令扫描分析程序，DOS命令执行程序多个环节。这些工作环节中有一处被修改都会造成命令不能正确执行。

例如：

### (1) 改变命令字符

修改 \$A884—\$A8C6 位址所对应的命令名称表可以改变DOS命令的形式。如将 \$A885 的 \$4E 改为 \$49, 那么 INIT 命令将改变为: IIIT。其它命令形式均可以改变, 但一般应维持原有命令字符个数, 并保证最后一个字符的二进制码最高位置为 1。

### (2) 改变DOS命令键语或命令属性:

位址 \$A908—\$A940 存有DOS命令的键语表, 每一命令用两个字节对键语和特性进行限定。改变这些键语表的字节值可以使命令的键语或属性改变。

例如, INIT 的键语标识字节值为 \$2170, \$2170

0 0 1 0 0 0 0 1 1 1 1 0 0 0 0
-------------------------------

把决定命令执行的第六位设置为 1, 这样 INIT 命令将只能以间接方式执行, 不能由键盘直接键入执行。

### (3) 改变DOS命令入口地址:

在DOS中 \$9D1E—\$9D55 位址区间按照DOS命令表的标准顺序, 存放DOS命令处理程序的入口位址表, 这些入口参数改变必然引起命令功能改变, 例如为了屏蔽 CATALOG 或 LIST 命令动能, 可将其对应入口变为 RUN 命令的入口 (\$A4D1), 那么在打入 CATALOG 或是 LIST 命令时执行的是 RUN 命令功能, 原来动能失效而达到防止列印清单或目录的目的。

## 3. 移动VTOC保护磁盘

标准DOS系统的磁盘文件管理主要是依靠VTOC (Volume Table of Contents) 提供规道扇区使用状况来管理磁盘。正常的DOS将VTOC固定放于17规 (\$11) 零扇区。移动VTOC, 那么正常DOS将因找不到VTOC而不能正常读写磁盘。

移动VTOC做法可在BASIC状态或监控状态下, 将VTOC所在轨道号送入有关单元:

```
] POKE 44703, N  
] POKE 44764, N  
] POKE 44033, N  
] INIT HELLO
```

其中N是小于35, 而不是0, 1, 2的整数。把正常DOS系统磁盘文件转到移动过VTOC的磁盘上, 在读和写操作时相应改变内存44033位址单元内容为VTOC所在规道号

对于移动过VTOC的磁盘, 只要 \$11 轨未用, 还可以复制一个假的目录轨到 \$11 轨上, 在真的VTOC中指出 \$11 已占用。对这样的磁盘用CAT—ALIG调查文件, 看到的是假目录, 这就是双目录保护法, 如果把两个目录用不同磁盘格式分配给两个区域使用, 各自的VTOC同时注册两个区的轨扇使用状态, 这就是双DOS保护法。这是一种行之有效的保护方法, 因为往往一种解密方法不会对两种格式都有效。

## 4. 备用轨道保护法

正常DOS系统使用的磁盘轨道是从零轨到35轨, 但实际上可以使用更多的轨道。磁盘的第36轨称为备用轨。如果将某些信息资料存放在备用轨上, 而正常的拷贝程序难以复制,

因而得到保护。

在BASIC状态下键入下列内容：

] POKE 44703, 36

] POKE 44741, 74

] POKE 44764, 36

] POKE 44033, 36

] POKE 48894, 37

] POKE 44725, 148

然后键入引导程序 (HELLO)

] INIT HELLO

格式化生成的新盘已将VTOC及目录放置于备用轨上。

原则上DOS系统的软磁盘还可以使用更多的轨道，但是随着轨道号的增加，轨道长度变小，数据的可靠性受到一定的影响。

## 5. 文件脱解方法

要把某些程序从加密盘上转移到标准DOS系统磁盘上的过程叫做文件脱解。进行文件脱解的基本原理是：把加密磁盘文件载入内存，将其移动到不受BOOT磁盘影响的区域，然后启动正常DOS系统，再把程序重置，最后存入正常的DOS系统磁盘上。通常BOOT一个空的DOS3.3软磁盘时，内存 \$900—\$95FF位址区域是不受影响的，这是隐藏程序的安全区域。

下面以BASIC程序和二进制文件的脱解为例，介绍有关文件脱解的办法和技巧。

(1) APPLESOFT BASIC程序由加密系统脱解

(a) 首先制作一个DOS3.3系统空磁盘，要求HELLO引导程序是空白的，或是及短小的程序。具体做法：

i. BOOT DOS3.3系统磁盘；

ii. NEW 清除内存；

iii. 换上空白磁盘，键入INIT HELLO

(b) BOOT加密磁盘系统

i. PR#n (n为磁盘机所在槽号)；

ii. LOAD 文件名 (cr) 装载要脱解的BASIC程序；

(c) CALL—151 (cr) 进入监控状态；

i. 视查 \$67, \$68, \$AF, \$B0位址内容，记下程序载放的起始位址，结束位址，并计算出其长度。

ii. 把程序移至“安全区域”：

例如，原来程序起始位址为 \$800，结束位址为 \$2000，则做如下操作：

• 400 < 800, 2000M (cr) 程序即移至内存 \$4000单元开始的区域。

(d) BOOT标准DOS3.3空磁盘

NAP (CTRL—P) N为磁盘驱动卡所在槽号。将正常DOS系统装入内存。

(e) 重置程序及有关指针

• 800<4000, 5800M ( cr )

将 \$67, \$68, 及 \$AF, \$B0位址内容置为程序起始和终止。

( f ) RESET或^B ( cr ) 回到APPLESOFT状态;

( g ) SAVE文件名 ( cr ) 将程序存入正常的DOS3.3系统磁盘上, 完成BASIC程序的脱解。

( 2 ) BINARY ( 二进制 ) 文件由加密系统脱解

( a ) 首先制做DOS3.3系统空磁盘;

( b ) BOOT加密磁盘系统;

( c ) BLOAD ( 二进制 ) 文件名 ( cr ) 装载要脱解的二进制文件。

( d ) CALL-151 ( cr ) 进入监控;

( e ) 调查记录 \$AA72, \$AA73及\$AA60, \$AA61位址内容, 找出程序起始单位及程序长度;

( f ) 计算移动程序前后结束位址;

( g ) 移动程序到安全位址范围;

( h ) BOOT DOS3.3空磁盘;

( i ) 把移动过的程序重置;

( j ) 更改 \$AA60, \$AA61, \$AA72, \$AA73位置内容为程序的长度及程序的起始位址;

( k ) 回到BASIC状态, 将二进制文件用BSAVE命令存盘, 即完成二进制的脱解。

磁盘格式主要参数位址表

标准参数值		WR子程序	RE子程序	DR子程序	初设子程序
位 址 区	起始标识符	\$D5		\$B955	\$BC7A
		\$AA		\$B95F	\$BC7F
		\$96		\$B96A	\$BC84
区	结束标识符	\$DE		\$B991	\$BCAC
		\$AA			\$BCB3
		\$EB		\$B99B	\$BCB8
资 料 区	起始标识符	\$DA	\$B853	\$B8E7	
		\$AA	\$B858	\$B8F1	
		\$AB	\$85D	\$8FC	
区	结束标识符	\$DE	\$B89C	\$B935	
		\$AA	\$8A3		
		\$EB	\$B8A8	\$B93F	

# APPLE-II磁盘加密一法

南京 陈弘

单一的加密方法往往很容易被破解，所以好的加密技术应是各种方法的结合。本文介绍一个加密程序，经加密后的磁盘即使别人拷贝去也无法使用。若能再结合其它方法，你的软件必能得到更好的保护。

在该加密技术中用了两种加密方法：

一、把目录磁道移到别的磁道上去(不是第17磁道)。这样，别人就不可能通过导引标准DOS来CATALOG你的磁盘了。

二、修改常用DOS命令。如LOAD、INIT、CATALOG等(修改后的命令用户必须牢记，否则自己也无法使用该盘)，别人即使导引了你的磁盘也无法运行你的程序。

加密程序如下：

10~70句修改目录磁道。用户只要键入一数( $2 < N < 35$ )即可。80~230为修改DOS命令，这些命令修改以后别人就无法对该盘进行操作了。为简便起见，修改后的命令必须与原来命令等长，240句为对一新盘进行格式化，B\$(1)为修改后的初始化命令。250句删去HELLO程序。这样处理的盘，PR# / 6，再CATALOG... .. SYNTAX ERROR! !

```
0LIST
10 REM
20 FOR I=1 TO 9:READ A(I), NEXT
30 INPUT "Which track ?" ; N
40 IF N < 3 OR N > 35 THEN 30
50 FOR I=1 TO 7:POKE A(I), N:NEXT
60 POKE A(8), 4 * N:POKE A(9), 4 * N + 4
70 DATA 44033, 44703, 44764, 45715, 46268, 46568, 46012, 44741, 44745
80 REM CHANG DOS COMMAND
90 DATA INIT, LOAD, SAVE, RUN, BLOAD, OPEN, CATALOG
100 DATA 43140, 43144, 43148, 43152, 43257, 43202, 43218
110 FOR I=1 TO 7:READ A$(I):NEXT
120 FOR I=1 TO 7:READ A(I):NEXT
130 FOR I=1 TO 7
140 HOME:VTAB 3:PRINT "    ", A$(I)
150 INPUT "You Want to Chang to" ?; B$(I)
160 IF LEN(B$(I)) < LEN(A$(I)) THEN PRINT "Sorr Wrongy
length ?" ; GOTO 150
170 NEXT
180 FOR I=1 TO 7
190 FOR J=1 TO LEN(B$(I))-1
```

```

200 POKE A(I)+J-1, ASC(MID$(B$(I), J, 1))
210 NEXT J
220 POKE A(I)+J-1, ASC(RIGHT$(B$(I), 1))+128
230 NEXT 1
240 PRINT CHR$(4); B$(1); "HELLO"
250 PRINT CHR$(4); "DELETF HELLO"

```

文献出处:《软件报》1988年3月19日

## 对汉字码加密的解密

何 柱

用汉字码在微型机上对子目录名(文件名、记录项名同理)作加密处理的主要思想是:当子目录名包含了残缺汉字,其它人无法用一般方法得到其完整的名称,也就无法对子目录进行操作,这样就达到了对子目录加密的目的。

其实,我们要进入某一子目录,并不一定要清楚目录名的每一个字符,只要能得到该目录名即可。从这一思想出发,我们用输出转向的方法,对子目录解密。

```
CY<DIR> 1-01-80 12:03a
```

1个文件233472字节空间

```
C>
```

```
C>cd\CY<CR>
```

无效的目录!

```
C>
```

当不知道子目录名CY之后的密码,将无法进入该子目录。

其解密过程:

```
C>dir CY.*>W.BAT
```

输出转向至批处理文件W.BAT,然后进入WORDSTAR字处理系统编辑W.BAT文件。

```
C: WBATFC=1 FL=1 列01
```

```
INSERT ON
```

标示 磁盘C没有标示!

目录 磁盘C: \

```
CY<DIR> 1-01-80 12:03a
```

1个文件229376字节空间

将无用的信息删除,仅留下子目录名部分,并在子目录名前插入“CD\”。

将上面修改后的W.BAT文件存盘,执行W.BAT批处理文件,即进入加密的子目录,也即完成该子目录的解密。

上述手法只要稍作变化,就可用于对文件名、记录项名的解密。

文献出处:《计算机世界》1988年2月3日

# 为dBASE III 增加共享文件加锁功能

刘炳涛

**编者：**能否为dBASE III 增加新功能？随着dBASE III 的广泛应用，会有越来越多的人提出这个问题。本文提出的方法，不仅可解决dBASE III 在网络上使用时的文件共享问题，而其中增加dBASE III 命令的技术，也能用来扩充dBASE III 其它功能，有兴趣的读者不妨一试。

我们在使用Omninet网络时，也遇到了本刊87年第5期“共享文件锁定”一文中所提出的问题，即由于dBASE III 是一个单用户数据库管理系统，因此当两个或两个以上的用户对dBASE III 数据同时进行读写操作时，数据库内容就有可能被破坏。所以在网络上实现dBASE III 数据库的共享，必须对dBASE III 进行一些处理。“共享文件锁定”一文所提出的方法有一定的局限性，例如，当两个用户同时打开信号库时，就可能出错。此外，共享文件较多时，信号库管理和操作较为烦琐。我们在解决这个问题时，采用了另一种方法，即在不增加dBASE III 系统内存占用的前提下，保留原dBASE III 的全部功能，同时增加两条新的命令：plug（加锁），unplug（解锁），并可利用dBASE III 的file（）函数对加解锁状态进行检测。其功能相当于dBASE III Plus中的lock（）函数，和unlock命令。扩充后的dBASE III 适用于Omninet网，以太网，K-NET网，plan网等。

新增加的命令格式如下：

加锁：plug 文件路径与文件名

解锁：unplug 文件路径与文件名

函数格式为：file（‘文件路径与文件名’）

file（）函数值等于.T，表示数据库没有被占用，可以使用，函数值等于.F，表示数据库已被占用，现在不能使用。文件加锁后，file（）值为.F.；文件解锁后，file（）值为.T.。

下面我们介绍一下增加命令时的一些考虑及新命令的添加方法。

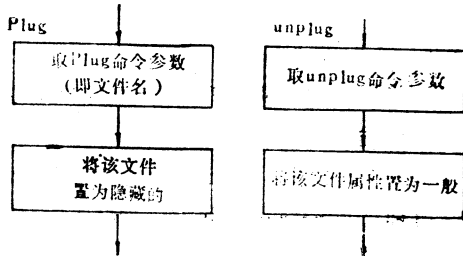
## 一、新命令功能

新增加的命令要达到下面的功能：

- （1）能对所有的数据库文件，命令文件进行加、解锁。
- （2）能对加、解锁状态进行检测。
- （3）命令的使用要简单，实用。

由于考虑到扩充后的dBASE III 要能用于网络，因此，加锁标志文件不能存于内存，必须存于共享的磁盘上，只有这样，才能使网络上的所有用户共同使用该加锁标志文件。但如果单独建立一个加锁标志文件的话，则必须增加新的命令建立和维护这个文件，这就使扩充工作更加麻烦，同时用户使用也不方便。因此，我们考虑使用磁盘文件目录的空余项来标志

文件的加解锁状态。同时我们在分析dBASE III file函数时，发现该函数对于不存在的文件及隐藏文件均返回逻辑值.F.，对于一般文件返回值为.T.，因此我们决定使用文件目录中的文件属性项来标志文件的加解锁状态，文件属性为隐藏时表示加锁，文件属性为一般时表示解锁。这样就可以借助file函数完成对加、解锁状态的判断，减少扩充的工作量，同时达



到预期的扩充目的。

至此我们对于plug、unplug命令所要做的工作已经清楚了，plug就是将指定文件的属性置成隐含的（加锁）；unplug就是将指定文件置成一般文件。

## 二、新命令的实现

要实现上面的考虑需要弄清以下几个问题：

- (1) 怎样使dBASE III接收新的命令（也即新命令的入口）
- (2) 新命令怎样接收输入的参数
- (3) 在不增加内存的情况下，怎样装配新命令。

通过分析，我们得知，dBASE III系统中有一个命令表，该表存于dBASE.OVL文件中。dBASE III启动时，将该表调入内存，对于键入的每条命令，系统首先查找该表，取出相应命令的入口地址，转入相应的命令处理子程序，命令执行完后返回系统。因此要在dBASE III中增加新的命令必须对该表的结构和内容进行分析。通过分析我们得知，也许是为了以后的扩充，dBASE III命令表中有一些手册中没有说明功能的命令。如poke、directory、load等这些命令的入口处是一些错误处理程序或一些没有提供的功能。因此我们决定利用这些入口编制我们自己的新命令。根据对系统的分析，我们选取了poke、unlock的入口地址作为我们新命令的入口地址，原因有两条：1. poke、unlock的处理程序在dBASE.exe文件内，这样我们的新命令也在dBASE.exe文件内，速度快。2. poke的处理程序是一个没有说明

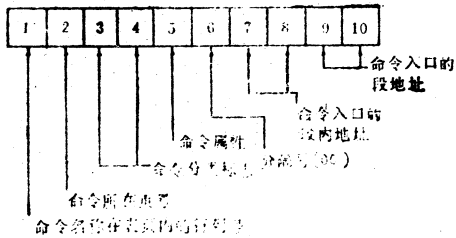
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
:																
:																
E																
F																
:																
:																



的功能，空间较其它未定义的命令大，便于增加新的命令。unlock 虽然入口处是一个出错处理程序，空间较小，但它同 poke处理程序同处于一个段地址内，以后编程方便，减少代码量。

命令表由不同的页组成，每页由一个 0 ~ F行，0 ~ F列的二维表组成。命令表后面是对应命令的属性及入口地址。命令表形式如下：

命令表后面是属性及入口地址，每一项10个字节，结构如下：



其中命令入口的段地址，在dBASE III启动后，会重新定位。

poke的入口地址为xxxx; 1751, unlock的入口地址为xxxx; 1EBE。其中xxxx为段地址。

这个命令表的地址，可以用下面的方法找到：

- a. 用debug将dBASE.OVL装入
- b. 用R命令查看CS寄存器内容
- c. 用R命令改变CS内容 (CS原值加1000H)
- d. 用S命令查找，如找POKE:  
—S0000 FFFE "POKE"  
—xxxx : xxxx

其中的xxxx : xxxx即为POKE的地址

下面的工作就是用 debug程序将编写的新命令程序写入dBASE.exe文件。程序如下：

```
plug程序: xxxx : 1751 MOV CX, 2
          1754 PUSH B, P
          MOV BX, DX
          SUB SP, +12
          MOV BP, SP
          PUSH ES
          MOV AX, 4301
          LES DX, [1120]
          POP DS
          INT 21
          MOV DS, BX
          XOR AX, AX
          ADD SP, +12
          POP BP
          RETF
```

unplug子程序如下：

```
MOV CX, 20
JMP 1754
```

最后利用debug程序将dBASE.OVL命令表中的poke改为plug, unlock改为unplug, 写回 dBASE.OVL文件即可。当然, 利用debug将这段程序写入dBASE.exe时还要注意 EXE文件重写的问题, 例如必须先将要改的exe文件改为非exe文件等, 许多文章对此均有介绍, 本文即不详述。

经过上述修改后的dBASE III, 启动后就可以直接使用plug、unplug命令, 并结合使用file函数, 完成对网络中数据库、命令文件的加、解锁, 及加、解锁状态检测。虽然只增加了两条命令, 但在网络上非常实用。

同时, 这种增加命令的方法, 也可为增加 dBASE III的其它功能借鉴。本文所使用的dBASE III为1.00S版, 其中dBASE.EXE为105442字节, dBASE.OVL为147456字节。

文献出处: 《计算机世界》月刊 1988年4期19—21页

## dBASE III 数据库软件的加密和解密

云南工学院计算机系 宋彬

dBASE III是目前最为流行的数据库软件, 但是, 此软件是加了密的, 要拷贝和使用都有些麻烦。下面向大家介绍一下dBASE III软件是怎样加密的以及解密的方法。

dBASE III的加密有两层。一是dBASE III原盘上的第0面第36道第5扇区是经过特殊处理的, 这个扇区的CRC标志被置为出错状态。对于这样的扇区, 用DOS的COPY命令或普通拷贝软件是无法拷贝的。在运行dBASE时, 程序先去检查A盘上的这个扇区的CRC标志是否是出错状态, 如果是, 则说明是原盘, 继续往下执行。如果不是, 则说明是拷贝盘, 不继续执行。这样就使得拷贝后的dBASE盘无法投入运行, 从而达到加密防拷贝的目的。dBASE的第二层加密是对dBASE文件的代码的每一个字节都和一个特定的值进行逻辑异或, 这样的代码文件, 就是用汇编软件及汇编出来, 也是无法弄懂的。dBASE本身运行时也是解密一段, 执行一段。这个解密程序的入口在XXXX:1A25, 入口参数: DI=要解密程序段的首地址, CX=要解密程序段的尾地址, AH=用于异或解密的值。

对dBASE III解密的思路是, 把程序中检查A盘特定扇区的指令INT 13H去掉, 并用STC来模拟CRC错。这样在运行时就不去检查A盘的那个扇区了, 用一般拷贝软件拷贝dBASE III软件时, CRC标记拷贝不过来也没有关系。这样, dBASE软件就能用DOS的COPY命令式一般拷贝软件拷贝, 也就是解了密。而且在运行硬盘中的dBASE软件时, 就再不必在A驱动器上放软盘。

具体解密步骤是:

1. 格式化一张新盘, 拷入DBASE.EXE文件后, 再拷入DBASE.OVL和COMMAND.COM等文件。
2. 把这张盘放在驱动器B上, 把带有DEBUG.COM文件的软盘放在驱动器A上。
3. A>DEBUG ↵

—L 0 1 10 1  
 —E 01AC B7 DE  
 —W 0 1 10 1  
 —Q  
 A>

这样，B盘上的dBASE文件就是已解了密的了。

文献出处：《微型计算机》1988年2期74—75页

## 计算机网络中的数据加密和密钥分配

黄祖良 何松涛

微处理机的发展和应用大大提高了终端和控制器的计算能力，可编程通信控制器和可编程控制器的应用，把网络管理职能转到网络设备上，从而提高了主机处理的功能和自动化程度。根据这些特点，网络中的密码设备必须满足高速、密钥量大，反复周期长、便于计算机处理等要求。本文就网络中的数据加密方式作一介绍，对数据加密设备和数据网络中密钥的分配问题提出了一些建议。

### 一、数据网络的加密方式

一般在数据网络中的加密方式有链路加密、节点加密和端对端加密等三种方式。

#### 1. 链路加密

链路加密是用于保护通信节点间传输的数据。这种方法比较简单，实现起来也比较容易，只要把一对密码设备安置在两个节点间的线路上，使用相同的密码即可。其关系如图1。

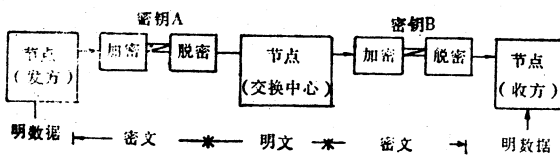


图1 链路加密方式

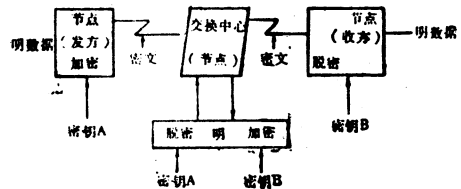


图2 节点加密方式

这种加密方式有两个缺点。一是全部报文都以明文形式通过各节点计算机的中央处理装置，在这些节点上，数据容易受到非法存取的危害。另一个是由于每条链路都要有一对加、脱密设备和一个独立的密钥，因此成本比较高。

## 2. 节点加密

这是链路加密的改进，目的是克服链路加密在节点处易遭非法存储的缺点。它与链路加密类似，差别只是加密算法要组合在依附于节点的保密模块中，其结构如图2。

节点加密也是每条链路使用一个专用密钥。但是一个密钥到另一密钥的变换是在保密模块中进行的。这个模块设在节点中央处理装置中可以起到一种外围设备的作用。所以，明数据不通过节点，而只存于保密模块中。但要注意的是，对于相当多的电报数据，在路由选择信息等，也要加密。这样，节点中央处理装置就能够恰当地选定数据的发送线路。

## 3. 端对端加密方式

在一个大型网络中，交换网络在多个发方与收方之间传输数据的时候，用端对端加密是比较合适的。该方式如图3。

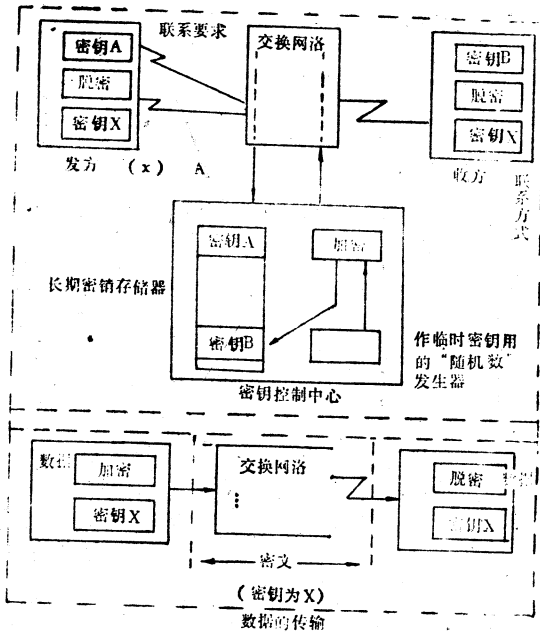


图3 端对端加密方式

这种方式在通信系统的某个地方设置一个密钥控制中心，它存储着系统中各端点专用的长期密钥。当某一端点欲与另一端点进行通信联系时，首先把这个要求发给密钥控制中心。接着该控制中心产生一个会话密钥，用收发双方各自的长期密钥对其加密后发给各方。双方在收到加密的临时密钥后，立即用各自的长期密钥进行解密，然后用解密后的对话密钥以端对端加密方式进行会话。

端对端加密具有链路加密和节点加密所不具有的优点：其一成本低。由于端对端加密在中间任何节点上都不解密，即数据在到达最终目的地之前始终是用密钥加密保护的，所以仅要求发送节点和最终的目标节点具有加解密设备，而链路加密则要求处理加密信息的每条链

路配备有分立式密钥装置。其次，数据在到达终点之前始终保持加密的形式，使得端对端加密比链路加密更为安全。采用端对端加密，其控制中心的加密设备可对文件、通行字、以及系统的常驻数据起到保护作用，然而由于端对端加密只是加密报文、数据报头仍需保持明文形式，所以数据容易为报务分析者利用。

由此可见，在大型的数据网中要较全面地衡量和考虑加密算法和密钥管理系统，使之在网络中使用方便、密度较高、成本较低，并且有验证等功能。

## 二、关于加密算法的选择

在通信和数据处理系统中，密码可分为分组密码和序列密码两种。密码算法可分为传统密码算法和公开密钥密码算法两种。对于数据加密，现在有三种代表性的加密体制，即数据

加密标准 (DES)、公开密钥密码体制、数据库加密技术。数据库加密体制通常用来加密传输中的数据。不同的应用对密码体制的设计和实现都有不同的要求。有时,两种不同的应用对密码体制提出的要求往往是互相矛盾的,一种加密体制不可能用于所有的用途,所以在数据网中考虑密码体制时,首先要考虑到该体制的应用场合和计算要求。对于智能终端、电子邮政系统、数据库计算机和磁盘或通道接口来说,现在大多是采用微处理器来实现系统的加密。当考虑采用微处理器技术时,应当要考虑到与数据流相匹配的速度和功能。要有足够的实现密码算法的指令系统,足够的存储量,相当长的工作时间。还要考虑有相当规模的密钥存储器。

对数据网中的数据库进行加密时,采用具有错误扩散特性的密码体制比较合适。对于适合数据库加密的密码体制来说,甚至当只发生一个比特的错误时,处理机也能觉察出已经出错。因为在计算机网中,数据是从远程终端加密后传输给主处理机的,随后在处理机内脱密,再存入数据库。存储的数据也许很长时间未被查询,而原始数据源存入后已经消失。因此,当数据存入数据库时,数据中任何一个错误都应该纠正过来。因此,当使用序列密码加密时,如果只发生一个错误,处理器很可能觉察不出。因此,这种密码体制的密法算法对每一个比特的数据的加密,不但与密钥有关,而且与其它一些数据的比特有关。

数据网中传输或存储的数据都具有一定的结构形式,字符和字符串往往会重复出现,有时重复量很大,如果用常规的方法加密,明数据的结构格式很容易在加了密的数据中反映出来。这就给破译者提供了很有利的条件。熟悉该数据结构格式的人利用明密对应关系可作出较合理的推断,从而破开密码,获得明数据。当然,这种现象也可用密文链接技术来解决。

在一个大型的数据网中,署名和验证也是一个必须考虑的问题。所谓验证就是鉴别数据的真伪性和完整性。使用验证技术,可以防止数据内部分位的更改、插入,以及删除数据,并避免重新出现前面已发过的有效报文。

如果在一个数据库中存放的数据不是共享的,则必须采取防止被误用或被窃取的措施。办法之一可用通行字技术。但也可用加密的方法。假定一个数据库中存有 $n$ 个用户的数据,这 $n$ 个用户的数据由处理机统一加密,各用户都可以对各自的密钥进行存取或者修改各自的数据,但不能存取其他用户的数据。当修改各自的数据时,不需要对整个数据都加以脱密。具有这种功能的加密体制,由于实际的需要已经被提了出来,并引起了研究工作者的兴趣,逐渐被人们所重视。

### 三、关于密钥和密钥分配

在一个大型数据网中,若有 $n$ 个用户,且这个 $n$ 用户如能两两相互通信,则需要 $C_n^2 = n(n-1)/2$ 个不同密钥。而且为了安全起见,密钥又需要经常更换,所以对于大型数据网中,产生、分发和存储这样大量密钥的任务确实非常艰巨。我们知道,密码体制的保密性,在某种意义上来说是依赖于密钥的选择,也就是说真正保密的不是密码算法而是密钥。因此,密钥的产生,比加密算法的设计更加重要。要使产生的密钥对任何截收者都是不可预测的,则比较理想的是采用真正的随机序列发生器,并用硬件实现。而且对所产生的随机序列要严格的检验,以免不利情况的产生。

在数据网中,由于要加密的对象、部位、层次不同,因此对于加密的要求也不同,需要

采用不同的密钥以完成不同的任务。所以在数据网中，密钥的使用和管理十分繁杂，密钥的分配是极其重要的一环。

密钥分配分人工密钥分配和自动密钥分配两种方式。人工密钥分配也称实物密钥分配。自动密钥分配即电子密钥分配。

自动密钥分配采用数学编码方法、用电子方式统一产生分配密钥。自动密钥分配的硬件设备包括节点中心和终点控制器，节点中心也称密钥分配中心，它是整个密钥分配系统的中枢，其主要设备是密钥变量发生器。由于密钥的产生、分配和变换都是通过电子方式自动进行的，因此密钥分配即迅速又可靠，即简化了密钥的管理，又大大提高了通信的准确性和保密度。

密钥自动分配是用通信线路进行传输的。目前多采用自动分配为主，人工分配为辅的方式。这种方式又用两种方法实现：一种是密钥分层体制，也称主密钥体制；另一种方法是目前密码学界研究讨论得比较热烈的公开密钥分配体制。

密钥分层体制把密钥分为主密钥、辅助主密钥和会话密钥三种。主密钥是用来加密辅助主密钥的密钥；辅助主密钥是用来保护会话密钥的；会话密钥是对数据进行加密的密钥。会话密钥在通信时产生，在通信结束后消失。主密钥仍由人工分配，是更为保密的密钥。

密钥分层体制的实现要根据数据网中的结构而定。有的网有一个集中的密钥分配中心（如星形网），有的网则实行节点间的密钥分配。这种分层的方法适用于传统加密设备的通信网。

关于公开密钥密码体制分配密钥，目前有用RSA公布密钥密码体制来实现自动密钥分配的。美国Racal—Milgo公司生产的Datacrypto—Ⅰ、Ⅱ密码机就是用RSA公开密钥密码体制传输主密钥的。美国MITRE密钥分配体制是用公开密钥分配体制交换密钥，用DES对数据进行加密的。这种体制还有验证的功能。但密钥分配体制的计算比较复杂。保密强度尚需进一步研究。

文献出处：《计算机世界》1987年9月8日

## COPYWRITE 的威力

### 一、简介

为了保护软件系统，读者一般都会做其拷贝备份加以保存。但常常会遇上磁盘被加有保护措施（或称加密），所以，无法通过一般的拷贝程序进行复制，在这种情况下读者可以试着用COPYWRITE。当使用COPYWRITE时，它会将整张磁盘加以拷贝，所以，如果磁盘上原来具有保护，则拷贝出来的磁盘也具有保护。在使用COPYWRITE时，读者的微型计算机系统的内存必须大于128K。

如果读者使用的是最新的COPWRITE版本，可以用TYPE命令，将“COPYWRITE DOC”的文件内容显示在屏面上，以作参考。

## 二、如何使用COPYWRITE

在使用COPYWRITE之前，请先在你的原磁盘上贴上防写纸。并确定目前所使用的内定驱动器中是否有COPYWRITE.COM文件，如果有，则直接键入命令COPYWRITE(如果读者没有DOS，请参考这部分的1—4)。当读者已经看到COPYWRITE的功能表后，即可将磁盘抽出，换入其它的磁盘。如此，可以有较多的选择机会(可用的驱动器数较多)。

当读者在DOS下发出copywrite命令后，可以看到如下的功能表(见下页)：

COPYWRITE功能表的左半部分，说明了键(F4, enter, “↓”与“↑”“+”与“-”)键的功能，而右半部是命令选择与参数设定。F4键是将屏面换成彩色，如果是单色显示器，则这个键不起作用。通过光标上移键“↑”与下移键“↓”来选择命令的方式将◀标记移到特定的项目上后，按enter键，开始执命令。例如：假设◀目前指在Start copy项目上，而读者打算使用 Help命令，则可以连续按↓键，使◀会移到 Help上后，按enter键，即再看到Help的内容了。而“+”、“-”键是用来改变参数的。例如：读者只有一台驱动器，则来源磁盘与目的磁盘都必须设定为a:。因此，读者可以用“↑”，“↓”键将◀被记移到目的磁盘项目(target diskette)上，然后按“-”键就可以看到屏面上的目的磁盘参数变为a:了；或者，读者可以连续按三次“+”键，则目的磁盘参数依次变成c:，d:，a:，这样也可以达到目的。

由于参数的设定与读者使用的是二台驱动器，还是一台驱动器有关，所以，本文以这两种情形进行分别讨论：

Cyopwrite		function	argument
April 1986 edition		source diskette	a:
		target diskette	b:
427k bytes workspace		start copy	
		Exit	
		Hpel	
key	purpose	sides	2
F4	color	track range	0—41
enter	do function		
↓↑	select function		
+—	change argumet		
Copyright 1986			
Rober 7 McQuaid			

### 2—1、两台驱动器

如果，读者将来源磁盘放在驱动器 a: 中，而将目的磁盘放在 b: 驱动器中，则读者可以直接将光标(◀)移到开始拷贝(start copy)一项上，按enter键后开始拷贝。当拷

贝结束后,你就可以再看到COPYWRITE的功能表。

在一些特殊的情况下,读者可以使用“↑”,“↓”键,再加上“+”,“-”键来改变来源磁盘与目的磁盘的设定参数。如果,读者设定来源磁盘与目的磁盘成相同的参数,那么COPYWRITE的使用程序就变成一台驱动器的使用程序了。

## 2—2、一台驱动器

当读者使用一台驱动器做拷贝时,COPYWRITE会先读入整个来源磁盘,接着,要求读者换入目的磁盘,然后,开始写入,也许读者要作多个备份,磁盘也就会被换来换去多次,为了防止错误发生,应先将来源磁盘贴上防写纸。

当参数设定好以后,选择“开始拷贝”一项,按enter键。COPYWRITE系统会显示下列信息。

```
I need the soure diskette in drive a;  
please push the S key when it is ready
```

所谓soure diskette指的是来源磁盘。将它放入驱动器中,关上驱动器的门,并按“S”键。系统会将整张磁盘的内容读入(这需要一些时间),然后,它会再问:

```
I need the target diskette in drive a;  
please push the T key when it is ready
```

target diskette指的是读者要拷贝的空白磁盘。从驱动器中拿出来源磁盘,放入空白(目的)磁盘,再按“T”键,这时系统开始将资料写入到空白磁盘上。

当COPYWRITE拷贝完整张磁盘后,就返回功能表。

注:如果读者的微型计算机内存比较小或常驻内存的程序较大(如用软汉字CC DOS 2. XX),那么读和写可能要交替进行几次才能拷贝完整张磁盘。

## 2—3、其它参数

通常读者必须将磁盘的面数设为双面(2),如果是单面的磁盘为了节省时间,也可以将此参数设为1。

磁道范围参数是对半离驱动器或者其它特殊的驱动器才有用。有些驱动器的磁头可以移动到第43道上,也有些软件产品在第43道上做了保护工作。而大部分的驱动器磁头不能移到第43道,若采用这样的驱动器,在磁头移到第43道前,读者就会听到撞击声,如果有这种情况发生,读者必须将磁道范围参数设小一点,以免发出撞击声。

## 2—4、使用COPYWRITE开机

读者在使用COPYWRITE时,可直接将COPYWRITE磁盘放入驱动器a:中开机(要先建立含有COPYWRIT的批处理文件)。当功能表出现在屏面上后,读者即可将COPYWRITE磁盘抽出,接下来的程序与前面相同。

# 三、拷贝保护与非法复制

多软件公司为了保护自己的产品常常在磁盘上加有保护,以免被非法复制,不过在防止非法复制的同时,也阻碍了合法拷贝,就是使用一个很好的拷贝程序,也显得无能为力。而



新版的COPYWRITE在这方面取得了极大的进展。

COPYWRITE能够帮助读者保护自己的软件系统。

为了有效地控制非法复制，本文建议，在硬件上做工作，如在每套微型计算机系统中设定特殊的识别码，它可以由软件读出并核对。这是目前最有效控制非法复制的方法，同时，也允许对该产品做无限的拷贝。

#### 四、COPYWRITE的功能

COPYWRITE可以使用一台或二台驱动器工作，并使用所有的可用内存空间。不过，COPYWRITE不能使用硬盘驱动器和虚拟驱动器（RAMdisks）。通常制作一份拷贝约需一分半钟到十分钟的时间，这是依据磁盘格式制作的方式而定。

COPYWRITE能够在IBM PC，长城0520及其兼容机上执行。为了得到好的拷贝结果，读者必须用好的空白磁盘做为目的磁盘。COPYWRITE不会检查目的磁盘中是否有坏的磁道。而读者在使用COPYWRITE时，空白的磁盘不必是制作好格式的磁盘，如果目的磁盘尚未制作格式，COPYWRITE会在拷贝前先制作格式。

COPYWRITE并不如DISKCOPY可靠，因为发生错误时它并不会停止拷贝。COPYWRITE是从来源磁盘中读到什么，就将什么写到目的磁盘上。因此，读者可能要尝试多次才能得到一份好的拷贝。

COPYWRITE有时会在拷贝完成后传回一份报告。这份报告并不是告诉读者错误的信息，而是说明来源磁盘的保护方式。接下来本文将对一些传回来的信息做一些解释。每一个磁道都由许多扇区组成，而每个扇区中的资料又分成识别栏与资料区。

下面是COPYWRITE可能传回的信息：

##### 4-1、bad CRC

在每张磁盘上都有CRC记录（CRC是cyclic redundancy check，又称循环冗余检查码）。所谓CRC是将正确的资料加入到一些位中，作为检查之用。当系统检查资料，发现不符合码时，即此处有错误，系统就会告诉使用者。

不过，有些磁盘的保护方法，采取有意地制造CRC错误，而COPYWRITE会无误地将它拷贝过来。

##### 4-2、duplicate id

duplicate id表示COPYWRITE发现磁盘中有一个以上的扇区，具有相同的识别栏值。如果这种情形有很多的话，则拷贝的时间较平常多花一些。

##### 4-3、deleted data

deleted data代表磁盘中某个扇区的资料被标有某种记号的码，并不表示有资料遗失。同样地，这种情形下拷贝会多花一些时间

##### 4-4、no data address mark

no data address mark表示在来源磁盘中有一个错误，或者是一种磁盘保护法。

##### 4-5、Verification

在写入目的磁盘的手续完成后，COPYWRITE会再从目的磁盘中读入资料并与来源磁盘比较，看是否相同。所谓的verification错误是表示COPYWRITE读回来的资料与写入的

资料不符合。你可以尝试重新拷贝一次，或换几张好的目的磁盘，或者换台不同型号的驱动器，甚至换一部微型计算机系统。如果这一错误在不同的目的磁盘上连续发生，那么，很可能是遇到了另一种新的保护方法。

#### 4—6、put a write protect tab on the target disk

有部分公司的磁盘保护法是程序一开始执行后，就尝试将资料写到装入的磁盘中。如果能写入，表示磁盘不合规格。当COPYWRITE发现磁盘的保护法是属于这种类型时，则发出相应信息，告诉读者：在拷贝软件前，先在磁盘上贴好防写纸。

#### 4—7、laster hole

laster hole信号发生在来源磁盘是Prolok磁盘。如何应付这种情形，请参考第五部分。虽然有些系统没有采用Prolok保护法，但也会产生这种信号，如Memory/shift对这种系统而言，这个信号可以忽视。

#### 4—8、I have encountered something I can not handle

发出这种信号的情况有两种：一是来源磁盘是由某种新的保护方法保护着，二是在COPYWRITE从来源磁盘读入资料到COPYWRITE写资料入的磁盘的这段时间内，内存的内容被改变了。试着用其它的微型计算机，或者将COPYWRITE装入到其它的位置。

## 五、诊 断

如果读者使用COPYWRITE做备份，经常出现失败，即使是COPYWRITE能够拷贝的软件，请先做下面一些事情。

首先，请读者检查操作方法是否正确，如果在拷贝DOS2.0磁盘时，发生了错误，则表示读者操作COPYWRITE的方法有错误。例如，将磁盘面参数设为单面等等。

当读者已经成功地使用COPYWRITE拷贝了许多磁盘之后，应该知道COPYWRITE传回信息所代表的涵义。若读者在拷贝完一张磁盘后，看到COPYWRITE显示出一些不寻常的信息，则不要使用这份拷贝。

由于COPYWRITE不能够容忍目的磁盘上有错误。因此，如果COPYWRITE核对出错(Verification)，那么请读者换一张好的空白磁盘，重新拷贝一次。

有些游戏程序不能在内存超过512k的微型计算机上执行，因此，读者必须在完成拷贝后，执行程序前，用微型计算机内部的调节开关将内存的容量设小。

有些软件产品是分散在多张磁盘上，并且，磁盘上没有贴防写纸的缺口。一旦程序开始执行，程序就会写资料到装入的磁盘上，如果能够顺利写入资料，则表示磁盘是非法的。所以，读者必须在执行程序前，先贴上防写纸。

COPYWRITE在作拷贝时，用的是软盘驱动器控制系统所提供的涵数与软盘驱动器，所以，若使用的驱动器有问题，COPYWRITE就不能正常工作；若作用两台驱动器可以分别作单驱动器的拷贝工作，选择出较好的一台驱动器，完成COPYWRITE拷贝工作；或者读者可以在其它的微型计算机上试试。

Lightning Software的Master Type在制作拷贝时会发出：22个扇区没有资料位置标记的信号。在拷贝这个软件时，读者最好将磁盘的面参数设为1。

Inssoft的My... 还是整卷面软盘... 不过使用反面的某些扇区。如果在拷贝时没有困难的话,不妨将磁盘面数设为1,重新拷贝。

Infocom 所发展出来的一系列游戏软件如: Star\_Cross和Dead... 不能在内存容量超过512k的微型计算机上执行;因此,在执行这类软件前,应先将内存容量调小。

Lotus Symphony 套装软件在拷贝后,会传回2个扇区有bad sectors和data address marks与23个扇区有bad CRC的信息。

Davison Math Blaster 与 Speed Reader II 并没有拷贝保护;因此,直接使用DQS拷贝即可。不过,它们要求在内存驱动器内的磁盘上必须贴有防写纸,否则,就会停止程序的执行。

## 六、COPYWRITE与COPY II PC的比较

此处所比较的版本均是目前国内所能见到的较新的版本,它们分别为COPYWRITE一九八六年四月版和COPY II PC Ver3.05(3.0x),两者推出的时间十分接近,因此较适合作比较。

下面分成二点,详细地为你说明彼此的差别:

### 6-1 拷贝磁盘威力的大小

最新版本的COPYWRITE和COPY II PC均能直接拷贝SOFTGUARD Ver2.03所保护的磁盘,所得备份磁盘与原版磁盘完全相同,因此,你可以直接地成功地拷贝诸如dbase II PLUS、Framework II、Lotus 1-2-3 Release 2、Paradox Ver1.0或2.0等软件。至于以SOFTGUARD Ver2.00所保护的磁盘,自然更不在话下了,你可以轻易地复制出Symphonig Ver1.1、Framework Ver1.1、dbase II Ver1.1、Better BASIC Ver1.00与Double DQS Ver1.00等软件。不过以这种方式所拷贝下来的磁盘,仍然留有原来的保护,你也可以选择彻底破解SOFTGUARD方法的保护,具体采用何种方式,视你的需要而定。目前,COPY II PC对PROLOK所保护的磁盘完全无能为力,而COPYWRITE则可以成功地对付很简单的PROLOK激光保护。目前国内也能买到打了激光孔的PROLOK磁盘(每张的价格大约30元左右),如果你手上刚好有一张早期的PROLOK磁盘的话,你可以试着用PROLOK保护某些文件,然后,再利用COPYWRITE制做复本。在执行复本前,请先运行RAMKEY程序。

至于拷贝其它程序的能力,原则上COPYWRITE较优于COPY II PC,例如COPYWRITE可以直接地成功地拷贝KING QUEST I,旧版的Ancient Art of War等游戏软件,而COPY II PC则无此能力COPYWRITE的威力对于拷贝游戏程序特别出色。

### 6-2 拷贝磁盘速度的快慢

用过COPYWRITE的读者一般都有这种体会,COPYWRITE的拷贝速度有时候慢得让人难以忍受,特别对于有经验的使用者而言,这种现象是无法原谅的。因为,同样拷贝一张Ancient Art of War游戏磁盘,如果使用COPY II PC OPTION BOARD可能只需要到不三分钟的时间,但是,如果使用COPYWRITE,可能就要花十多分钟了。因此,如果

你只有COPYWRITE和COPY PC，则在拷贝磁盘时，建议你先采用COPY PC试试。如果不行，再使用COPYWRITE，即一般情况下，尽量避免使用COPYWRITE。COPY PC的拷贝速度较让人满意，而且，一旦遇到困难状况时，它会较“聪明地跳过”，而不会象COPYWRITE一样与磁盘上的某处纠缠不清，而耗费过多的时间。

比较完COPY PC和COPYWRITE这二个拷贝程序的差别后，最后，为你叙述它们的共同弱点。通过下面列出的简表，指出它们所无法拷贝成功的软件，究竟包括哪些程序。

Electronic Arts所推出的一系列游戏程序，其中多半是由Apple所转换过来的，这些游戏采用了一些特殊的防拷保护措施，其中包括：

Archon  
Hit Hack Mack  
Music Construction Set  
Murder of Zindernuft  
One On One  
Seven Cities of Gold

这些游戏必须使用COPY PC OPTION BOARD来拷贝，并且最好把Keep track length（保留磁道长度）这个参数改成Y，麻烦的是这些游戏往往会挑驱动器，所以，如果驱动器匹配不当的话，你可能仍然无法成功地拷贝出这些游戏，其中以Music Construction Set这张游戏磁盘最为挑剔。

至于其它无法拷贝的软件还包括：

新版的Ancient Art of War  
Silent Service

以及国内许多单位研制成功的中文操作系统和汉化软件，因为，这些软件纷纷采用了一些特殊的保护措施，以确保他们的权利。

文献出处：《IBM PC实用COPY指南》 222—230页

## 如何去掉 PROLOK 的保护

### 一、激光加密盘简介

激光加密盘(PROLOK)是一张5 1/4标准格式化的软盘，它在软盘上(一般在0面)打了一个或数个激光孔，即破坏了激光孔所在磁道的磁性材料正常的记录资料功能被加密的文件在运行时会寻找被定义的激光孔，若激光孔存在，则将密码解开执行程序，若PROLOK加密的文件找不到所定义的激光孔则在屏面上显示：

Please insert Key diskette and try again

然后返回操作系统（PROLOK加密的文件会自动寻找在任何驱动器上是否具有原版的

PROLOK磁盘)。

PROLOK要求读者微型计算机系统的内存至少有128K,使用的MS/PC DOS的版本在1.1以上,我们建议读者最好不要使用1.xx的版本。PROLOK磁盘能够加密的程序的扩展名仅为.EXE和.COM。一旦程序被PROLOK加密后,被加密的文件将比原文件增大6~10K,这6~10K的内容就是PROLOK文件执行中对源文件加的密。每当执行被加密文件这6~10K的内容将引导它首先寻软盘上的指纹(激光孔),并将其与加密文件内所记录的原始指纹比较,如果相同则执行这个被加密的文件,否则退回到DOS的指示符下,从而实现阻止非法复制文件的目的。

PROLOK的另一种用法是在计算机网络的磁盘中存入被加密的信息,只有密码盘的拥有者才能使用这些信息,达到在共享资源的计算机网络系统中某些资源不共享的要求。

## 二、巧破激光加密盘

破解激光加密盘从密码学的角度来讲要分析出加密程序的加密算法和密钥。PROLOK的加密算法比较简单,不同的激光加密盘有不同的密钥。这涉及到了一些专门的知识,关于这方面的内容有兴趣的读者可以参看有关书籍。

下面我们以一张原版的激光加密盘DBASE III 1.00为例介绍如何去掉PROLOK的保护。在完成去掉PROLOK的保护后的程序约会缩小6~10K。这种方法只限于寻找扩展名为.EXE文件的开头,即原版程序在应用PROLOK之前,必须有一个像DBASE.EXE这类文件。

读者应当特别注意,目前正在使用的激光加密盘种类很多这里所给的地址或数字只是一个示范,这些地址和数字会随着被解密文件和使用磁盘的不同而有所改变。我们的目的是为读者提供解决问题的思路,如果读者熟悉IBM PC的基本原理和中央处理器8088并对汇编语言、DOS的内部结构和DEBUG程序有较多的经验。那么以此为借鉴,便可破解多种激光加密盘。

屏面中需要由读者键入的命令为大写字母。

```
DEBUG DBASE.EXE
R <CR>
  example:BX=0001      CX=750
U <CR>
  example:xxxx:0167 LOOP 0160
                xxxx:0169 XCHG DX,AX
G 169 <CR>
S 100 3000 83 C4 08 <CR>
  example:xxxx:0746
                xxxx:13C4
A 13C4 <CR>
XOR AX, AX <CR>
```

```

RET < CR >
< CR >
S 100 3000 C0 45 F8 < CR >
    example:xxxx:1140
E 1140 0B 46 < CP >
M 0:C F 0:200 < CR >
G < CR >
S 100 3000 4D 5A
    example:xxxx:03D6
        xxxx:06E0
        .
        .
        xxxx:1B50
example    1 C750
          - 1B50
          -----
          1 AC00
H C750 1B5a < CR >
    example E2A0 AC00
RBX < CR >
:0001 < CR >
RCX < CR >
:AC00 < CR >
N DBASE < CR >
W 1B50 < CR >
Q < CR >
REN DBASE DB.EXE

```

整个破解过程修改的含义如下：

用DEBUG把DBASE.EXE文件装入内存。最前面的循环可将PROLOK的密码解开，在此循环结束后我们使用一个GO命令，然后搜寻“83 C4 08”这个数组，以找出其特定的机器码，此机器码当PROLOK无法找到它所需要的特殊指纹磁盘（激光打孔磁盘）时，即发出错误信息。我们在那里传送一个“0”值到AX寄存器内的地址上汇编一段新的机器码。这可使PROLOK认为它所需要的磁盘找到了，然后它就将原版的.EXE文件解开密码。搜寻“C0 45 F8”的目的是要找出装入原版的.EXE机器码，由于PROLOK会破坏我们的BPT向量，因此我们不可以在那里置一个中断点。所以我们在此地址上输入一个“INT 80”的机器码，并将我用的BPT向量拷贝到尚未使用的“INT 80”向量上。读者可能已经注意到这个机器码已经“编上密码”故我用的“INT 80”就成为了“0B 46”。“Move 0:CF0:200”命令可将DEBUG的中断向量拷贝到“INT80”向量以供该处使用。然后我应执行一个GO命令，PROLOK就检查驱动器A中是否为PROLOK磁盘，当它误认为已经

找到时就将原版的.EXE解开密码。在它将要执行该文件前会遇到我们的“INT80而返回到DEBUG中。

最后我们搜寻.EXE文件的开头，计算一下文件除去PROLOK机器码后还有多长。然后将它写为一个文件，在DOS状态下将该文件的扩展名改为.EXE就可以了。

### 三、解密程序UNGUARD.COM介绍

也许读者认为用DEBUG去掉PROLOK保护的步骤较多或通用性差。我们为读者介绍一个UNGUARD程序。UNGUARD.COM是目前解密能力较强的一个实用程序。利用这个程序可以破解较多的用PROLOK Ver 2.01保护的磁盘。

使用方法如下：

<pre>March 1988  keys purpose   F4 color   ↑ ↓ select function   + - change drive   letter file name   Enter do function  copyright 196 Robert T McQuaid</pre>	<pre>function option  Source drive ◀ C: Source file target drive C:  Run source  Help Exit to DOS</pre>
--	---

在DOS提示符下键入UNGUARD。程序执行后会在屏面上显示出一个选择清单。其中FA键：切换彩色/单色的屏面显示方式。

↓或↑键：改变所选择的项目。

+或-键：更改驱动器的名称。

ENTER键：执行该项功能。

请把原始磁盘放入原始驱动器，已格式化的目的磁盘放入目的驱动器，键入原始文件的

名称，将光标移到“Run Source”选项并按下“ENTER”键，UNGUARD会自动从原始磁盘读入指定的文件经过解密后会自动地写入目的磁盘。

UNGUARD程序不仅可以去掉激光加密盘的保护，而且也可以拷贝某些用SOFTGUARD加密的磁盘，有兴趣的读者可以试一下。

文献出处：《IBM PC实用COPY指南》292—295页

## 如何破解SOFTGUARD2.00版的保护

被softguard系统保护的程序，可以从隐藏在主目录中的文件CML0200.HCL和VDF0200.VDW来判断此程序是否加上了softguard的“锁”，您将程序安装到硬盘驱动器中时便可知。文件名中的“0200”是softguard的版本代号（2.00），CML表示Common Loader，VDF表示Volume Descriptor File。另外，还有一个扩展名为EXE或LOD的基本文件被隐藏起来，这便是真正的程序但已被编上密码了。

在执行DBASE时，程序DBASE.COM将CML0200.HCL调入内存并执行。CML解开本身的密码并读取VDF0200.VDW文件。VDF文件含有一些在安装时硬盘驱动器FAT的资料。将VDF文件内的资料与目前的FAT比较后，CML就可以判断CML、VDF和DBASE.EXE文件是否在其所安装磁盘的相同位置上。如果它们被移动过，即表示该磁盘是复制的或是被重存过的，则DBASE就无法执行。

本节是指导您如何破解任何利用softguard系统保护的程序。我们将以DBASE III 1.10版为例，说明如何绕过softguard2.00的防拷保护。若您要将这种方法用到其它有Softguard2.00防拷保护的磁盘上时，要更改下面给定的数值，修改文件名和BX: CX寄存器中所记录的长度即可。

首先，利用原版的DBASE III 磁盘，将它安装到硬盘驱动器中，softguard会将3个文件（CML0200.HCL，VDF0200.VDW，DBASE.EXE）隐藏在你的硬盘驱动器的主目录中。它同时也将DBASE.COM拷贝到你选定的DBASE目录内，而DBASE.EXE是真正的DBASE III 程序，它已被编上了密码，此文件也可能被命名为DBASE.LOD，但是作用完全相同。

其次，把主目录中的3个文件更改为一般文件，将这3个文件和DBASE.COM拷贝到其它的目录中，然后将这3个文件隐藏起来。

按照DBASE的指示，解除安装DBASE III，然后您就可以把原版的DBASE III 磁盘拿开。此时，我们已完成需要用到这张磁盘的所有工作。

接下来我们要对CML0200.HCL做一些修补工作，以便我们能在DEBUG中跟踪整个程序。这些修补工作可以防止CML0200.HCL破坏我们所设定的中断向量。

将CML0200.HCL，VDF0200.VDW，DBASE.EXE和DBASE.COM四个文件拷贝回主目录中，并将前三个文件隐藏起来。

现在我们可以用DEBUG来运行DBASE.COM，一直跟踪到它将DBASE.EXE的密码



```

DEBUG CML 0 2 0 0 .HOL
E 3F9 <CR>    2A.4A <CR>
E 49D <CR>    F6.16 <DR>

E 506 <CR>    E9.09 <CR>
E A79 <CR>    00.20 <CR>
E AE9 <CR>    00.20 <CR>

E 73C    97 FA FA F4 F1 7E <CR>

```

W

Q

```

DEBUG DBASE.COM
R <CR>

```

解开为止。

下面使用的DS值仅供参考：

```

A 0:300 <CR>                                ; we must assemble some
                                                code here

    pop     ax
    cs:
    mov     [320],ax    ; save return address
    pop     ax
    cs:
    mov     [322],ax
    push    es          ; set up stack the way
                        we need it

    mov     ax,20
    mov     es,ax
    mov     ax,0
    cs:
    jmp     far ptr [320] ; jump to our return address

<CR>
G 406
T
G 177                                         ; now we can trace CML

```

```

< CR > ; this stuff just tracesess
O 406 ; now we can trace CML
T
G 177 ; this stuff just traces
           past some
G 1E9 ; encryption routines.
T
G 54E ; wait while reading VDF
           & FAT

G=559 569
G=571 857 ; DBASE.EXE has been
           decrypted

```

键入DBASE Ⅱ 1.10的文件长度：

```

RBX < CR >
: 1
RCX < CR >
: AC00

```

键入文件名：

```

NDBASE.bin ; name of file to write to
W XXXX, 100 ; where XXXX is the value of DS that
           ; you wrote down at the begining
Q ; quit debug

```

然后，将CML0200.HCL，VDF0200.VDW和DBASE.EXE改为一般文件并将它们和DBASE.COM全都删除，把DBASE.BIN重新命名为DBASE.EXE。这时破解便完成了，新的DBASE.EXE可以拷贝，在运行时只需要DBASE.DVL文件。

以上就是破解Softguard的方法，有些磁盘因版本不同，Softguard应用的方法和地址不同，可能您要多费一些时间才能破解，不过您只要了解了Softguard的检查保护原理，再配合本文所介绍的破解要领，再解开所有的Softguard保护并不困难。

文献出处：《IBM PC实用COPY指南》288—291页

# “坏”软盘起死回生

浙江 张晋新

IBM—PC微机用户可能会发现，当使用FORMAT命令对一张旧软盘进行格式化，有时会指出盘上有许多坏块，或者干脆出现0磁道坏，格式化失败的提示信息。碰到这种情况用户往往以为是软盘质量不好，这些坏块是原先就有的，或者是使用不当和磨损造成的。可是我在使用中发现有时事实并非如此，特别是对那些采用COPYWRIT一类的拷贝软件复制的加密盘和游戏盘。

我曾做过试验，将一张新软盘先用FORMAT命令格式化发现是完好的，然后用COPYWRIT复制成一张游戏盘，经使用复制的新游戏盘也是好的。紧接着再用FORMAT命令对该盘格式化一次，令人奇怪的是此时很可能会出现许多坏块或0磁道坏（视游戏盘不同而异），连续换了数张新盘情况依然如此。显然这位多盘不可能经一次拷贝之后都坏了，而且将那些0磁道坏不能格式化的游戏盘放在机器上又都是可以正常启动进入游戏的。于是我想很可能是这些加密盘使FORMAT命令不能正常运行造成的。针对这一情况，我又格式化了一张完好的新盘，然后将这张新的空盘作为源盘用COPYWRIT命令拷贝到那些有问题的“坏盘”上，结果这些“坏盘”又都复原了，重新用FORMAT命令格式化也都一切正常。通过这个办法，我使手头积累的许多“坏盘”起死回生了。

用户们如发现同样问题，也不妨按上述方法一试。有时用COPYWRIT命令不成功可换用其它如COPY5PC等再试，如还是不成功则软盘就确实坏了。

文献出处：《软件报》1987年年11月16日

## PROLOK激光加密系统分析与解密方法

郑州电子技术学院 刘亚南

### 一、PROLOK系统分析

PROLOK保护系统由一张PROLOK软盘组成，每一个软盘上有一用于保存文件密码的唯一指纹（随机打上的激光孔）和一PROLOK.EXE程序。一旦用户程序被PROLOK.EXE程序加密，以后每当运行该加密后的用户程序时，它首先查寻软件上指纹并将其与加密程序内记载的原始指纹比较，如相同则运行用户程序，否则退回系统，用户也可以在用户程序内任何地方插入检查指较的代码。由于程序的执行离不开软件上的物理指纹，而该指纹不随盘的复制而复制，因此达到了保护作用。

我们知道，程序分析手段大都是①反汇编程序清单脱机分析与使用动态调试程序跟踪联机分析相结合。为防止用户分析找出判断指纹的程序段并跳过它使指纹作用无效，PROLOK系统首先采用许多手段来对付反汇编与动态跟踪，下面分析介绍这些措施：

1. 反反汇编：通常程序装入内存执行时不再改变，因此用户可用DEBUG程序装入程序，然后用反汇编命令得到完整的程序清单。PROLOK对付反汇编的方法，是程序逐块加密，在执行过程逐块解密、执行、加密。这样，任何时刻内存中都不存在完整的程序，只有当前执行块可见。加密方法主要采用程序代码逐字节异或（ $A \oplus B = C$ 加密， $C \oplus B = A$ 还原）及程序上下搬移。

2. 反动态跟踪：为防止用户使用DEBUG调试程序跟踪分析程序，PROLOK采取了以下几个措施：

a. 破坏断点与单步中断向量：程序的跟踪是通过断点与单步中断实现的，PROLOK程序把中断向量区中0000:0004, 0000:000C等用作程序的数据区，程序在执行过程中经常对这几个单元存取，因此破坏了单步与断点中断向量。为了防止用户把该数据区移向别处，程序中大量地使用这类操作，使用户在跟踪时改不胜改。为了解决这个问题，我们修改了DEBUG程序，只要找到DEBUG的T命令，G命令处理部分，修改几条指令，就可把原先的1号（单步），3号（断点）中断改用我们选择的30号中断。

b. 隐蔽转移：为了不让用户发现程序的调用走向，PROLOK采用与JMP、CALL等显式调用不同的隐式调用。如INTO调用，当加法溢出时，转向溢出处理程序，而这时的溢出处理程序已由加密程序取代。还有一种隐式调用为单步中断调用。在中断处理返回时，要恢复标志寄存器，如在程序中把将要恢复的标志字节中单步标志位置位，程序在返回前就要首先进入单步中断处理程序，这时的单步中断处理程序也已被加密程序取代。往往这种类型的隐式调用使生手不知所措，但在这里一点即破。

c. 大循环、多出口：PROLOK采用大循环措施拖垮跟踪者，在PROLOK程序块中，有时一大段程序故意多次循环（一般四十五次），在这些循环中，出口有许多个，并且许多出口的转向由程序动态设置，这样，跟踪者就无法事先决定程序走向，正确设置断点。要解决这个问题，我们在循环开始每次必经之地插入一段程序计数，这样就可知循环次数，然后就只要跟踪最后一次循环，找到程序的出口。

d. 分块：前面在反反汇编中提到程序的分块加解密执行，它还有一附加作用是防止设置断点。因为下一块程序在执行前要解密，因此太早设置断点，插入的断点指令经过解密操作变为不正确的指令。

e. 设置堆栈：PROLOK程序有意把堆栈设置在内存高端（小地址端），这样一方面破坏中断向量区，另一方面可破坏跟踪。在无限跟踪情况下堆栈大小可满足程序调用进出栈操作，如在此期间设置了断点和单步，情况就发生变化。举例说明如下：设堆栈在0000:0008处，当执行调用CALL  $\times \times \times \times : \times \times \times \times$ 时，压入堆栈返回地址共4字节（CS:IP），这时堆栈为0000:0004，如在这以后设置断点，发断点中断时，系统压入断点返回地址与标志寄存器共6字节，这时堆栈上卷入地址低端（大地址），这是因为PC机内存是循环组织的。在低端一般无内存或为EPROM，造成返回地址丢失，系统混乱。

以上我们列举介绍了PROLOK防止反汇编与动态跟踪的几个措施。接下来，我们将介绍PROLOK指纹判断方法与解除方法。

指纹，是指激光孔在盘上位置，该位置是随机的，随盘不同。激光孔的位置信息装载在经过加密的应用程序中，只有判定了盘上激光孔位置与程序中装载的相符，才可执行程序，因此程序离不开指纹盘。PROLOK判定程序中装载的激光孔位置指定的扇区是否有激光孔的原理如图：

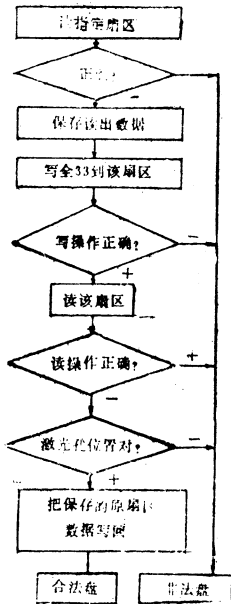


图 1

通过写全33再读回得到的激光孔位置如下图，我们可以看到两个孔。

前面我们给出的判合法盘过程只是原理性的，真正的判定过程非常混乱复杂以防用户理解。

通过分析指纹判定过程我们发现，PROLOK2.00版在××××.16C9处调用了判定过程，接下来判断判定结果，在有合法盘情况下转向正确程序段执行，即成功转，那么在没有合法盘下一定判定不成功，我们也要让它转向正确程序段执行，因此我们把原来的JZ指令改为JNZ指令即可化解指纹作用。进一步，我们不要让程序读盘，我们找到读盘子程序用返目语句与置盘成败标志替代。修改后程序可在任一驱动器上运行并不进行盘操作。

到此，我们已可解除PROLOK的指纹保护，但以上步骤是通过调试程序完成的，而真正的PROLOK程序不允许修改指令（因为指令在执行前先要进行解密操作），下节我们将介绍如何使PROLOK程序不用调试程序运行。

5020:0000	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0010	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0010	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0030	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0040	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0050	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0060	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0070	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0080	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0090	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:00A0	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:00B0	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:00C0	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:00D0	33	3A	00	03	83	12	08	C2-10	43	14	91	0A	21	10	91
5020:00E0	03	28	08	C8	40	41	33	33-33	33	33	33	33	33	33	33
5020:00F0	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0100	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0110	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33
5020:0120	33	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33

5020 : 0130	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	
5020 : 0140	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	
5020 : 0150	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	
5020 : 0160	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	
5020 : 0170	00	18	00	14	D4	04	0A	86-04	00	19	01	26	89	69	43
5020 : 0180	21	67	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 0190	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 01A0	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 01B0	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 01C0	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 01D0	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 01E0	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33
5020 : 01F0	33	33	33	33	33	33	33	33-33	33	33	33	33	33	33	33

## 二、一种解密方法

要解除PROLOK的保护，可采用两种方法。一种在跟踪执行到真正的应用程序时把应用程序从内存写盘，丢掉前面的指纹判定部分。另一种修改程序，如前节提到的JZ改为JNZ。前种方法人为地去掉一大段程序，破坏程序完整性，因此不大可靠。我们选择第二种方法，这种方法的困难之处在于我们前面提过的程序分块加解密执行（程序块的累加和有时还作解密密钥），因此无法事先修改，只有在执行到一定时机（解密后）才可修改。因此，我们设计了一种方法模拟调试程序，举例说明如下：

前面提过16CCH处的JNZ要改为JZ，但该处指令只有在执行到165C处才被解密可修改，因此我们要在165C处中断一下以在16CC处修改指令，这是通过在165C处插入INT30H指令实现的，而在165C处插入指令要在27F0时才行，依次类推27F0处要在253A时才可修改，253A要在385时才可修改。这样我们设计INT 30H模拟程序逐步引导程序执行，见图2

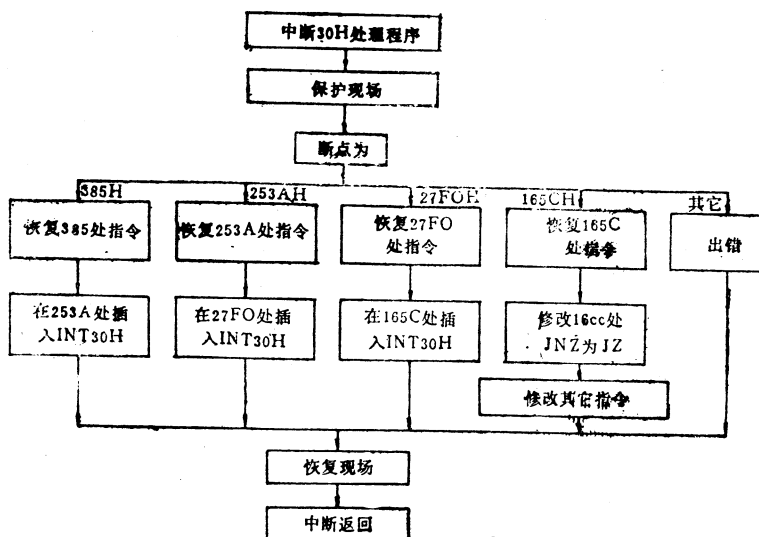


图 2

使用以上解密方法，我们几乎不要修改加密程序，只要在每次系统启动时把INT 30H. 驻留内存，这样即保证了程序的完整又简化解密方法，经过使用效果良好。

```
N1=10
N2=10
M1=70
M2=80
KC= 2
I= 3
CALL CLS ( KC )
CALL LINE ( N1, N1, M1, M2, I )
END
```

### 三、结 语

该绘图软件库目前只编写了最常用最基本的几个功能子程序，它为用户使用FORTRAN绘图提供了非常方便的手段。进一步的打算，可以扩充如下功能：绘椭圆（包括圆）；写字符串；制作光标等。用户许多在BASIC语言中能实现的功能都可在FORTRAN语言实现了。

文献出处：《微型电脑》1988年1期8—10转35页

## 谈谈加密软磁盘的解密与拷贝

蚌埠 季平

目前流行的加密软件分两大类，即硬加密和软加密。硬加密就是用硬件的方法在软盘上做某种记号。这种永久性不可恢复的记号是不可复制的。激光加密就属硬加密。软加密是用软件工具在软盘上产生特殊的格式化数据作为检测记号。这种记号是可以恢复但不可复制的。两种加密盘都有自己的索引程序，索引程序根据盘上有无记号来判断此盘是否是复制品。

各类加密软件属激光加密最难拷贝。所谓激光加密就是在格式化的同时，用激光在数据区或扇区标识符上烧若干个痕迹，使磁盘的某几点失去磁性，软盘工作在以上区域必然产生CRC错误，但加密软盘都有自己的索引程序，当判断有CRC错时它认为是对的，如没有CRC错它认为此软盘是复制品，拒绝从磁道上读出程序。问题是一般的磁盘控制器下DC是不能够往磁道上“错写”信息的，所以激光加密盘只能用解密的方法来得到复制品，而软加密的磁盘用特殊的拷贝方法就可以复制。下面简单的谈一下这两种方法。

## 一、解密的方法

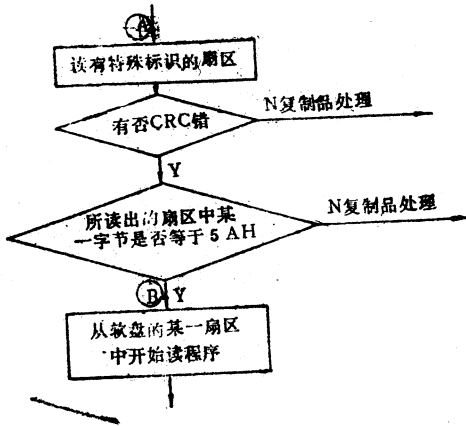
因为无论怎样的加密，最终还是要调出程序并执行的。我们紧紧跟踪索引程序的运行，并修改索引程序，定能将加密程序所设置的障碍解除。解密一般分四步进行

1. 用COPYWRIT·EXE拷贝工具拷贝一张复制品做为工作盘。此时的工作盘，已将程序拷上了，只是标记没有拷贝过来。

2. 工作盘上有一个隐含的AUTOEXEC·BAT文件，用TYPE命令打印一份并接文件的内容一步步执行，目的是找出索引文件。当执行到索引文件时机器会出现不正常的现象或死掉，或者用COPY命令将AUTOEXEC·BAT中所有列的文件一个个拷贝一遍。当碰到哪个文件拷贝出错时，那个文件就是索引文件。

3. 索引程序一开始的内容就是破坏机器的单步中断。我们修改之，使在DEBUG状态下可以跟踪索引程序的执行。

4. 修改索引程序。此部份是加密软件的核心，下面是索引程序的一部份担框，借此来举一反三。



很明显只要修改索引程序使之从A跳到B即可。

只要从索引程序中找到类似于以上判断特殊标识的程序段并将其修改，那么经修改后的软盘就是已解密的用DOS命令可以拷贝的软件了。

解密的过程是一个十分复杂的过程，必须在DEBUG状态下紧紧跟踪索引程序的运行并一步步修改，才能摸到判断特殊标识的部份。

## 二、拷贝的方法

所谓拷贝的方法就是利用各种型号软盘机转速、磁头步进电机的差别，还有各种拷贝工具功能的差别来碰。笔者用以下的方法拷贝了一张加密软磁盘获得成功。

1. 先用DOS命令DISKCOPY A: B: 拷贝一遍。A是源盘，B是目的盘。

2. 再用拷贝工具BACKUP 4 A: B: 连续拷贝二遍。A是源盘，B是上一次那张目的盘。

这样就得到了一张连续拷贝三遍的复制品。经测试，所得到的复制品用COPYWRIT EXE等拷贝工具均不能复制，这说明将加密软盘的软记号也拷贝过来了。分不出那是源盘和复制品。

如用以上方法不行就换其它型号的计算机以及其它的拷贝工具反复试，就可能有一种组合生产“错位”，从而将加密盘的标志拷贝过来。

文献出处：《软件报》1988年2月20日



# 一种加密dBASE—Ⅱ软件的解密过程与方法

李世光

加密过的软件一般是不可读的，但变码一般要遵循某种规律，如能对这类软件进行认真的剖析，有一些是能够解密的。本人在使用经加密处理的某dBASE—Ⅱ软件时，由于要进行一些改动，对它进行了分析，找到了它的加密原理，并编制了一个BASIC程序将其解出来，下面就将该软件的加密原理和解密方法介绍给大家，供使用参考。

该软件的加密措施有两个：

a、将dBASE—Ⅱ命令文件的每条语句的主保留字（在句首）换成系统能够识别的双字节控制代码。如用“B9H”代替SET，用“8AH”代替STORE等。

b、将语句的其余字符（回车符ODH、换行符OAH除外）的ASCII代码（十六进制的）均加上80H，这样使原本是高位为“0”的ASCII码字符全变成了高位为“1”，如在CC-DOS下按常规方法打印出来就成了没有词意的汉字。而高位“1”的汉字代码，其高位全变成了“0”，打出来就成了西文字符。一个加密前后的dBASE—Ⅱ命令文件如图-1、图-2所示。

```
type exdbpl0.grt
stor "Y" TO FLAG1
do while FLAG1=Y
stor " " TO FLAG1
stor " " TO DD
@ 9, 5 SAY "请给出日期要求 "GET DD PICTURE "99.99.99./99.99.99"
read
do ca SE
case ¥ ( DD, 1, 2 ) = " " .AND. Y ( DD, 10, 2 ) # " "
stor " < = ' " + STR ( VAL ( ¥ ( DD, 10, 2 ) ) , 2 ) + " , " + STR ( VAL ( ¥ ( DD, 13,
2 ) ) , 2 ) + " , " + STR ( VAL ( ¥ ( DD, 16, 2 ) ) , 2 ) + " , " TO D
stor " " TO D1
case ¥ ( DD, 1, 2 ) # " " .AND. ¥ ( DD, 10, 2 ) = " "
stor " > = ' " + STR ( VAL ( ¥ ( DD, 1, 2 ) ) , 2 ) + " . ' + STR ( VAL ( ¥ ( DD, 4, 2 ) ) ,
2 ) + " , " + STR ( VAL ( ¥ ( DD, 7, 2 ) ) , 2 ) + " . " TO D
stor " " TO D1
case ¥ ( DD, 1, 2 ) * " " .AND. ¥ ( DD, 10, 2 ) * " "
stor " < - ' " + STR ( VAL ( ¥ ( DD, 10, 2 ) ) , 2 ) + " , " + STR ( VAL ( ¥ ( DD, 13, 2 ) ) , 2 )
+ " , " + STR ( VAL ( ¥ ( DD, 16, 2 ) ) , 2 ) + " , " TO D
stor " .AND. DATE > = ' " + STR ( VAL ( ¥ ( DD, 1, 2 ) ) , 2 ) + " , " + STR ( VAL ( ¥ ( D
D, 4, 2 ) ) , 2 ) + " , " + STR ( VAL ( ¥ ( DD, 7, 2 ) ) , 2 ) + " ' " TO S
otherwise
stor "Y" TO FLAG1
stor " " TO D
```

```
stor ' ' TO D1
eadcase
enddo
```

图1 解密后的dBASE-Ⅰ命令文件

```
. type a: exddp10.prg
の①d韵铺燎恹
の铺燎恹句①Q
の d d韵铺燎恹
の d                b韵哪
α 宫朵惠肝(3)k 3x 3v HU FZ R火Gs    d桥珍哪 猩迷找其 9 巩构 汞构 巩构Q
把
の优
とま哪 』 d d 文 ま哪 艾博" Q
の12润 釉白至台ま哪 艾博…博 碰日摺a 默背 … : 釉白至台ま哪 冬博一
博 B韵叛
の d d韵谋
とす哪 d d 文 す哪 艾博觀 Q
の14润釉旨至台す哪 … : 釉旨至台す哪 … : 釉旨至台す哪… : d韵叛
のdd韵谋
とま哪 d d文 ま哪 艾博" Q
と 12润 釉旨至台大哪 艾博一博 碰日摺a 默背 … : 釉旨至ま哪 冬博…
博 B韵叛
の廖溪 牧耘窘B 碰日摺a 默爆博…博 碰日摺a 默船博博… 碰日摺a 默番
博…博 B韵忧瓮
吉
の ①d韵铺燎恹
の d d韵叛
の d d韵谋
と
と
```

图2 解密前的dBASE-Ⅰ命令文件

该软件在运行前并没经过复原处理，并且在其它版本的dBASE-Ⅰ系统支持下也可正常执行，至于dBASE-Ⅰ系统为什么能识别并执行这样的软件，由于本人未对dBASE-Ⅰ系统进行过全面剖析，也未得知其道理。

对该软件的解密过程如下：

首先用DEBUG程序对其进行分析，可发现其程序为半可读状态，如图—3所示。

由于DEBUG程序字符识别只取其ASCII码的后7位。所以程序的大多数字符（保留字和汉字除外）均呈其原形，据此即可找出其字符的变码规律。然后再分析每条语句的句首代码与保留字的对应关系——可根据程序可见部分所体现的逻辑意义反复推测出来。如此处理

d1000

```

6036: 0100 8A A0 A2 D9 A2 A0 D4 CF-A0 C6 CC C1 C7 B1 0D 0A "Y" TO FLAG1. .
6036: 0110 88 A0 C6 CC C1 C7 B1 A0-BD A0 A2 D9 A2 0D 0A 8A FLAG1= "Y" . . .
6036: 0120 A0 A2 A0 A2 A0 D4 CF A0-C6 CC C1 C7 B1 0D 0A 8A " " TO FLAG1. . .
6036: 0130 A0 A2 A0 A0 A0 A0 A0 A0-A0 A0 A0 A0 A0 A0 A0
6036: 0140 A0 A0 A0 A2 A0 D4 CF A0-C4 C4 0D 0A 8F A0 B9 AC " TO DD. . . 9.
6036: 0150 B5 A0 D3 C1 D9 A0 A2 47-6B A0 38 78 A0 33 76 A0 5 SAY" GK 8x 3v
6036: 0160 48 55 A0 46 5A A0 52 2A-A0 47 73 A0 A0 A2 A0 C7 HU FZ R• GS "G
6036: 0170 C5 D4 A0 C4 C4 A0 D0 C9-C3 D4 D5 D2 C5 A0 A2 B9 ET DD PICTURE" 9
-d
6036: 0180 B9 AE B9 B9 AE B9 B9 AF-B9 B9 AE B9 B9 AE B9 B9 9. 99. 99/99. 99. 99
6036: 0190 A2 0D 0A AF 0D 0A 89 D3-C5 0D 0A 85 A0 A4 A8 C4 " . . . / . . . SE. . . S(D
6036: 01A0 C4 AC B1 AC B2 A9 BDA2-A0 A0 A2 A0 AE C1 CE C4 D. 1. ) = " " .AND
6036: 01B0 AE A0 A4 A8 C4 C4 AC B1-B0 AC B2 A9 A3 R2 A0 A0 . S(DD, 10, 2) # "
6036: 01C0 A2 0D 0A 8A A0 A2 BC BD-A7 A2 AB D3 D4 D2 A8 D6 ". . . " <= "+STR(V
6036: 01D0 C1 CC A8 A4 A8 C4 C4 AC-B1 B0 AC B2 A9 A9 AC B2 AL(S(DD, 10, 2) ) , 2
6036: 01E0 A9 AB A2 AE A2 ABD3 D4-D2 A8 C6 C1 CC AB A4 A8 ) + ". " +STR(VAL(S(
6036: 01F0 C4 C4 AC B1 B3 AC B2 A9-A9 AC B2 A9 AB A2 AE A2 DD, 13, 2))+ ". "
-d
6036: 0200 AB D3 D4 D2 A8 D6 C1 CC-A8 A4 A8 C4 C4 AC B1 B6 +STR(VAL(S(DD, 16
6036: 0210 AC B2 A9 A9 AC B2 A9 AB-A2 A7 A2 A0 D4 CF A0 C4 , 2)), 2)+ ", " TO D
6036: 0220 0D 0A 8A A0 A2 A0 A2 A0-D4 CF A0 C4 B1 0D 0A 85 . . . " " TO D1. . .
6036: 0230 A0 A4 A8 C4 C4 AC B1 AC-B2 A9 A3 A2 A0 A0 A2 A0 S(DD, 1, 2) # "
6036: 0240 AE C1 CE C4 AE A0 A4 A8-C4 C4 AC B1 B0 AC B2 A9 . AND. S(DD, 10, 2)
6036: 0250 BD A2 A0 A0 A2 0D 0A 8A-A0 A2 BE BD A7 A2 AB D3 = " " . . . ">=" +S
6036: 0260 D4 D2 A8 D6 C1 CC A8 A4-A8 C4 C4 AC B1 AC B2 A9 TR VAL(S(DD, 1, 2)

```

图 3

就可将所有程序分析出来。

下面是根据加密原理编制的BASIC解密程序，程序中将被解密的软件作为BASIC顺序文件进行处理，“program 1”是被解密命令文件名，“program 2”是生成命令文件的文件名，程序中的所有代码均是用十进制表示的。该程序只要稍做改动即可变成一个dBASE-Ⅱ命令文件的加密程序。解密程序如图-4所示。

```

5 INPUT "program1: ", P1S
6 INPUT "program2: ", P2S
10 OPEN P1S FOR INPUT AS#1
20 OPEN P2S FOR OUTPUT AS#2
35 LINE INPUT #1, CS
40 L=LEN(CS)
41 DS=LEFTS(CS, 1)
42 E=ASC(DS)
43 DS=STRS(E)
45 FOR J=1 TO 34
46 READ EE, EES

```

```

47 IF E=EE THEN DS=EES:GOTO 50
48 NEXT J
49 PRINT E, DS
50 FOR I=2 TO L
60 BS=MIDS ( CS, I, 1 )
70 B=ASC ( BS )
80 IF B=10 OR B=13 THEN 100
90 IF B=>128 THEN B=B-128 ELSE B=B+128
100 BS=CHR(B) : DS=DS+BS
110 NEXT I
175 PRINT #2, DS
176 RESTORE
180 GOTO 35
190 CLOSE
195 DATA 185, "set", 160, "go", 138, "stor.", 136, "do while",
143, "e" 161, "find"
196 DATA 137, "do ca", 133, "case", 132, "enddo", 134, "otherwise",
135, "endcase"
197 DATA 193, "use", 129, "else", 159, "erase", 128, "if", 130, "endif",
180, "replace"
198 DATA 169, "loca", 156, "continue", 142, "sele", 145, "append",
151, "copy", 139, "?"
199 DATA 170, "loop", 194, "wait", 155, "display", 172, "pack",
154, "delete"
200 DATA 174, "quit", 186, "skip", 163, "index", 175, "read",
131, "do", 140, "release"

```

图4 解密BASIC程序

文献出处:《计算机世界》月刊1987年10期 45—46页

## 对IBM PC/XT 上一些游戏程序的解密方法

南京 王佩峰

在IBM—PC/XT上运行的游戏程序大多是经过加密处理的,用COPY或DIKSCOPY命令不能进行复制。(用COPYWRIT程序虽可复制,但复制后的程序仍然是被加密的)。因为这些游戏程序在DOS状态下寻找不到目录,所以无法用DEBUG进行跟踪分析。对这些游戏可用下述方法进行分析解密。

方法一：在游戏程序调入内存后，通过键盘中断将控制转给预先调入内存的DEBUG程序，然后用DEBUG进行分析，步骤如下：

1. 用DEBUG将DEBUG程序读入，在DEBUG状态下通过M(MOVE)命令，将DEBUG移动到比较安全的内存段。

2. 通过E命令修改INT5(屏幕打印中性的中断向量，使之指向移动后的DEBUG程序的入口。

3. 通过A命令写一条INT19(调A盘引导程序并将控制转给它)语句，并执行以调入A驱动器的游戏程序并运行。

4. 按下“↑prtsc”(屏幕打印中断键，使控制转给DEBUG，这时就可以分析解密后调入内存的游戏程序。

说明：如果不通过INT19调入游戏程序，采用其它方法(例如总清调入)，将会重新装入INT5的中断向量，用控制无法移动。

许多游戏程序采用了防止DEBUG跟踪的措施，以“LODE RUNNER”(警察抓小偷)为例，当控转给DEBUG后，屏幕出现“You are not alloneil to read or modify”提示，DEBUG无法运行。这时可采用第二种方法。在游戏读入运行后，通过键盘中断将控制转给ROM BASIC，然后在ROM BASIC状态下调用机器语言子程序进行读写操作或分析，步骤如下：

1. 启动DEBUG在DEBUG状态下通过E命令修改INT5的中断向量，使之指向ROM BASIC入口。

2. 执行INT19软中断同上。

3. 按下(↑Prt—Sc)使控制转给ROMBASIC。

说明：ROM BASIC是磁带BASIC，它不支持磁盘操作，我们可以通过机器语言调用ROM BIOS实现对磁盘的操作。

这两种方法虽可实现控制的转移，但破坏了原程序的断点。这样虽然知道游戏程序存在于内存，但无法知道入口地址，也很难知道游戏存在于内存的范围。而对于不算短的机器语言程序，读懂并非易事，此外，DEBUG或ROM BASIC的RAM通讯区也可能会覆盖原程序。

我们注意到，许多游戏程序，虽可不需DOS的支持，却仍要调用许多ROM BIOS中的子程序如：INT10(显示器驱动程序)INT16(键盘驱动程序)INT13(磁盘驱动程序)。如果我们恰当地修改这些软中断的入口或出口参数，则可不必要分析原程序，巧妙地实现对游戏的解密，达到事半功倍的效果。

仍以“LODE RUNNER”为例，引导程序将主程序调入后，它并不立即运行，而是对加密的磁道进行读操作，以进行版权检查，如果出口参数表明操作成功，则认为是原版，转入主程序。如失败，则认为是复制品，程序不再继续。对此，如果我们修改一下INT13的出口参数，使出错标志强制启成成功标志，则可很容易地实现对原程序的解密，使DISKCOPY复制的程序能正常运行，步骤如下：

1. 调入DEBUG运行。

2. 将INT13的中断向量移动到INT50的向量区。

3. 将INT13的中断向量地址指向RAM中某区(如4000:0)

4. 在INT13的入口处编写如下一段程序:

```
4000:0000 INT 50
4000:0002 MOV AX, 8
4000:0005 RETF 2
```

5. 用INT19调游戏程序(用DISKCOPY得到的盘)

文献出处:《软件报》1988年5月14日

## 用2字节的短程序解密

上海 史友蓼

关于在IBM—PC机中用BASIC语言编制的程序用“P”命令加密后如何解密过去都谈了很多。这里我想介绍一个只需2个字节的短程序来解除“P”命令的密。这2个字节程序编制方便,使用更方便。

由于判断加“P”关键是在于首字节是否为“FE”还是“FF”以及不同版本的BASIC在内存缓冲区的位置。所以要解密,必须要找到缓冲区的首地址及修改首地址的内容。现编制一个2字节的短程序来解决以上的问题。

编制方法如下:在DOS的环境下

```
A>debug↵
e- DS:0100 ff 01↵
-D PJM:bas ↵
-r DX ↵
CX 0000
:0002 ↵
-W ↵
H iting 0002 bytes
-q ↵
```

这样在A盘中留下了一个2字节的PJM.BAS程序。

使用时,只需在BASIC状态下,装入加过“P”命令的程序,再装入PJM.BAS,再读加过P命令的程序即可。步骤如下:

在BASIC状态下:

Load“文件名”↵(该文件即加过P的BASIC的程序)

Load“PJM.BAS”↵

List↵即可,若要保存则

SAVE“文件名”不加P

文献出处:《软件报》1988年5月14日

## 过程对称的制、解密程序

银川 彭庆杰

本程序采用信息替换方法进行制密和解密，其算法如下：

加密： $Y_i = X_i \oplus K$ .....①

解密： $Z_i = Y_i \oplus K$ .....②

证明：将①式代入②式，有：

$Z_i = (X_i \oplus K) \oplus K$

由结合律：

$Z_i = X_i \oplus (K \oplus K) = X_i \oplus 0 = X_i$

证毕。

用计算机实现以上过程时，对每一字符均采用同一加密钥进行处理。这样，该程序就象一个开关一样，执行次数为奇数时，则获得密文，执行次数为偶数时，则获得明文，值得注意的是，密钥最好选取为文件中不曾出现过的字符，例如：其ASCII码为91, 93, 96, 123, 124, 125, 126的那些字符。

本程序适于IBM PC/XT机及兼容机，如在CCDOS下使用，应安排如下批处理程序：

```
REM JIAMI.BAT
```

```
ECHO OFF
```

```
JM
```

```
MO
```

```
ECHO ON
```

(JM MO均为执行程序)

其中，JM.EXE是JM.BAS经过编译后产生的执行程序，MO.EXE可利用小汇编来编写：

```
MOV AH, 00
```

```
MOV AL, 06
```

```
INT, 10
```

```
MOV AH, 4C
```

```
INT 21
```

以下是过程对称的制、解密程序的清单：

```
2'----- JM.BAS/NINGXIA COMPUTER CENTER -----
```

```
3'----- BY PengQingjie ON 870819 -----
```

```
40 SCREEN 1
```

```
50 CLS : COLOR 9 : LS=CHRS(7)
```

```
51 SOUND 2800, 7 : LOCATE 4, 15 : PRINT "1, JIA/QU MI" : LOCATE
```

```
5, 15 : PRINT "2, BREAK" : LOCATE 6, 15 : PRINT "3, QUITE"
```

```

LOCATE 8, 15: PRINT " Please select": ZS=INPUTS( 1 ): IF ZS
  =' 3 ' THEN SYSTEM ELSE IF(ZS< )" 1 ")THEN 52ELSE COTO 54
52 IF ZS=" 2 " THEN END ELSE GOTO 54
54 IF ZS< )" 1 " THEN 50 ELSE 55
55 COLOR 20: CLS: LOCATE 4, 15: INPUT "Filename:": WMS: LOCATE
  5, 15: PRINT "K E Y: "; KS=INPUTS( 1 ); K=ASC( KS )
56 CLS: PRINT LS; LS; SOUND 1600, 7: SOUND 500, 10: LOCATE 4, 15:
  PRINT "Please Wait"

68 :
130 OPEN WMS FOR INPUT AS # 1
140 OPEN "SSS" FOR OUTPUT AS # 2
150   WHILE NOT EOF( 1 )
160   LINE INPUT #1, AS: N=LEN( AS )
170   DS=""
180     FOR I= 1 TO N
190     E=ASC(MIDS( AS, I, 1 ) )
200     D=E XOR K: DS=DS+CHRS( D )
210     NEXT I
220   PRINT # 2, DS
230   WEND
240   CLOSE
250   KILL WMS
260   NAME "SSS" AS WMS
170 GOTO 50

```

文献出处: 《软件报》1988年5月14日

## APPLE—II BASIC程序加密后的一种解密方法

武汉钢铁学院 史华忠

有加密就有解密, 它就象矛盾双方一样, 既对立又统一。本文针对本刊1986年第三期武进军、李应刚文章(下面简称“武文”)的加密方法, 提出个程序加密后的解密方法。该程序短小精悍, 通俗易懂, 从中还可体会到字符串处理函数的功用。

### 一、对程序加密的分析

苹果机上有两种方式可以对程序列清单, 一种在主机运行时, 可以按CTRL—C或按



CTRL—RESET键来停止程序运行且不破坏程序，然后键入LIST命令列出清单；另一种是对已存盘的文件使用DOS的内务命令LOAD将磁盘上的文件调入内存，然后列清单。因此要使用户看不见程序内容，就必须分别对控制键和LOAD命令进行屏蔽。

### 1. 对控制键的屏蔽

“武文”用POKE1012, 0对CTRL—RESET键屏蔽，用ONERR GOTO NEW对CTRL—C键屏蔽。

### 2. 对LOAD命令的屏蔽

“武文”提出在程序名中加入不可显示字符（如CTRL键+字母键），用SAVE命令将源程序存盘，如键入SAVE SHZ CTRL - W -1，其中加框者不可显示，即文件名为SHZ-1，然后编写一个辅助程序：

```
10 PRINT CHR$(4); "RUN SHZ CTRL - W -1"
```

以文件名SHZ-0存入磁盘，用户键入RUN SHZ-0即可运行源程序。

## 二、解密程序分析及用法

上面两种加密方法中，不管对控制键如何屏蔽，只要把文件名“SHZ CTRL - W -1”中的不可显字符CTRL-W检查出来，即可用LOAD命令将源程序装入内存，用LIST命令列出清单。故该问题归结为如何查出不可显示字符。为了方便，将解密程序用法及功能一并叙述如下。

首先，LOAD SHZ-0，键入LIST命令，显示：

```
10 PRINT CHR$(4); "RUN SHZ CTRL - W -1"
```

用苹果机的编辑功能，按ESC键，再按I键，光标上移到O，按J键，光标移至1字上，再用→键移光标到P上，键入A\$=字符串，用空格键删去“；”及其前的字符，用→键移光标至末尾，回车，LIST，显示：

```
10 A$="RUN SHZ CTRL - W -1" 然后再键入下列解密程序段，
```

```
20 A=LEN(A$)
```

```
30 FOR I=1 TO A
```

```
40 A$=RIGHT$(A$, A-I+1)
```

```
50 A(I)=ASC(A$)
```

```
60 PRINT "A(" ; I; ")=" ; A(I)
```

```
70 NEXT
```

```
80 END
```

运行该程序，即可打印出原字符串中所有字符（包括不可显示字符）的ASCII码，对照ASCII码表，即可查出原字符串中（即程序名中）的不可显示字符。

程序20句计算字符串长度，40句从左到右依次剥去第一个字符，50句将字符串中的一个字符的十进制ASCII码放入A(I)中，如此循环往复，直至字符串“RUN SHZ CTRL - W

1”中全部字符的ASCII码都查出为止。本例中，查出不可显示字符的ASCII码为23，即CTRL-W。此时，再键入LOAD SHZ CTRL-W -1即可将源程序调入内存，用LIST命令列出全部清单。

文献出处：《电脑与微电子技术》1987年2期36页

## 一种解密加“P” BASIC程序的方法

广州 萧矩明

IBM-PC/XT的BASIC解释语言提供了一种对用户程序加密的方法，只要在SAVE命令中加上“P”参数，以后调用该程序就只能运行而不能列表（LIST）。在使用中，笔者找到一种简单的解密方法。设现有一个名为MUSIC.BAS的待解密程序，解密过程如下：

一、读出加密程序的长度。

```
DIR MUSIC.BAS
```

```
MUSIC BAS 8591
```

二、运行图一中的程序，求出装载程序的内存地址。在这里所使用的操作系统是MS DOS 2.1，BASIC是A2.1版本，求得段地址SEG：3200，偏移地址OFFSET：272。对于其它版本的操作系统和BASIC语言，所求出的内存地址可能是别的值。

三、在BASIC状态下顺序执行图二中的各命令进行解密。第一条命令是装入加了密的源程序。第二条命令是根据第二步的结果定义段地址。第三条命令是把内存中的程序以内码形式存入磁盘，其中第一个参数是偏移地址（OFFSET=272），第二个参数是程序的长度。第四条命令是释放内存和放弃其执行模式（如不接受SAVE命令等）。第五条命令是把程序装回内存的原位置。第六条命令是把程序以ASCII码形式存到磁盘上。至此解密工作完成，只要执行第七条命令后就可使程序列表和运行。

图一

```
10 FIND ADDRESS'  
20 K=0  
30 J=64*K  
40 DEF SEG=J  
50 I=0  
60 I=I+1  
70 IF I=1025 THEN K=K+1 : GOTO 30  
80 IF PEEK ( I ) < > 70 THEN 60  
90 IF PEEK ( I+1 ) < > 73 THEN 60  
100 IF PEEK ( I+2 ) < > 78 THEN 60  
110 IF PEEK ( I+3 ) < > 68 THEN 60
```

```

120 IF PEEK (I+5) < 5 THEN 60
130 IF PEEK (I+6) < 58 THEN 60
140 IF PEEK (I+7) < 58 THEN 60
150 IF PEEK (I+8) < 82 THEN 60
160 IF PEEK (I-8) < 0 THEN 60
170 PRINT "SEG=", J "OFFSET="; I-8
180 END

```

OK

RUN

SEG=3200            OFFSET=272

图二

LOAD "MUSIC"

DEF SEG=3200

BSAVE "MUSIC", 272, 3591

NEW

LLOAD "MUSIC", 272

SAVE "MUSIC", A

LOAD "MUSIC"

文献出处：《软件报》1988年5月14日

## “P”BASIC源程序的解密方法

刘冬喜

BASIC语言是人们最熟悉，使用最为广泛的一种语言。在IBM PC上使用BASIC语言程序设计开发的系统很多。但由于BASIC解密程序版本资料一直没能和用户见面，所以BASIC的加“P”参数的源程序，用户无法列表阅读。

目前，对“P”程序解密的办法很多。在实际工作中，我们发现用debug对“P”参数程序进行解密，操作简单，易掌握。这里介绍给大家。

### 1. BASIC存储文件的格式。

在使用SAVE命令存一个内存文件时，可以用不同的参数产生三种不同格式的磁盘文件。

### 1. ASCII代码格式:

SAVE 文件名, A

这种格式直观,它的内容和列表时一样。

### ii. 二进制压缩代码格式:

SAVE 文件名

这种存储格式的代码不能阅读,它经压缩以后变换了,只有某些字符串保留原样。但它是唯一可以以执行代码,也可以列表。

### iii. P代码格式:

SAVE文件名, P

用这种代码存储文件,存储的内容不能阅读,也不能列表。

这里用一简单的语句:

10 Print "1 2 3 4 5 ABCDE" 为例,看看三种格式的存储代码情况(见图一)。

图一 BASIC源程序三种存储代码举例

7 B29 : 0100 31 30 20 50 52 49 4E 54—20 22 31 32 33 34 35 41

7 B29 : 0110 42 43 44 45 22 0D 0A 1A—BA 39 09 E8 5D 00 E8 5E

7 B29 : 0100 FF 34 10 0A 00 91 20 22—31 32 33 34 35 41 42 43

7 B29 : 0110 44 45 22 00 00 00 1A 1A—BA 69 09 E8 5D 00 E8 5E

7 B29 : 0100 FE 9B A7 BF 54 E2 12 BD—40 53 26 55 E5 D1 61

7 B29 : 0110 99 67 91 13 E6 13 1A 1A—BA 39 09 E8 5D 00 E8 5E

从图一可以看出,它们的程序代码是不同的。特别是第一字节标志位不同。P代码文件的标志位是FE,压缩二进制文件的标志位是FF,而BASIC对这些文件的操作主要取决于标志位。

### 2. BASIC中LOAD, LIST命令对“F”文件的处理。

LOAD命令装入文件时,首先把文件代码读入一缓冲区BF<sub>1</sub>中,然后判断标志位。若标志位为“FE”,则执行一转程序将BF<sub>1</sub>中的P代码转换成压缩二进制代码存入另一缓冲区BF<sub>2</sub>中;若标志位为“FF”,则直接将BF<sub>1</sub>的程序内容移入BF<sub>2</sub>;若二者都不是,则执行一段A代码转换程序;将BF<sub>1</sub>中A代码转换成压缩二进制代码存入BF<sub>2</sub>中。既然BF<sub>2</sub>中都是压缩二进制形式,为何List命令不能对“P”文件列表呢?这是因为在LOAD命令装入标志位为“FE”的文件的同时,将某些开关单元进行了一定的设置,封锁了List命令的执行。

### 3. BASIC中SYSTEM命令情况

在LOAD命令时,将P文件转换成了二进制压缩形式存入BF<sub>2</sub>中。那么是否有办法将BF<sub>2</sub>中的内容复制出来呢?回答是肯定的。

在执行BASIC的System命令后,用调试程序debug的命令进行查找,发现BF<sub>2</sub>依然存在。这样用修改标志位字节和写文件的方法,把BF<sub>2</sub>的内容保存下来。这份保存下来的程序

就是我们所指的“P”方式解密程序。在debug中查找到文件长度为26112字节，进入余空间为60891字节，版号为A2.10的BASIC的BF<sub>2</sub>的首地址为3921。这样如果在该版本BASIC中要对某一个P文件进行解密，只需知道P文件的长度（16进制表示）就可以了。

#### 4. 解密具体方法

这里用一具体的实例，来说明上述的解密方法。C盘上有一个“P”文件，文件名为：YCX.BAS，长度为十六进制的3AE字节。其具体操作方法是：

```
C>B A S I C           ; 进入BASIC
  LOAD YCX           ; 装入“P”文件
  SYSTEM            ; 退出BASIC
C>d e b u g         ; 进入debug
  —N Y C X 1 . B A S ; 为解密程序取名
  —R C X
    C : 0000         ; 定义解密程序文件长度
    : 3 A E
  —E100, FF         ; 定义新标志
  —M 3921, BCCF, 101 ; 将BF2的内容移到101开始的缓冲区
  —W
    writing 3 A E bytes ; 写文件
  —Q               ; 退出debug
```

这样，YCX1.BAS文件就是“P”文件YCX.BAS的解密程序。可以用List命令列表。

对于其它不同版本与的BASIC，用同样的方法找到BF<sub>2</sub>的首地址，同样可以对“P”文件进行解密。

文献出处：《电脑与微电子技术》1987年2期31—32页

# 加密BASIC程序的一种简单破译方法

唐志翔

## 一、引言

笔者曾应一些用户之邀，破译了一些加密的BASIC程序。在这个过程中，对IBM-PC的BASIC解释程序作了一点研究，获得了一种简捷方便的破译方法。这种方法比今年3月8日在“计算机世界”报上发表的方法要简单得多。它不需要借用其它任何工具程序，对存贮介质也无特殊要求。下面就把这个方法介绍如下。

## 二、原理

BASIC源程序在存贮时若加了P参数，则以密码形式输出。如果设法寻找密码表，再通过解密算法把源程序还原，则需要编制专用的汇编程序，这样虽然可行，但工作量相当大，并非好办法。直捷的途径应从BASIC本身去寻找答案。

IBM-PC的BASIC占用最大可达64KB空间的内存区作为数据段。这个空间依次用作为解释程序工作区、用户的源程序存放区、单变量区、数组区、字符串变量区、及堆栈区。在解释程序工作区中，存放大量的标志和指针，指示着程序运行的状态，内存分配情况等等。IBM-PC的BASIC源程序除了有以密码形式存贮的以外，另外尚有以机内码及ASCII码格式存贮两种方式。用ASCII码格式存贮的源程序除了可在BASIC解释程序下显示外，还可以在DOS下用TYPE命令显示。而用机内码格式存贮的源程序则要在BASIC解释程序下才能显示，用密码格式存贮的源程序则无法用常规方法显示，即使用DEBUG来观察它的内容，也很难找出什么端倪。然而，这三种不同格式的源程序，一旦调入内存，就以统一的机内码形式存放在相同区域。（见BASIC使用手册）。换言之，对加密的BASIC源程序，解释程序自己先对其作了解密工作。只不过当用户企图用“LIST”命令来显示程序时，它用“ILLEGAL FUNCTION CALL”（非法功能调用）来回答罢了。BASIC解释程序之所以能够作出这样的反应，是因为它对源程序译码解密之时，在工作区内设置了一个“软开关”，在用户用“LIST”请求显示源程序时，若这个“开关”是置位状态，解释程序就拒绝这个要求，否则就执行命令。

请看下面这段汇编程序：

```
F600:7E92 FA          CLI
F600:7E93 BA6000     MOV    DX, 0060
F600:7E96 8EDA      MOV    DS, DX
F600:7E98 8EC2      MOV    ES, DX
F600:7E9A 8ED2      MOV    SS, DX
F600:7E9C 32C0      XOR    AL, AL
F600:7E9E A26404     MOV    [0464], AL
```

这是IBM-PC的BASIC解释程序在开始时的几条指令。在最后两行，程序将464H单元的值复位。请注意这个单元，它正是我们要寻找的“软开关”。只要这个单元处于复位状态，BASIC解释程序就不拒绝“LIST”命令。因此，破译方法的关键就是设法在BASIC解释程序解码完毕之后，把“开关”复位。具体做法如下。

### 三、步 骤

全过程可分为6步。

1. 在DOS控制下，进入BASIC解释程序。

这里，对DOS无特殊要求。源程序内带汉字的也可以在西文DOS下实现解密。但为了便于破译后立即阅读源程序，建议此时启动相应的中文DOS。

2. 生成一个解码工作文件“DECODE.BAS”。

在屏幕上打入命令：BSAVE “DECODE” . , 1124, 1 (CR)。

生成文件时，可以用A, B, C中任意一个驱动器。生成的文件本身也很短，只要盘体上有一个CLUSTER的空余位置就行了。（事实上，它只有9个字节长。）

3. 调入欲破译的BASIC源程序。

这一步很简单，只需用命令：LOAD “文件名”即可。此时，若用“LIST”命令，仍可看到屏幕出现“ILLEGAL FUNCTION CALL”字样。

4. 调入解码工作文件“DECODE.BAS”在屏幕上打入命令：BLOAD: “DECODE” (CR)。

把刚才生成的解码文件调入内存。这是我们方法中关键的一步。在这一步中，我们把BASIC解释程序设置的“开关”给复位了。至此，对源程序保密的禁令已经完全解除。此时，若用“LIST”命令，可以看到屏幕上显示出源程序。

5. 保存解密后的源程序

获得解密后的源程序，除了用“LIST”显示以外，还可用“LLIST”命令来获得硬拷贝，或者用“SAVE”存储起来。这时，对使用的盘片和文件名都没有限制。你可以用原来的盘片，甚至还可以用原来的文件名，把原有程序复盖掉。当然，为了谨慎起见，建议你另用一张盘片保存解密后的程序，或者，在同一盘片上，但冠以不同的名称。

6. 继续破译其它加密程序或结束

这一步首先判断需要破译的程序是否全部处理完毕，若还有，则转到第3步，重复步骤3~5，若已全部完成，则可退出BASIC解释程序，结束本次破译工作。

### 四、推 广

以上介绍的方法，虽是针对IBM-PC型微机而言。但事实上，它完全适用于IBM-PC/XT型及IBM-PC/AT型微机。更推而广之，在与IBM-PC相兼容的许多其它型号的微机上也适用。当然，由于兼容程度不同，有些机型如GW0520A型，就不能用此方法（但GW0520C-H型却又可以）。这里提供一条简单的判定法则：若你的微型机上不能运行IBM-PC机的BASICA.COM解释程序，则以上方法对你的微型机是不适用的。否则，不妨一试

文献出处：《计算机世界》月刊1987年10期43—44页

# 用PASCAL编写加“P”保护的BASIC程序的解密程序

浙江省计经委计算中心 俞杰

在IBM PC/XT及其兼容机上编制的BASIC程序，在存盘时可以用加P参数的方法，使其被存在磁盘上的程序受到保护，程序只能使用，不能列出清单或将程序修改。根据其加密的原理，用汇编语言可以很方便地设计解密程序。但是用汇编语言编制程序难度较大，可读性较差，对不怎么熟悉汇编语言的人，是有一定困难的。为此，笔者对怎样用高级语言PASCAL编制解密程序，进行了一些尝试。

## 一、二进制代码转换成密码

在此，先简述一下二进制代码是如何转换成密码的。变换使用了二个数据表和二个表指针。表一中有十三个数值，表二中有十一个数值，指针一的初值为十三，指针二的初值为十一。每转换完一个代码，两个指针分别减一，当减到零时，又恢复初值继续使用。转换方法为 $Y_i = ((X_i - a) \oplus k2b \oplus k1^a) + b$ ，其中 $Y_i$ 为加密后的密文字节， $X_i$ 为未加密数据的明文字节， $a$ 为指针一， $K1^a$ 为表一中的数值， $K2$ 为表二中的数值， $\oplus$ 为异或运算。有了加密过程算法，我们便可求出解密算法。

$$x_i = ((Y_i - b) \oplus K1^a \oplus K2b) + a$$

表一：9 A, F 7, 19, 83, 24, 63, 43, 83, 75, CD, 8 D, 84, A 9.

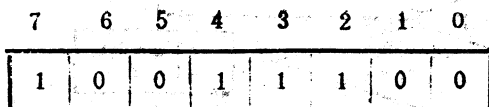
表二：7 C, 88, 59, 74, E 0, 97, 26, 77, C 4, 1 D, 1 E, .

## 二、程序的数据结构

在上面所述的解密算法中，涉及到按位进行的异或操作，而在高级语言PASCAL中无直接进行按位操作的语句。因此就必须进行一些变换，使得能用PASCAL可以实现按位进行的异或操作。

在PASCAL中有一结构类型集合，可以把集合理解成内存中的二进制字节，令8位二进制字节的每位表示集合中基元素的出现或不出现。0表示基元素不出现，1表示基元素出现。例如表示一个字节1001 1100就可用集合{2, 3, 4, 7}表示，可形象地用图一表示。

图1



{2, 3, 4, 7}

在算法中用到的两张数据表，都直接用于异或运算，我们可以用一集合数组存放它们。



类型可以表示成: Var binary: Set of ..7; con: array (1..24) of binary. 两张表中的数值都直接化成集合常量, 存入此数组。

另外, 把未解密的输入文件及已解密的输出文件都作为PASCAL程序的文本文件(字符文件)进行处理。可表示成 Var outf, nf: fite of char; .

### 三、几个转换过程及集合异或操作

对于从原输入程序读出字符, 我们可以把用ord函数, 使其转换为十进制整数进行处理。但因为对数有按位进行的异或操作。因此, 应该有一个从十进制整数转换成表示它的二进制字节的集合的过程, 此过程如程序 1。

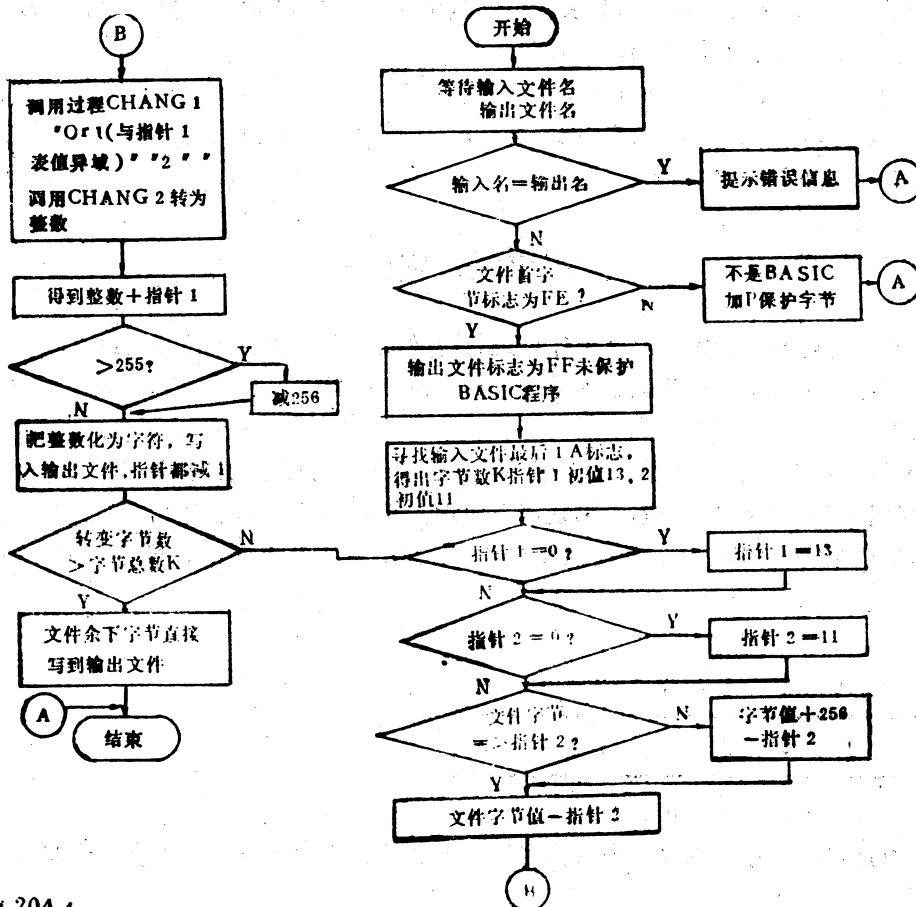
同样, 对于已经进行了异或操作后的集合, 我们必须把其还原成十进制整数, 此过程如程序 2。

有了表示八位二进制字节的集合, 就可以很容易实现按位进行的异或操作, 此过程如程序 3。

### 四、程序的实现

有了上面几个过程, 程序的实现就比较简单了, 下面给出程序框图, 如图 2 所示。

图 2



程序 1

```
7:  PROCEDURE CHANG1 ( AA:INTEGER, VAR BB: BINARY );
8:  VAR I, N: INTEGER;
9:  BEGIN
10: BB:= [ ] ;
11: FOR I:= 0 TO 7 DO
12: BEGIN
13: N:= AA MOD 2 ;
14: AA:= AA DIV 2 ;
15: IF N= 1 THEN
16: BB:= BB+ [ I ] ;
17: END;
18: END;
```

程序 2

```
19: PROCEDURE CHANG 2 ( BB 1 , BINARY; VAR AA1, INTEGER ) ;
20: VAR I, J, MM:INTEGER;
21: BEGIN
22: AA1:= 0 ;
23: FOR I:= 0 TO 7 DO
24: BEGIN
25: IF I IN BB1 THEN
26: BEGIN
27: MM:= 1;
28: IF I > 0 THEN
29: FOR J:= 0 TO I-1 DO
30: MM:= MM * 2;
31: AA:= 1=AA1+MM;
32: END;
33: END;
34: END;
```

程序 3

```
35: PROCEDURE OR1 ( BBB, BBB1, BINARY; VAR BBB2, BINARY ) ;
36: VAR I: INTEGER;
37: BEGIN
38: * BBB2:= [ ] ;
39: FOR I:= 0 TO 7 DO
40: IF NOT ( I IN BBB ) AND ( I IN BBB ) AND ( I IN BBB1 ) OR ( I IN
   BBB ) AND NOT ( I IN BBB1 )
41: THEN BBB2:= BBB2+ [ I ] ;
```

42: END;

## 五、结束语

本文提供的设计方法,虽然具有较好的可读性,相对编制程序较为方便,但需要花费较多时间执行程序。相对汇编程序编制的解密程序,效率有一定降低。而且用PASCAL编写此程序也不止一种方法,可以不采用集合类型,亦同样能达到按位进行的异或运算的目的。但为了加深对PASCAL集合概念的理解,还是采用集合类型,相信它对编写其它程序不无益处。

文献出处:《电脑与微电子技术》1987年2期34—35页

## 一种简便的IBM—PC BASIC程序的解密

陈克明

在IBM-PC机上拥有众多的BASIC编制的实用的程序。但这些程序用P参数存盘以后,用户只能运行,不能列表显示,更不能对它进行修改或功能扩展,给用户带来了困难。这里介绍一种简便的解密方法,它适用于IBM-PC及兼容机BASIC2.0版本和3.0版本。

### 原 理

BASIC源程序在用SAVE命令时加上了P参数,则源程序以密码形式存磁盘。而用Load命令将加密的BASIC源程序调入内存时,BASIC解释程序对源程序进行解密,最后以一般BASIC源程序形式存于内存中,但这时在内存中置了一个软开关,禁止用户用List命令来显示源程序。

若这时退回到系统中去,用debug程序将内存中已经解密的程序写到磁盘中去即可完成解密工作。

### 步 骤

#### 1. 寻找源程序在内存中的位置

首先,先编一程序以加密形式存于磁盘。

例: 10 ABCD

Save "P", P

System

(注:虽然这个程序是不能运行,语法有误,但能说明问题)

```

A>debug
S 0 FFFE 'ABCD'
-xxxx xxxx
(段地址位移量)
-Q

```

进入debug程序，用搜索命令S从位移量0到位移量FFFE中，查找字符串'ABCD'，字符串'ABCD'就是BASICA程序P.BAS第10句中的字符串。计算机显示出'ABCD'在内存中的位置，记住位移量。

## 2. 计算被解密文件的长度

如果要解密的文件为MS.BAS，用DIR命令显示其长度，并将其长度转化为十六进制，如用户转化有困难可用下面一种方法：

```

A>debug MS.BAS
-RCX
CX xxxx (记下其长度)
:
-Q

```

## 3. 将被解密文件调入内存

```

A>BASICA
LOAD "MS"
SYSTEM

```

## 4. 用DEBUG程序进行解密

### (1) 修改文件头

文件头的位置=位移量-5 A>DEBUG

E文件头位置FF

非加密BASICA源程序文件头为FF。

### (2) 修改文件长度寄存器CX将文件长度放到寄存器CX中去

```
-RCX
```

长度

### (3) 将解密文件存盘

先将解密文件告诉计算机，然后将内存中已解密的文件存盘。

```
-NMS.BAS
```

```
-W文件头位置
```

```
-Q
```

到这里已经将加密的程序解密了。

## 例 子

```
C>BASICA
```

```
10 ABCD
```

SAVE "P" , P

SYSTEM

用DEBUG程序找程序在内存中的地址C>DEBUG

-S 0 FFFE '\ABCD'

0BF 3 :382F

这里0BF 3为段地址, 382F为位置量。

-Q 退出DEBUG程序。

如果要解密的程序名为M.BAS, 用DIR查其长度。

C>DIR M.BAS

显示 M BAS 42 1 -01 -80 12:13a

可见M.BAS长度为42其十六进数为2A。或:

C>DEBUG M.BAS

-RCX

CX 002A

-Q

将M.BAS调入内存

C>BASICA

LOAD "M"

SYSTEM

C>DEBUG

-E382AFF 修改表头, 表头位移量=382F-5=332A

-RCX

CX0000

; 2A 修改文件寄存器

-NM.BAS 指出文件名

-WBF 3 :382A 将文件存盘

Writing 002A bytes 计算机显示有2A个字节已写盘。

-Q 退出DEBUG程序

文献出处: 《电脑与微电子技术》1987年2期32—33页

## 用DEBUG解密IBM PC BASIC程序

蒋朝根

BASIC源程序有几种存盘方式, 但在调入内存后其在内存的存贮方式, 除了一个文件属性标志不同外, 都以压缩二进制方式存贮。因此可通过改变文件标志再次存盘便可实现解

密。

虽然本方法操作不如专用解密软件那样方便，但是由于本方法除了使用DEBUG外，不需要任何其它软件，因此，在没有专用解密软件，或专用解密软件不在身边时，该方法就显示出它的优越性了。它随时随地都可方便地使用，对于偶尔想解密的用户来说尤其实用。

某些用户可能对DEBUG不大了解，但只要按步骤进行，很容易实现。

这种方法的解密时间基本上取决于键盘操作时间，当键盘操作完成时解密也就完成。

本方法的操作步骤以实例给出。这里被解密的基本程序文件名为SM.BAS，解密后的文件名为TEMP.BAS。如果解密其它文件，则在相应的地方打入对应的文件名即可。在DEBUG中的文件名包括扩展名。

方法实例如下：

( 1 ) BASICA ( 或 BASIC )

```
10 ABCCCCCCCCCCCCCCCCCCCCCCCCCCCCC  
SYSTEM
```

( 2 ) DEBUG

```
-S 0 FFF0 ' ABCCCCCCCCCCCCCCCCCCCCC'
```

```
4 F08: 10EE
```

```
-H 10 EE 5 该行方框中的值引用上一行方框中的值
```

```
10 F3 10 E9 记下该行方框中的值，设为XX
```

```
-Q
```

( 3 ) DEBUG SM.BAS SM.BAS为要解密的基本文件

```
-R CX
```

```
CX 3 E92 记下该行方框中的值——文件长
```

```
-Q
```

( 4 ) BASICA ( 或BASIC )

```
LOAD" SM
```

```
SYSTEM
```

( 5 ) DEBUG

```
-NTEMP.BAS 定义解密后的文件名为TEMP.BAS
```

```
-RCX
```

```
CX 0000
```

```
3 E92 方框中的值为文件长，由( 3 )得
```

```
-M 10 E9: FFFF 100 方框中的值由( 2 )得到的XX
```

```
-E100 FF
```

```
-H3 E92: FF 方框中的值为文件长
```

```
3 E91: 3D93
```

```
-E 3 F91: 3D93 1A 该行方框中的值引用上一行方框中的值
```

-W 写盘  
-Q 退回DOS

- 注：1. 每次键入一行字符后以回车结束。  
2. 若需解密下一个文件，只需从（3）开始。

文献出处：《计算机世界》月刊1987年10期47页

## 用IBM—PC BASIC对P参数文件解密

任 信 良

众所周知，PC-BASIC文件一旦经P参数存盘，代码即被加密。此后，用户既无法在屏幕或打印机上列表，也不可能再以机内码或ASCII码形式重新存盘。本文将介绍的是用PC-BASIC 解析程序对P参数文件解密方法，这种方法对 IBM-PC / XT及其各种能运行 PC-BASIC兼容机均有效。

### 一、BASIC的解密原理

事实上，BASIC文件在内存中运行前，代码必须被转换成压缩二进制形式的机内码。当用Load命令从盘中读取文件后，加密的代码和ASCII码均将被转换为机内码。图一示出BASIC代码转换流程。

用SAVE命令将BASIC文件存盘时，机内码文件头一个字节前加入FF标记；P参数文件前加入FE标记。因此，在BASIC读文件时，就可根据文件标记来辨认文件代码是否被加

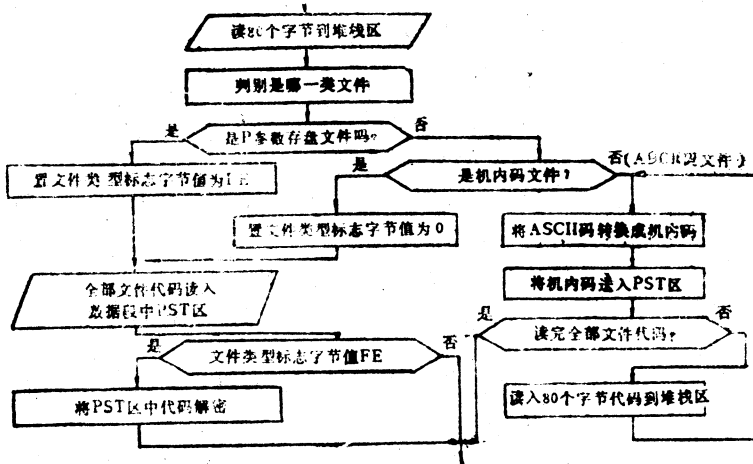


图1

密。同时，根据不同文件在文件类型标志字节内写入一个值，这个值将保存到下一个BASIC文件被读入内存为止。

当文件类型标志字节的值表示文件是以P参数存盘时，解析程序立即将代码还原成二进制机内码，同时不允许用LIST命令对程序文本列表，从而保证受保护的文件既能满足BASIC对运行代码的要求，又不会让用户得到程序文本。表1为文件类型标志字节中值与文件类型的关系。

表 1

文件标志	文件类型标志字节中值	文件类型	备 注
FE	FE	P参数存盘文件	读入内存后由BASIC解密，不能列表
FF	0	机内码文件	可以列表
30~39	0	ASCII码文件	BASIC将ASCII码转换成机内码可以列表

由此可知，若能设法使文件类型标志字节值始终为零，或使该字节内非零数值不影响列表操作，则BASIC将不理睬读入的文件原先是否以P参数存盘。这样，利用BASIC解析程序本身的解密功能，实现对P参数存盘文件解密。

## 二、实 现 办 法

目前的磁盘BASIC和扩展BASIC中，有一些版本将涉及到ROM BASIC，因而实现解密的方法也有所不同。调用ROM BASIC的COM文件可使用第一种方法，与ROM BASIC无关的EXE文件则可采用第一或第二种方法。

1. 运行一个使文件类型标志字节值归零的服务程序SVS：在读入欲解密BASIC文件前，先调入SVS程序并运行之，然后调入各解密文件，再用Call命令调用SVS带入的复零子程序。

SVS源程序清单：

```

10 dim a(5): restore 30
20 for i=0 to 5: read a(i): next : go to 40
30 data 198, 6, 100, 4, 0, 203
40 def seg=&H7000: for p=0 to 5: poke p, a(p): next
50 end

```

2. 用debug修改BASIC解析程序，具体步骤如下：

A>ren gwbasic.exe gwbasic

A>olebug gwbasic ✓

- e940 a 00 00 ✓

- w ✓

- g ✓

A>ren gwbasic gwbasic.exe ✓

A>

经修改后的gwbasic将始终具备解密功能，当将P参数文件调入内存并列表时，将不会出现“非法函数调用”提示，而立即列出被保护的BASIC文件程序文本。



### 三、结 束 语

本文介绍的方法主要是从BASIC解析程序中设计的文件保护环节入手的，文件类型标志字节位于BASIC的数据段内，由BASIC自行建立而与硬件无关。对不同版本的BASIC这个工作单元的位置仅稍微有些不同。因而，只要硬件能支持BASIC运行，上述的修改都是有效的。只不过相应的修改要针对这一工作单元而进行。到目前为止，笔者在不同版本的BASIC解析程序上作过的修改获得了成功。

文献出处：《计算机世界》月刊1987年10期41—42页

## APPLE 微机解密小程序

曹 小 多

许多加密软件采用在文件目录中加入不显示字符（按CTRL+字母键产生）的方法使使用者无法读出程序，笔者编制的以下小程序可将文件名中的不显示字符转变为相应的小写字母而显示出来，可使真实的文件名一目了然。读者不妨一试。

```
10 FOR I=768 TO 783: READ A:POKE I, A: NEXT  
20 POKE 43603, 0: POKE 43604, 3  
30 PRINT CHR$(4) "CATALOG"  
40 DATA 201, 141, 240, 4, 201, 160, 144, 3, 76, 240, 253, 105, 192, 76,  
240, 253
```

文献出处：《电子与电脑》 1987年1期48页

## 自制 BASIC快速解密程序

吉 林 张 蒸

去年以来，有关报刊曾陆续登载过一些 BASIC 加 P 存贮程序的解密方法，虽然都能解密，但操作比较繁琐，且比较费时，而有的只能解小程序，稍大的就会出现丢失或误解，这里介绍的解密软件，人人都可以十分容易地制作出来，利用这个软件，无论多大的程序，只 1 秒钟就可解密，该程序可以在 IBM—PC、0520 及各种兼容机上使用。

### 一、制作方法

(1) 用屏幕拷贝 COPY CON 的方法造出一个文件，名为 P.BAS，只打入一个回车

之后就存盘，该文件占两个字节。

(2) 用 DEBUG 对这两字节的文件进行修改，即将回车符0D、换行符0A分别改为FF和1A，存盘后退出，解密软件就造出来了。

## 二、用法

在 BASIC 状态下，将一个加P存贮的BASIC 程序调入内存，假定此文件名为 AAA.BAS，再用LOAD去调P.BAS，这时AAA.BAS立刻就变为可读的了。

下面是软件制作和解密的全过程

注：^Z为Ctrl+Z

```
C) COPY CON P.BAS ↵
```

```
↵
```

```
^Z ↵
```

```
C) DEBUG P.BAS ↵
```

```
-D CS:100 101 ↵
```

```
XXXX:0100 0D 0A
```

```
-E 100 ↵
```

```
XXXX:0100 0D FF ↵
```

```
-E 101 ↵
```

```
XXXX:0101 0A 1A ↵
```

```
-W ↵
```

```
Writing 0002 bytes
```

```
-Q ↵
```

```
C) GWBASIC ↵
```

```
LOAD "AAA.BAS" ↵
```

```
LOAD "P" ↵
```

```
LIST ↵
```

文献出处：《软件报》1988年3月19日

## 对BASIC程序加密的一种新方法

天 津 黎汉生

我根据多年的编程经验，剖析了 BASIC 的解释程序，找到了一种新的加密方法，在此介绍给大家。此方法需做一项准备工作，即用 DEBUG 程序建立一个加密文件，假定名为 S.BAS 以备加密时调用。加密方法是这样的：在 BASIC 状态下，先调入需加密的程序 X.BAS，再调入加密文件 S.BAS，最后存盘。整个加密方法如图所示。注意一点，加密文件 S.BAS 只需建立一次，以后加密时调用就行了。

本方法操作简单，加密后的程序不能用LIST命令列表，运行同未加密一样，很难破密，有兴趣者不妨一试。

C) DEBUG

-E 100 FF FF FF FF FF

-R BX

BX 0000

:100

-R CX

CX 0000

:5

-N 5.BAS

-W

Writing 0005 bytes

-Q

C) BASICA

LOAD\*X

OK

LOAD\*S

OK

SAVE\*X

文献出处：《软件报》1988年3月19日

## 制表软件 OFFICE 1.00A 的解密

湖 北 孟小华

某部开发的汉字制表软件OFFICE1.00A版是进行了加密处理的，其1号盘一般的拷贝程序无法拷贝，且在软件启动时，A驱动器必须插入1号原盘，否则无法进入系统给用户使用带来诸多不便。经分析其命令文件OP.COM发现，在OP.COM中增加了两次额外的软盘I/O中断INT13，专门用来判断A驱动器是否放有带有特殊记号（4个扇区为特殊扇区）的1号盘，若有则装入系统继续运行，否则退出运行返回到DOS。下面给出了用DEBUG程序进行解密的具体操作步骤：

1. 将DEBUG.COM和OP.COM拷入一子目录。
2. 用DEBUG.COM修改OP.COM。
3. 将2号盘拷入子目录试运行；若启动成功则进行第4步，否则检查操作是否有误和查看OFFICE是否为1.00A版。

4. 将OP.COM拷回1号保存或者用COPY命令生成一份已去密的1号盘。

进行上面的解密操作后，OFFICE既可不要软盘直接启动，也能用DISK COPY或COPY命令随意拷贝。

```
C) DEBUG OP.COM
```

```
-E6011 90 90
```

```
-E6016 EB
```

```
-E60A1 90 90
```

```
-E60A6 EB
```

```
-W
```

```
Writing 9000 bytes
```

```
-Q
```

文献出处：《软件报》1988年3月19日

## 对加密汉字操作系统（9针小字）的去密

上 海 钱宗家

IBM-PC/XT 汉字操作系统（9针小字），能在FX-100等9针打印机上打印出2.5MMX1.5MM超密度汉字，当采用S字体时每行打204个汉字（408个ASCII字符），因此拥有9针打印机的用户大都采用这种操作系统。该系统是一个加密软件，把它拷贝到硬盘后，不能在硬盘上启动，给使用带来了不便。针对这个问题，我剖析了该操作系统，并编了一个去密程序，利用此程序可以对加密系统去密。去密后的操作系统各功能不变，并能用DISK COPY命令复制。

### 一、系统加密方法剖析

该系统同电子部六所的PCBIOS系统类似，是在PC-DOS 2.0的基础上对其中的文件管理系统（IBMDOS.COM）和基本输入输出系统（BIOS）扩充了汉字功能而成的。9针系统的主要文件是CHINESE1.EXE、CHINESE2.EXE和CL，其中CL是字库，CHINESE2.EXE为CL开辟内存，核心文件是CHINESE1.EXE。

CHINESE1.EXE是一个加密文件，它的重要代码并不在文件内，而是放在磁盘的备用磁道（28H）上，因此用普通的COPY或DISKCOPY命令并不能得到CHINESE1.EXE所需要全部代码，这样就达到了加密的目的。

CHINESE1.EXE文件首先执行加密子程序，启动A驱动器把0面28H磁道的数据读入起始地址CS:2CE5的内存，然后开始执行CS:2D00为入口的主程序，把CL调入内存，建立汉字字库和一套汉字输入与输出的管理方法。

搞清楚了系统的加密方法以后，就可以设法去掉系统的加密子程序。我编的去密程序，专门为该系统去密。

## 二、去密操作步骤

首先在A驱动器上起动9针系统，然后按以下步骤操作（见打印件，划线部分由键盘输入）：

1. 把PC-DOS盘放入A驱动器，把空盘片放入B驱动器，再 FORMAT B: /S 命令格式化空盘片，并把DEBUG.COM文件拷贝到B盘。

2. 把9针系统盘放入A驱动器，拷贝4个系统文件到B盘。

3. 用COPY命令在B盘上建立4个去密文件。这时要注意每行结束必须紧跟回车符，行尾不能有空格，特别是EDIT3文件的3行只有一个回车符。

4. 执行去密程序。这时A驱动器中必须放入9针系统盘（已在A驱动器中）。

至此，B盘中已经是去密后的9针操作系统，它可以被拷贝，也可以被装入硬盘，以后就可以在硬盘上起动9针系统了。

以上的操作也可以直接在硬盘上进行，这只要把操作命令中的“B:”换成“C:”就可以了。但需注意，格式化C盘将破坏C盘中原有的全部文件，因此要预先作好备份。

## 三、去密程序简介

该去密程序由4个文件组成，主要文件是EDIT.BAT，这是一个DOS批命令文件，批命令文件所需要的数据分别由EDIT1~EDIT3三个文件提供，下面简单介绍一下该批命令的功能。

第2~3行修改CHINESE1.EXE的加密子程序，使它能在DEBUG状态下执行，第4~5行在DEBUG状态下执行加密子程序，把A驱动器28H磁道的数据读入内存，并复制到磁盘临时文件DAT中保存。第6~8行把DAT中的数据装入CHINESE1.EXE文件中，并去掉加密子程序。第9行删除磁盘临时文件DAT。

(1) A)

FORMAT B: /S

Insert new diskette for drive B:

end strike any key when ready

Formatting...Format complete

System transferred

362486 bytes total disk space

40960 bytes used by system

321536 bytes available on disk

Format another (Y/N)? N

A) COPY DEBUG.COM B:

1 File(s) copied

(2)

A) COPY AUTOEXEC.BAT B:

1 File(s) copied

A) COPY CHINESE?.EXE B:

CHINESE1.EXE

CHINESE2.EXE

2 File(s) copied

A) COPY CL A:

1 File(s) copied

(3)

A) R:

B) COPY CON EDIT.BAT

```

ECHO OFF
REN CHINESE1.EXE CHINESE1
DEBUG CHINESE1 < EDIT1
REN CHINESE1 CHINESE1.EXE
DEBUG CHINESE1.EXE < EDIT2
REN CHINESE1.EXE CHINESE1
DEBUG CHINESE1 < EDIT3
REN CHINESE1 CHINESE1.EXE
DEL DAT
^Z
    1 File(s) copied
B) COPY CON BDI1
E1388
90 90
W
Q
^Z
    1 File(s) copied
B) COPY CON EDIT2
G226
RCX

```

```

0200
NDAT
W2CE5
Q
^Z
    1 File(s) copied
B) COPY CON EDIT3
A5A4
JMP 300
NDAT
L2FE5
RCX
D000
NCHINESE1
W
Q
^Z
    1 File(s) copied
( 4 )
B) EDIT

```

文献出处：《软件报》1988年2月20日

## 五笔字型操作系统探讨

新疆 李国军

### 一、简介

五笔字型操作系统（简称WBZX—DOS）是运行于IBM—PC/XT/AT以及0520—CH等各式兼容机的汉字信息处理系统，它和汉字操作系统CC—DOS兼容，也是在MS—DOS的基础上扩充了汉字处理功能而实现的。本文讨论4.3和4.0版本，以4.0版为主。另外，WBZX操作系统约占400KB内存，它分软盘系统和硬盘系统，其系统文件和外部命令分装在两个盘片中。

### 二、系统组成

以下为WBZX—DOS4.0和4.3版的系统组成清单，其中括号内的系统文件名为4.3版本

与4.0版本的不同之处。

IBMDOS.COM

IBMBIO.COM

COMMAND.COM

LHZ.EXE

WBZXZZ.DAT

WHZ.EXE

LCH.EXE

WBZXCH.DAT

WCH.EXE

MS-DOS 系统文件

WBZX造字软件

WBZX造词软件

WB.EXE

DAO.EXE

ZHENG.EXE (WJP.EXE)

(WJP.RLM.WJP.TOU)

ZHANG(WZK)字库

DBLD.EXE

IX.EXE

2024 P16.EXE

3070 P16.EXE

NK9400.EXE

AUTOEXEC.BAT

WBZX

主体

打印模块

其中WHZ.EXE和WCH.EXE分别为造字、造词菜单管理程序，只有在WBZX-DOS启动后才能运行使用。

### 三、软盘系统和导硬盘系统文件的异同之处

软盘系统和导硬盘系统4.0版，除了两个系统文件DAO.EXE和ZHEG.EXE稍有差异外，其它文件完全相同。

软盘系统的上述两个文件在对字库查找、装入等操作时，以指定盘符和路径的形式执行，即按“B:\ZDZ\NY\_DOS\ZHANG”的形式进行操作，而导硬盘系统对应的两个文件，其路径名和文件名同软盘系统，但驱动器号由B改为C。另外，文件ZHE-NG.EXE中显示WBZX-DOS的版本、研制单位等注释部分，对应软盘系统和导硬盘系统分别设置有“软盘系统”和“硬盘系统”，即有软硬两字之别。

4.0版与4.0版基本类似，但4.0版对字库设置在主目录中，故对字库(WZK)的操作，仅以驱动器号加字库名的形式实施。

### 四、系统加密与目录隐含

WBZXSC.DOS ZDZNY.DOS

盘符

A: WB.EXE 2024 P16.EXE

IX.EXE 3070 P16.EXE

DAO.EXE

B: ZHENG.EXE ZHANG

DBLD.EXE

为确保版权所有和防止非法拷贝，WBZX操作系统本身采取了改写标准格式和文件名做花以及目录文件隐含等措施来加以保护。本文只对目录文件的隐含情况作一简单介绍。

1. 软盘系统4.0版的两个系统盘片各有两个子目录文件，其子目录名分别为WBZXSC.DOS和ZDZNY.DOS，至于系统文件所属子目录情况列表如下：

其中字库文件ZHANG本身也是隐含的，即文件属性为只读、隐形、系统。

2. 导硬盘系统4.0版的隐含子目录名为 ZDZ, 有兴趣者可分别进入该系统 A、B盘的 ZDZ子目录, 以查看各系统文件的出处。

3. 4.3版与4.0版类似, 详况略述。

## 五、系统的硬盘安装

1. 根据 WBX-DOS 的系统组成情况和子目录隐含情况, 一一将主目录和子目录下的系统文件拷贝到硬盘。

2. 对于4.0版, 其字库应拷在硬盘子目录ZDZNY.DOS下, 否则要相应修改系统文件 DAO.EXE和ZHENG.EXE中的字库路径名。若无此子目录, 则先建立并进入该子目录, 尚能拷贝字库ZHANG。

3. 若无导硬盘系统, 直接用软盘系统往硬盘上安装, 根据本文“三、软盘系统和导硬盘系统文件的异同之处”提出的两点差异, 使用调试程序DEBUG改之即可。

4. 建立自动执行批文件AUTOEXEC.BAT以启动硬盘WBZX-DOS系统。例如4.0版除了字库在子目录下外, 其它均在主目录上, 其批处理内容如下:

```
ECHO OFF                                WB
CLS                                       DAO
ECHO Please wait.                       ZHENG
LHZ WBZXZZ.DAT                          DBLD
LCH                                       IX
```

· 若造字、造词功能不常使用, 可去掉LHZ至LCH的两行处理内容, 同时也可将对应打印机的驱动打印模块列入批处理之中。

## 六、系统的启动

WBZX-DOS的启动与CC-DOS一样, 是在MS-DOS启动成功之后, 通过执行批处理文件来实现的。但由于WBZX-DOS占用过多的内存空间, 对于512KB内存的主流微机来说, 在该系统下运行一些大的应用软件, 常出现内存不足的现象。同时, 五笔字型除了汉字输入功能较强外, 其它功能与CCDOS一样, 所以何时使用何种系统启动, 应根据处理任务决定。对此本文提出如下硬盘启动方法供参考。

假定硬盘上有CC-DOS和WBZX-DOS两个汉字操作系统文件, 它们的批文件分别为CCDOS.BAT和WBZX.BAT, 且无自动执行批文件AUTOEXEC.BAT。这样, 每次冷、热启动只启动MS-DOS, 若要启动CC-DOS, 键入批文件CCDOS便可, 同样, 键入WBZX便可启动WBZX-DOS。

## 七、使用说明

有关五笔字型的输入方法及输入模式切换等有关事宜请参考王永民同志编著的《五笔字型计算机汉字输入技术》成套用户资料三本, 即《培训教材》、《五笔字型码本》和《五笔字型用户手册》。

文献出处: 《软件报》1988年4月16日



# APPLE— I 超级软汉字系统

## DOS2.0的一个错误

李 顺 启

北京大通咨询公司开发的APPLE— I 超级软汉字DOS2.0 (64K) 系统有一处错误, 这个错误导致了下面的现象: 当内存中存有一定量的字符串变量后, 再输入汉字时, 将不能正确地从此盘上读入汉字。

通过对系统分析发现, 出现上述情况的原因是系统为了调用RWTS来读取字库盘而设置的IOB表中有一处出错, IOB表中有两个字节的内容为机械特征表(DCT)的地址, 该系统将这个地址写为\$B7FB, 在冷启动该系统后, \$B7FB处有一个机械特征表, 但系统把这个区域定为字符串变量存放区, 所以, 只有少量的字符串输入时, 这个地址及其后的几个未被更新, 系统可正常工作, 一旦有较多的字符串输入到内存时, 这个区域的内容被刷新, 原来的那个机械特征表被破坏, 使得系统不能从汉字库盘上读出信息。

改错的方法很简单, 只要将内存\$1FD6单元中的\$B7改为\$F7就可以了。这样机械特征表的地址成了\$F7FB, 在那里有DOS所设置的机械特征表。具体做法是: 对于加了密的系统, 在每次冷启动后执行一个POKE 8150, 247即可, 对未加密的系统盘除上述方法外, 还可采用一劳永逸的做法, 对系统文件进行修改, 在DOS3.3的环境下:

- (1) 找到系统文件名, 这里设为(FILENAME)
- (2) 键入BLOAD(FILENAMC)(CR)
- (3) 用CALL—151进入监控系统, 打入1FD:F7再用CTRL-C退出监控系统
- (4) 最后BSAGE(FILENAME)A \$800, L24830(CR)。

文献出处: 《软件报》1987年12月2日

# 使用27916KEPROM实现软件加密

南京无线电厂 张天石

随着微机技术的不断发展,软件在微机系统成本中的比重日益增长,许多厂商出于他们的经济利益对软件进行加密处理,限制用户非法拷贝,以保护公司利益。但是,尽管采取这样或那样的加密手段,还是被一一破译,造成巨大经济损失。

Intel公司利用它生产LSI电路的优势,在1984年底推出了“上锁”EPROM(KEPROM) 27916。使用它,可实现“永久性”的软件加密。这种芯片对用户是透明、可靠、容易推广的,对非法复制软件和防止非法进入系统利用电脑作案具有重要意义。本文将介绍利用27916进行软件加密的原理和方法。

## 一、27916KEPROM的性能

Intel公司生产的KEPROM27916是与27128兼容的16K字节的EPROM。

27916最大的特点是软件保密功能。凡是要进入一个KEPROM系统的用户,必须有一个含有KEPROM的设备,而且该设备的KEPROM中锁匙密码一定与所进入系统中KEPROM所设定的锁匙密码一致。锁匙密码是存放在一个64位长“只写”存储器内,该“只写”存储器是指一旦对27916配置完成后,写入的内容是不能再读出的。并且能存入 $18 \times 10^{18}$ 种锁匙密码,如此大的密码,对要窃取密码的非法入侵者来说是极其困难的。但是,锁匙密码对一个已有使用权的用户来说,既不要破密更不用去记住,因为锁匙密码已经写入设备的KEPROM中,所以,对用户是透明而又省事的,一开机便可进入系统。

但是,为什么锁匙密码不会出现在数据总线上呢?这是因为所采用的随机数字发生器是一种特殊电路,随机数字并不是用一种算法来实现的,根据它的原理,在对发生的随机数900000个采样中,99.97%的数只出现一次,即是对900000个随机数据来说,出现一次以上的概率才为0.03%,把这样的各不相同的随机数与锁匙密码组合后,根本不可能被破译了。为了进一步防止有人采用穷举法,不惜花费大量时间来逐一采集密码作分析,故在27916芯片中,还没有一个可编程延时计数器,它可以在27916配置时,用程序设置四种不同的速度进行双向鉴别信息交换,其速度分别为0.08、2、4和15秒四种。当延时计时器不作特定的编程时,初始的双向鉴别信息交换速度为最快。如果使用不同的程序设置,以不同的速度进行信息交换,将给以穷举搜索法窃密的人带来更大的困难。

## 二、27916KEPROM的加密原理

在使用KEPROM的系统中,至少要用两片事先处理过的KEPROM芯片。将一片安置在主机内,另一片在用户终端内。开机时,系统自动地由两片KEPROM进行类似的双向鉴别信息交换。当双方鉴别认为符合原来设定的密码时,终端上的用户才能使用系统,否则主

机拒绝用户使用系统。

KEPROM的上锁机构是怎样起到保密作用呢？请看。图中的“锁匙密码”是事先写在两个PROM内的相同的密码，每一片KEPROM中有一个随机数发生器，它能产生几乎永远不同的各种代码。每一片KEPROM中有一个加密电路，这一电路按照一定规则，把锁匙密码与随机数组合成一个新的代码。这样，在两片KEPROM之间交换的数据也是一个随机数，而锁匙密码数据永远不可能直接出现在系统总线上。只要KEPROM的锁匙密码写好后，从整个KEPROM来看不可能被仪器所读取，从而达到高度保密。

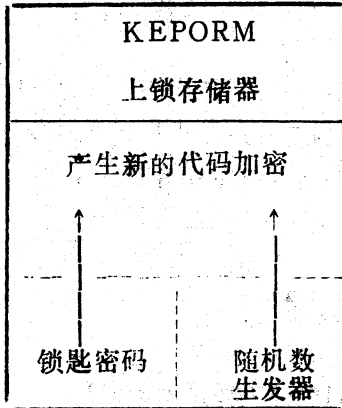


图 KEPROM的“上锁”机构

主机内的KEPROM与用户终端上的KEPROM之间是按顺序进行秘密接头——双向鉴别。

由此可见，KEPROM的双向鉴别信息交换过程就象两人秘密接头那样进行双向两次鉴别无误后，才能进一步交换情报。KEPROM在双向鉴别中，可能呈现出三种状态：

1. 发起状态，它发出随机数。
2. 接收状态，它对接收到的随机数加密。

3. 存储器状态，当在KEPROM双方鉴别密码成功之后，使地址0200H至3FEFH的内容可供用户访问。

必须指出，KEPROM27916的存储器空间为0000H~3FFFH，当它作为“上锁”使用时，地址0000H~01FFH及3FF0H~3FFFH被引导程序占用，随时供用户访问。但是，当它处于发起状态或接收状态时，引导区(0000H~01FFH, 3FF0H~3FFFH)之间的空间是用户不可读的，只有当它处于存储器状态时，地址0200H~3FFFH的内容才可供用户访问。

### 三、27916在软件加密中应用

从27916的内部结构来看，除了随机数发生器、锁匙、密码逻辑、控制和锁之外，其余部分跟EPROM结构是一样的。并且，它有14根地址线(A<sub>0</sub>~A<sub>13</sub>)，所以它可当作标准的27128EPROM来使用，片上有128K字节的存储容量，都是可以读出或用紫外线抹除重写的。但设计27916的目的是作“上锁”PROM使用，实现软件加密。

27916用作KEPROM时，可分为两种情况：①应用于简单的微机系统；②应用于复杂的微机系统，如多用户系统。不管是哪一种情况，都可以在通电后使其中一片KEPROM进入发起者状态，而另一片KEPROM进入接收者状态。

#### 1) 8位微机系统

当27916作KEPROM使用时，只要把27916芯片的第24脚(A<sub>5</sub>)接上11.5~12.5V高电平。27916芯片内的一个特殊的命令掩码寄存器，就会接受一定规则的命令掩码来确定27916是处于发起者状态还是接收者状态。在一个8位微机系统中，只要用两片KEPROM就可以

构成简单的锁匙密码管理系统。

## 2) 16/32位微机系统

27916用于多用户系统的锁匙密码管理系统，放在主机内的那片KEPROM放置了各用户的锁匙密码和引导程序，由它负责逐一地对各用户作双向鉴别信息交换的初始化工作，这种配置同样是按要求对命令掩码寄存器写入代码来实现的。而且写入这片命令掩码寄存器的锁匙密码的代码是不可抹除的，即此片KEPROM内的地址是1000H~2FFFH永远不可读取的，就在这些不可读的地址内存放1024个锁匙密码。

文献出处：《电脑应用时代》1988年3期43—44页

# 汉字码文件名解密方法

刘先荣

**摘要** 本文概述了按查找汉字码文件名所在区码号中设立等代效用汉字码文件名以破译汉字名文件的解密方法。从而论证了文献《简便的汉字码文件名保密方法》所述的保密手段存在着不容忽视的疏漏。

文献《简便的汉字码文件名保密方法》（《计算机世界》1986年第六期，总129期）介绍了成都电讯工程学院计算机系应用汉字编码原理而实施的汉字码文件名保密的方法，文献提出了“可以交替利用汉字的前后汉字码字节或利用多个单汉字的汉字码字节来提高保密程度”的可靠原理与简易可行的具体手段。

对于以汉字码命名的计算机文件，解密的前提自然是通过判定，检索其文件的汉字码命名名称。但设若作此计，则势将迫使解密者需在浩如瀚海的汉字码集中靠实现各种排列、组合与选择的巨大工作。这种方法不啻“海底捞针”，获得成功的概率低，很小。

本文所提出的方法与直接查找文件的汉字码命名名称，实为殊途同归。其方法可概述为：按文献所述，作删除处理的汉字，都在“国际汉字字符集”中有其特定的“区”号。可使用该区的任一汉字，并保留前汉字码字节，亦即意味该“汉字码文件名”中相应汉字的单汉字码字节被保留的手段，从而能顺利破译“汉字码文件名”。兹将本方法分述于下：

## 一、汉字机内码的特点

在CCDOS中，汉字被当作字符处理，每个汉字与ASCII码均按十六进制数的形式贮于机内。但因ASCII码字符占用1个字节，而汉字却占用2个字节。且每个字节之最高位均被置为1，以与ASCII码区别，例如：

ASCII码字符中的“C”字符，在机内表示为42(H)，占一个字节；而以纯中文方式存贮的字符“C”，在机内被表示为A3(H)C3(H)，共占二个字节。若化为二进制，则为：

10100011

11000011

每个字节的最高位永远是1。按此区别标志，则凡任一字节的最高位为1，系统便视之为汉字码字节。

## 二、汉字机内表示与汉字“区位码”的关系

由前例不难看出：汉字机内表示即以其两个汉字码字节的十六进制数与汉字所在“国标字符集”中的区位码之间存在一定的关系。例如：

ASCII码中空格的机内表示为20(H)，化为二进制，则为，

00100000

将其最高位置为1得

10100000

其十进制为160，化为十六进制为A0(H)。

纯中文方式存贮的空格(SP)需用两个字节，于是纯中文方式存贮的空格(SP)的机内表示为A0(H)A0(H)；而空格(SP)在“国标汉字字符集”中的“区位码”为0000；

再如：汉字“交”在机内表示为BD(H)BB(H)。将第一个字节BD减去A0得1D(H)，即十进制数29，此为“交”字位于“国标汉字字符集”中的“区”号；同法，将BB减去A0得1B(H)，即十进制数27，此为“交”字位于“国标汉字字符集”中的“位”号；故：“交”字在“国标汉字字符集”中之区位码应为2927；

如上述，不难看出下面的规律：

汉字机内表示的第一个汉字码字节减去A0，其十进制数为该汉字在“国标汉字字符集”中之区号；第二个汉字码字节减去A0，其十进制数为该汉字在“国标汉字字符集”中之位号。

## 三、汉字码文件名保密方法的破译

综上所述，对按《汉字码文件名保密方法》所建立的汉字码文件名可按下列步骤破译之：

1. 使用DEBUG调试程序，找出与汉字码文件名生成有关的汉字码字节，并记录其机内码；
2. 将此机内码化成的十六进制数各减去A0(H)，并将结果化为相应的十进制数以获得“国标汉字字符集”中的一区号；
3. 使用该区内任一汉字，同时需删除其后汉字码字节。

按上述程序执行的结果是：我们可由原汉字码文件名所在“区位码”得到另一汉字码文件名，这样做，则可完全避免对原汉字码文件名以检索、查找、组合方式进行劳而无功的“破译”，极易获得对使用《文献》所述的文件名设密后的文件的破译结果。

# MicroVMS和VMS的文件保护系统

中国科学院沈阳计算所 赵国泰

**摘要** MicroVMS和VMS的文件保护系统是VMS操作系统很有特色的一部分，随着时间的推移和VMS版本的升级，VMS又增添了一项新的文件保护手段——ACL（存取控制列表）。本文以较小篇幅扼要地介绍VMS文件保护系统的两个重要手段：UIC（用户识别码）和ACL。

文件保护机构是增强系统安全的极重要的工具，任何一个操作系统对它都十分重视。VMS提供两个主要地保护机构。一个是基于UIC的标准保护，这个保护机构根据用户范畴SYSTEM、OWNER、GROUP和WORLD控制存取。第二个保护机构是以ACL为基础的保护，由ACL指出允许或者拒绝某个用户或某组用户的某种访问。但是这两种文件保护机构的适用范围是不尽相同的。几乎所有用户都会遇到标准的UIC文件保护机构。某些用户使用存取控制列表。ACL适用于VMS所有用户的重要文件，通常对系统保密要求比较高的介质采取这种措施。

## 一、概 述

### 1. 以UIC为基础的保护

系统中的每个用户进程都由 AUTHORIZE 实用程序在 UAF 文件中建立用户标识码（UIC）。另一方面，系统中的每个目标（通常目标指文件、目录和设备）又都与一个UIC相关（一般为创建者的UIC），系统中的每个目标又都与一个保护屏蔽相关。根据用户UIC和目标UIC之间的关系决定该用户有哪种类型的访问。

这里首先解释一下用户UIC和目标UIC间的关系。在VMS上把按用户UIC和目标UIC之间的关系把用户进程分为四类：即系统、主人、组和全体。

系统进程——是指哪些进程的UIC在八进制的1~10之间的进程或者有SYSPRV（或者GRPPRV）特权的进程。

主人——UIC与目标UIC相同的进程。

组——进程UIC的组号与目标UIC的组号相同的进程。

全体——进程的UIC和目标的UIC有或没有共同之处。

如果一个用户进程是系统型的进程，那么它一定具有主人型、组型和全体型关系。如果一个用户进程是属于主人型的进程，那么它一定具有组型和全体型的关系。如果一个用户进程是属于组型的进程，则它一定具有全体型的关系。

进程访问目标的方式有如下几种：

- Read（读）——读一个文件；读磁盘文卷。
- write（写）——写一个文件；写磁盘文卷。
- Execute（执行）——执行某映像文件。

• Delete (删除) —— 删除文件。

我们再解释一下什么是保护屏蔽？保护屏蔽有四个字段，每个字段有四个指示符。每个字段表文一种所属关系。它们分别代表SYSTEM (简称S)、OWNER (简称O)、GROUP (简称G) 和WORLD (简称W)。可能的四个指示符分别是：R(read), W(write) E (execute) 和D (Delete)。

目标保护必须由下列格式指定：

(所有关系 [ : 访问方式 ], ...)

在下例中，系统关系可全面访问；主人关系可全面访问，但不能删除；组和全体关系无访问方式。

(S: RWED, O: RWE, G, W)

当一个用户希望访问一个目标 (譬如一个文件或者卷) 时，通常系统都要检查UIC保护码。系统把用户UIC与目标UIC进行比较，看用户是属于哪一个范畴的，如果比较结果表明用户是系统范畴，则该用户对该目标具有系统型的访问方式。如果比较结果表明用户是属于组型范畴的，则该用户对该目标具有组型的访问方式。对主人型和全体型用户亦然。

## 2. 以ACL为基础的保护

访问控制表 (ACL) 指出允许或拒绝某个用户或某组用户的某种访问类型。这是一种新的文件保护方法，此方法的核心是权力数据库，权力数据库规定标识符、标识符的持有者以及ACL。而ACL建立了是允许还是拒绝标识符持有者的访问关系。

ACL由一组访问控制列表项 (ACE) 组成。每一个访问控制列表项 (ACE)。定义了是允许还是拒绝对一个实际系统目标，诸如文件、目录、设备的访问。我们比较 ACL 保护和UIC保护就会发现，UIC保护比较适用于一般用户的保护，也就是通常我们说的四大类，即系统型用户、主人型用户、同组型用户和全体型用户。目标 (或文件) 对不同类型的用户授予不同的访问方式，以达到保护的目。所以说，UIC保护的通用性比较强，保护面比较宽，但其灵活性和针对性是比较差的、UIC面不能针对某个用户或某些用户对一个文件规定访问方式。而ACL保护确有比较大的灵活性和针对性。

我们大家都知道，对以UIC为基础的保护设置缺省是可能的，对安全管理员来说设置缺省ACI也是可能的。甚至会出现这样的现象：某些用户把根本就不知道他们的文件有ACL，也从来没有想过要修改他们自己的ACL，但他们文件的ACL又确实存在。

当系统收到一个有ACL的目标访问请求时，系统从 ACL 的第一项开始搜索，以期在ACL 中找到与目标访问请求的匹配项。当找到第一个匹配项时，则搜索停止。而下面的ACL匹配项不起作用。所以，在ACL中，标识特指定用户的ACE应该放在标识“大路”用户的ACE之前。

ACL是一个选件。使用ACL 可以提高安装系统目标的安全性。因为它可以详细地定义谁允许访问，它允许什么样的访问，借以提高系统的可靠性，但也增加了消耗在建立和维护ACL的用户时间，以及执行ACL强制功能所需要的处理时间。

ACL由一个或多个ACE组成，对ACE的数目没有限制。但是ACE 的数目太多，势必造成ACL太长，势必增加批准一个目标访问的平均时间。

有三种类型的ACE：

- Identifier (标识符) ACE——控制一个实际用户或一组用户所允许的存取类型
- Default protection (缺省保护) ACE——为目录定义缺省保护,因而该保护可以扩散到该目录下创建的所有文件和子目录。
- Security alarm (安全报警) ACE——当按指定的方式访问一个目标时,它提供一个警报消息,告诫管理员安全处理。

ACE的正确格式随类型不同而不同,但是所有的ACE都用括号括起来。ACE的一般格式是:

(type [, options] [, access\_to\_grant] )

## 二、以 UIC 为基础的保护

### 1. 用户标识码 (UIC) 的组成

UIC由两部份组成,即组和成员。标识码的格式如下:

[组、成员]

UIC既可以是数字的,也可以是字符的。

- 数字UIC——即组和成员由数字组成。组号的范围是八进制的0~37776。成员号的范围是八制的0~17776。
- 字符UIC——由成员名和组名组成。但在字符UIC中,成员名和组名不是对等的,成员名是必备的,并且在整个系统中成员名互不相同。组名是选项,可有可无。通常,成员名是AUTHORIZE Utility的ADD命令的用户名,而组号由ADD/ACCOVNT命令指定。成员名和组名的最大长度为31个字符。

不管你用哪一种格式,系统把UIC转换成一个32位的值表达一个组号和一个成员号;高16位表示组号;低16位表示成员号。32位的数字UIC保存在系统权力数据库中,权力数据库是一个文件,它包含属于访问权力与标识符属性的信息。

字母 UIC 的成员号在整个系统中必须唯一。因此在系统上,不可以有两个这样的UIC [Group1 JONES] 和 [Group2, JONES],因为在系统上,成员JONES只能有一个相关的UIC。当系统转换字母 UIC 时,既包括组名,也包括成员名,系统的长字与成员名相关;系统以成员名为背景检查组名。

当你进入VMS时,你的进程UIC已被指定。当然,你还可以用SET UIC命令改变UIC,根据缺省,通常目标采用创建进程的UIC。

### 2. 进程UIC和目标UIC

当用户想要访问一个目标时(例如,文件或卷),差不多所有的情况都要检查UIC保护码。这主要是通过检查比较用户的UIC和目标拥有者的UIC实现保护的。为了比较好的理解保护码的规则,我们首先讨论一下用户进程的分类。VMS把用户进程分为四类,即,SYSTEM,主人,同组和全体。

SYSTEM——1.所有有系统特权(SYSPRV)的用户。

2. 组号在1到10(八进制)之间的用户。当然这个值是一个变值,系统管理员可以改变它。



3. 有用户特权GRPPRV并且用户的组号与目标主人的组号相同的用户。
4. 对于磁盘文件, 那些与文件所在卷主人的UIC相匹配的UIC用户。  
 主人——与创建和拥有该目标的用户有同样UIC的用户。  
 同组——与目标的拥有者有同样组号的所有用户。  
 全体——所有用户, 当然也包括以上三类用户。

图1表示了以上四类用户进程的相互关系。

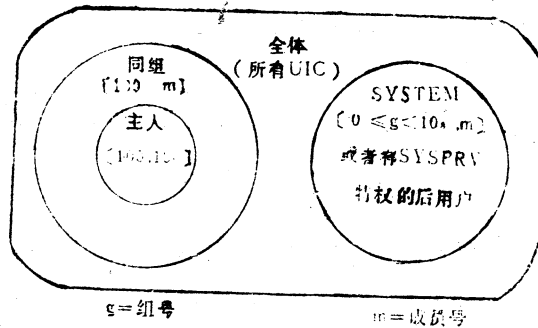


图1 UIC [100, 100] 的用户类型图解

当你进入VMS时, 用户进程的UIC已被指定。即采用 AVTHORIZE 实用程序的ADD命令的/UIC限定词指定的UIC。当然在进入系统之后, 你可以使用SET UIC命令修正VIC。例如, 系统管理员为用户JONES创建一个如下的帐号。

```

$RUN AUTHORIZE
UAF>ADD JONES—
  __/PASSWORD=—
  __/UIC[014, 006]—
  __/OWNER="ROCKET, JONES"—
  __/DEVICE= $DISK1—
  __/DIRE CTORY= {JONES}
UAF>EXIT
  
```

那么用户JONES的UIC就是 [014, 006], 当用户JONES进入 VMS 时, 也就指定了 JONES的UIC= [014, 006]。

我们再回过头来看一下目标UIC的和目标的保护码。

目标的缺省UIC——我们在为用户建立了一个帐号之后, 还应该为用户建立一个缺省目标。譬如上例, 系统管理员已为用户 JONES 建立了帐号, 那么可以用下面的命令建立缺省目录。

```

$CREATE/DIRECTORY $DISK1: {JONES} —
  __ $/OWNER_UIC= {014, 006} —
  __ $/PROTECTION=(S:RWE, O:RWE, G:RE, W:RE)
  
```

我们建立的目录名为 {JONES}, 即用户名和目录名相同。此目录下文件的缺省 UIC 为 [014, 006]。保护码为 (S:RWE, O:RWE, G:RE, W:RE)。即允许系统和主人完全访问你的目录 (不包括删除)。拒绝同组和全体用户对该目录的写访问和删除。

#### 4. 保护系统文件

为了保护系统文件，通常系统文件的UIC是〔001, 004〕。系统文件只允许全体型进程读和执行，但对下列文件不允许全体型进程进行任何方式的访问：

```
SYS $SYSTEM:PAGEFILE.SYS
SYS $SYSTEM:SWAPFILE.SYS
SYS $SYSTEM:SYSDUMP.DMP
SYS $SYSTEM:SYSUAF.DAT
SYS $SYSTEM:RMTNODE.DAT
SYS $SYSTEM:SYSALF.DAT
SYS $SYSTEM:NETUAF.DAT
```

这些文件在创建时就采取了适当的保护。只有以SYSTEM 帐号进入系统时，你才可以复制和修改这些文件。

#### 5. 建立和修改基于UIC的保护

##### 1) 建立和修改卷保护

当一个卷被安装时，VMS就要确定基于UIC的卷保护；可以缺省，也可以明确地规定。你可以用 \$INITIALIZE/OWNER\_\_UIC=uc，指定卷的拥有者的 UIC。缺省是你的缺省UIC。你也可以用 \$MOUNT/OWNER\_\_UIC=〔uc〕规定被安装卷的拥有者 UIC。如缺省，用卷初始化的 UIC 值。如果需要改变卷保护，可以使用 SET VOLVME 命令的 /OWNER\_\_UIC〔v=Uic〕限定词。

##### 2) 建立和修改目录保护

我们说的目录保护只适用于磁盘目录。通常目录的创建者在创建目录时用 /PROTECTION限定词指定保护代码，如果你忽略了保护，那么使用目录树上较高一级目录的缺省保护码。如果目录是顶级目录的话，则保护采用主文件目录（MFD）的保护。例如，我们可以创建如下的顶级目录文件MONRDE.DIR。

```
$CREATE/DIRECTORY/PROTECTION=(S:RWED,D:RWED,G:RWED,
W)
BOTANYDISK:〔MONROE〕
```

如果你希望改变目录〔MONROE〕的保护删除同组的访问，可以发出如下的命令：

```
$SET PROTECTION=(S:RWED, O:RWED, G, W) —
$_MONROE.DIR
```

##### 3) 改变和修改文件保护

如果你新建一个文件时，该文件从它所在目录上得到UIC保护码。

如果创建一个已存在文件的新版本时，则新版本文件接受上一版本的保护码。

当你复制文件时，可以使用/PROTECTION 限定词定义文件的保护，例如你可以发出如下的copy命令：

```
$COPY USE1:〔PAYDATA〕PAYROLL.DAT PAYSORT.DAT —
$_/PROTECTION=(S:RW, O:RWED, G:RW, W)
```

根据缺省, 用户在 [JONES] 目录下建立的文件, 它的保护码亦为 (S:RWE, O:RWE, G:RE, W:RE)。也就是说, 在创建一个新文件时, 也同时指定了新文件的保护码。

· 用DCL命令SET PROTECTION可以指定文件的保护码。例如, 对 DATAFILE.DAT文件发出如下的命令, 指定DATAFILE.DAT文件的保护码:

```
$SET PROTECTION=(S:RWED,O:RWED,G:RE,W)DATAFILE.DAT
```

也就是允许系统和主人的所有访问, 允许同组READ和EXECUTE 操作, 拒绝全体的所有访问。

用户进程本身有一个 UIC, 目标 (如文件) 也有一个 UIC。当用户希望访问某个目标时, 系统首先要比较用户的UIC和目标的UIC。比较的结果把用户划规到SYSTEM、主人、同组和全体中的某一个范畴中。譬如说, 用户的UIC= [014, 006], 目标的UIC= [014, 006]。比较两个 UIC 我们得知, 用户是属于主人范畴的。我们再根据主人类型的保护码与用户所希望的访问进行比较, 看用户所希望的访问是否合法。如果合法, 则系统批准用户的这次访问, 否则系统拒绝用户的这次访问。我们仍以上例说明之, 我们已经知道, 文件DATAFILE.DAT的保护码是 (S:RWED, O:RWED, G:RE, W)。假定用户 JONES 希望读DATAFILE.DAT文件。用户JONES是属于主人范畴的, 主人所允许的访问是: 读、写、执行和删除。读访问是合法的访问, 应该批准。所以系统批准这一访问。

### 3. 特权与保护

因为系统管理员是管理全局的, 所以系统管理员有权访问一切文件, 其中包括系统文件。有四种系统特权可以改变一个用户实际可以接受的访问, 可以不管实际ACL保护和UIC保护如何。这四种特权是SYSPRV、GRPPRV、READALL和BYPASS。如果某一个用户拥有这四种特权的一种特权的话, 那么上述的保护检查将有很大的不同。

**SYSPRV** 拥有 SYSPRV 特权的用户作为系统用户访问保护文件和其他目标, 以及改变文件和其他目标的保护。即拥有SYSPRV特权的用户是属于 SYSTEM范畴的。

**GRPPRV** 如果拥有SYSPRV特权的用户, 并且他的 UIC 组与目标主人的 UIC组相同的话, 则该用户拥有SYSTEM范畴的访问。

**BYPASS** 拥有 BYPASS 特权的用户接受对目标的所有访问, 而不管它的保护如何。

**READALL** 拥有READALL特权的用户可以接受对所有目标的READ和CONTROL访问, 而不管ACL和UIC保护是否拒绝访问。当然他也接受保护代码所批准的访问。

综上所述, 我们清楚地意识到, 拥有以上四种特权之一的用户是“危险”用户。因为这些用户手中有很大的权力, 这类用户利用他们手中的权力, 可以任意访问系统上的所有文件, 也就是说整个VMS系统对这些用户都是敞开的。那么这些用户也就有可能“破坏”VMS系统。例如, 对于拥有BYPASS特权的用户, 你就没有任何办法可以阻止他访问任何文件。所以系统管理员和安全管理员对特权的分配要特别小心。

上面的命令复制了一个新文件PAYSORT.DAT, 新文件的保护码是(S:RW,O:RW ED, G:RW, W)

当然你也可以使用SET PROTECTION命令改变已存在文件的保护。

#### 4) 建立和修改设备保护

由于不保存设备的UIC保护码, 所以每次自举之后, 你必须重新建立设备UIC的保护。譬如, 你可以用下面的命令设置设备保护码。

```
$SET PROTECTION=(code)/DEVICE device-name [:]
```

#### 5) 建立和修改逻辑名表保护

由于不保存逻辑名表保护, 所以每次自举时, 必须重新建立逻辑名表的UIC保护。你可以用\$CRELNT系统服务的保护变量置保护, 你也可以用如下的命令建立保护:

```
$CREATE/NAME_TABLE/PROTECTION=(code) table-name
```

你不可以修改已存在的逻辑名表保护。

#### 6) 建立和修改排队保护

你可以用INITIALIZE/QUEUE、START/QUEUE、或者SET QUEUE 命令的/PROTECTION限定词对排队置UIC保护。

### 三、以ACL为基础的保护

VMS提供一种新的文件保护方法, 称作ACL。这种方法的核心是权力数据库(right Database)。权力数据库规定了标识符和这些标识符的持有者, 以及标识符是批准还是拒绝标识符持有者的访问。权力数据库是一个文件, 它把用户与持有称作标识符的特殊名字联系起来。某些标识符表明用户的名字和UIC。

#### 1. 标识符和标识符的持有者

我们先不正面回答什么是标识符, 什么是标识符的持有者。我们先看一看标识符和标识符的持有者都起什么作用。假设系统有如下的一个访问矩阵。

对象 \ 目标	目					
	标	V	W	X	Y	Z
A			•			•
B			•	•	•	
C			•	•	•	
D		•	•	•	•	
E		•				

我们可以把这个矩阵写成两个表, 一个以对象为主的表, 一个以目标为主的表。它们分别是:

A:W, Z	V:D, E
B:W, X, Y	W:A, B, C, D
C:W, X, Y	X:B, C, D

D:V, W, X, Y	Y:B, C, D
E:V	Z:A

我们把以对象为主的表称作能力系统表。把以目标为主的表称作授权系统表。授权系统表也称为访问控制表(ACL)。我们仔细地观察一下两张表,就会发现目标B, C, D都可以被对象W, X, Y访问,我们用Q表示这件事, D和E都可以被V访问,我们用P表示此事。这样我们就可以化简这两张表。

A:W, Z	V:P
B:Q	W:A, Q
C:Q	X:Q
D:P, Q	Y:Q
E:P	Z:A

我们把Q和P称作标识符, 而把W, X, Y称作标识符Q的持有者。把V称作标识符P的持有者。由此我们可以看出, 访问矩阵中只有标识符, 而采用了标识符大大地化简了访问控制表, 因而也大大地缩短了搜索时间。但是它又增加了一张表, 即标识符与标识符持有者之间对应表。

## 2. 标识符的类型

标识符提供了指定ACL用户的方法。有三种类型的标识符: UIC标识符; 一般标识符; 系统定义标识符。

### 1) UIC标识符

根据UIC唯一地标识系统上的每一个用户。UIC既可以用数字格式的, 也可以用字母格式的。例如下面的UIC都是合法的:

{GROUP1, JONES}	(字母格式UIC)
JONES	(用户名)
{306, 210}	(数字格式UIC)
%X08001006	(十六进格式UIC)

### 2) 一般标识符

一般标识符是由安全管理员定义在权力数据库上的标识符, 用它来区别系统上的用户。一般标识符是1~31个字符, 其中至少必须有一个字母。一般标识符由安全管理员使用AUTHORIZE创建和赋值。

### 3) 系统定义标识符

系统定义标识符是当系统安装建立权力数据库时, 由系统自动建立的。系统定义标识符包括:

BATCH	由批处理作业发出的所有访问
NETWORK	由DECnet任务发出的所有访问
INTERACTIVE	由交互进程发出的所有访问
LOCAL	在本地终端联入的用户发出的所有访问
DIALUP	在拨号终端联入的用户发出的所有访问
REMOTE	通过网络联入的用户发出的所有访问

在联入期间, 用户自动成为这些标识符的持有者。

当你进入系统时，你在权力数据库中持有的标识符（包括UIC和你的系统定义标识符）作为进程的一部份被复制到权力列表中。

### 3. ACE类型

访问控制列表ACL由访问控制列表项ACE组成。需要保护的访问保护类型由ACE的类型确定。有三种类型的ACE。

#### 1) 标识符ACE (Identifier ACE)

Identifier ACE控制一个用户或者一组用户所允许的访问类型。标识符ACE的格式是：

( IDENTIFIER=Identifier [, option] [, access] )

identifier ACE的头一个字段由关键字IDENTIFIER后面跟以标识符组成。标识符包括上面叙述的三种标识符：即，UIC标识符、一般标识符和系统定义标识符。但通常比较器重系统定义标识符。以上三种标识符可以组合，对于多种标识符，标识符之间用(+)号隔开。例如如下的ACE：

( IDENTIFIER= {301, 25} +BATCH, ACCESS=READ )

说明批作业用户 {301, 25} 希望批准读访问。

虽然许多用户不可能共享同一个UIC，但是多个用户共享同一个一般标识符是可能的。拥有同样标识符的用户不一定需要是同一组的用户。但是一个用户拥有多个不同的一般标识符是完全可能的。

identifier ACE的option字段控制是否ACE可以被扩散，显示或删除。option可以取如下四种值：

- DEFAULT——指出该ACE适用于在目录上建立的所有文件。此选择项仅对目录文件有效。
- PROTECTED——该选择项指出当想删除整个ACL时，将保存该ACE项。如果需要删除被保护的ACE的话，需要使用ACL编辑程序或者ACL命令。
- NOPROPAGATE——当ACL从一个版本向高版本复制时，或者从父目录向新建立的目录复制时，NOPROPAGATE选择项指出该ACE不能扩散。
- NONE——指出该ACE没有options。

如果你规定多个选择项的话，选择项之间用(+)号隔开。

identifier ACE的第三个字段是ACCESS。它指定所允许的访问类型。共有六种访问类型：

READ——访问者可以读文件

WRITE——访问者可以读或者写文件

EXECVTE——访问者可以执行一个文件

DELETE——访问者可以删除一个文件

CONTROL——访问者与目标拥有者有同样的特权

NONE——访问者对此目标不能进行任何访问

我们举一个ACL的例子来说明identifier ACE。

例：

( IDENTIFIER=SECURITY, OPTIONS=PROTECTED, ACCESS= R+W +  
D+C)

( IDENTIFIER= PERSONNEL, ACCESS=R+W+E+D )

( IDENTIFIER=SECRETARIES, ACCESS=R+W )

( IDENTIFIER= {20, •}, ACCESS=R )

( IDENTIFIER=NETWORK, ACCESS=NONE )

( IDENTIFIER= {SALES, JONES} , ACCESS=NONE )

在上面的例子中, SECURITY, PERSONNEL, SECRETARIES 是一般标识符, NETWORK是系统定义的标识符, 而 {20, •}, {SALES, JONES} 是 UIC标识符。

目录的 identifier ACE

目录的 identifier ACE与其它文件的 identifier ACE 没有本质区别, 只不过需要令 OPTIONS=DEFAULT。缺省ACE支持在目录上新创建的文件; 不支持以前已存在的文件。

例如, 对目录文件MALCOLM.DIR有如下的ACE:

( IDENTIFIER=PERSONNEL, OPTIONS=DEFAULT, ACCESS=READ+  
WRITE )

这一ACE项的含义是: 你希望拥有标识符PERSONNEL的用户对在 {MALCOLM} 目录下创建的文件可以READ和WRITE。

### 2) 缺省保护

为目录定义缺省保护, 因而保护可以扩散到目录下创建的文件和子目录上。所以这种类型的ACE只适用于目录文件。

缺省保护ACE的格式是

( DEFAULT-PROTECTION [, OPTIONS=PROTECTED], protection-  
mask )

protection-mask应该是UIC保护码, 例如你可以对目录 {MALCOLM} 建立如下的ACE:

( DEFAULT-PROTECTION, :SRWED, O:RWED )

既允许SYSTEM和OWNER对 {MALCOLM} 目录下的文件READ、WRITE、EXECUTE和DELETE。

### 3) 安全警报ACE

如果发生了某种访问类型的话, 则安全警报ACE允许你请求发出一个安全警报(消息) 发送到安全操作员终端。我们都知道, 了解目标被访问的情况是很重要的, 安全警报ACE就提供了这种方法。安全警报ACE是否有效取决于是否使用了DCL命令SET AUDIT。如果没有这一步骤的话, 那么安全警报就绝不能送到操作员终端。

安全警报ACE的格式是:

( ALARM-JOURNAL=SECURITY [, options] [, access] )

安全警报ACE的第一个字段是ALARM-JOURNAL, 由它指出ACE的类型。

安全警报ACE的第二个字段是options, 它可能采用如下值: DEFALUT、PROTECTED、NOPROPAGATE和NONE。其含义与Identifier ACE的options的解释基本相

同，这就不重复了。

安全警报ACE的第三个字段是ACCESS，由它控制引起警报的访问类型。ACCESS可能取如下的值：READ、WRITE、EXECUTE、Delete、control、success和Failure。前五个值的意义在前面我们已经介绍过了。下面我们着重介绍一下success和Failure。

success——如果你选定了访问类型READ、WRITE、execute、Delete和(或)Control访问的话，则访问成功产生一警报。

Failure——如果你的访问类型是READ、WRITE、execute、Delete和(或)Control的话，则访问不成功产生一警报。

请注意，为了使警报生效，你必须在警报ACE中包括success或Failure或者两者兼而有之。因此，如果需要对成功地写访问或者删除一个文件产生警报的话，你应该这样指定ACE。

(ALARM-JOURNAL=SECURITY, WRITE+DELETE+SOCCCESS)

#### 四、ACL表的次序

我们已经知道，ACL保护是系统通过把目标ACL和访问者持有的标识符相比较，寻找符合的项，达到保护的目。那么ACL中ACE的次序不同，实现的保护也不同。

假定目标EXAMPLE.DAT的ACL有如下的ACE：

(IDENTIFIER={200, 201}, ACCESS=R+W+E)

(IDENTIFIER={FRED}+BATCH, ACCESS=W+E)

(IDENTIFIER=PAYROLL, ACCESS=R)

(IDENTIFIER=DIALUP, ACCESS=NONE)

即，允许UIC为{200, 201}的用户对EXAMPLE.DAT文件进行读、写和执行访问。允许UIC为{FRED}的用户在进行批作业时，对目标可以进行读和执行访问。允许标识符为PAYROLL的用户对文件进行读访问。拒绝持有系统定义标识符DIALUP的用户进行任何访问。

系统在比较ACL时，从ACL表的第一项开始逐项向后进行。访问者持有的标识符必须与某项中的说明符完全相同才算匹配。访问者的访问方式由表中的第一匹配项决定，所以项的顺序是很关键的。例如，在上例中，如果把最后一项放在第一项上，那么所有的DIALUP用户将无任何访问方式。即，当说明符{200, 201}、{FRED}+BATCH和{PAYROLL}的用户作为DIALUP用户访问时，访问就不能进行。如果把最后一项，仍然放在最后一项的位置上，那么{200, 201}、{FRED}+BATCH和{PAYROLL}的用户作为DIALUP用户访问时，访问就能进行下去。

如果用户希望对某文件进行任何访问的话，那么就可以把禁止访问的ACE从ACL底部移到顶部即可。

#### 五、确定访问的步骤

如果一个用户被允许访问一个特定的目标，那么VMS遵守如下的步骤：

1) 如果目标有相关的ACL，则系统利用此ACL确定该用户是否允许对目标的访问。



如果ACL指定允许该用户的访问，则批准访问，并且不再作任何进一步的检查。如果ACL没有明确拒绝或允许访问，则系统使用UIC保护码确定访问。如果ACL拒绝访问，则仅SYSTEM, OWNER用户可以访问。

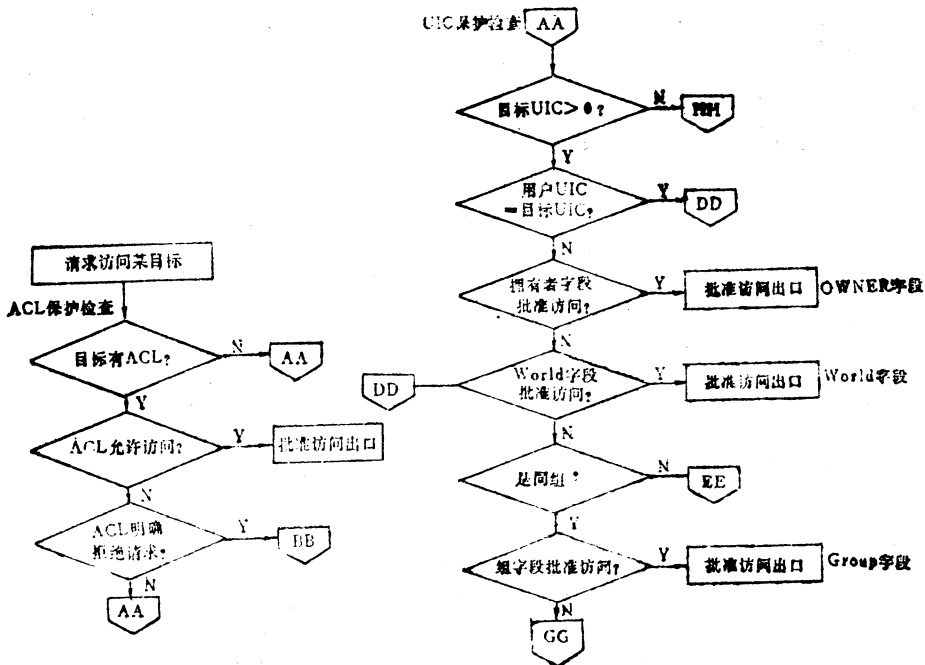
2) 如果目标没有相关的ACL，则系统使用UIC确定访问。拒绝或允许访问决定于用户UIC和目标UIC之间的关系。

3) 检查用户是否拥有GRPRV, SYSPRV, READALL和BYPASS特权，拥有上述四种特权的用户可以访问目标。

我们把上述的步骤再简化一下：

- 首先检查ACL
- 如果ACL规定的允许访问失败，则检查UIC保护。
- 如果ACL指定拒绝访问，但用户是SYSTEM, OWNER用户，则仍可以访问目标。
- 具有系统特权的用户可以访问目标，而不管ACL或者UIC保护码提供的保护。

下面我们以框图的形式解释系统实现保护的流程：



## 六、例子

假设ASSURANCE.MEM文件有如下特征：

① ACL

( IDENTIFIER=QAMANAGEMENT, ACCESS=R+W+E+D )

( IDENTIFIER=MADISON, ACCESS=R+CONTROL )

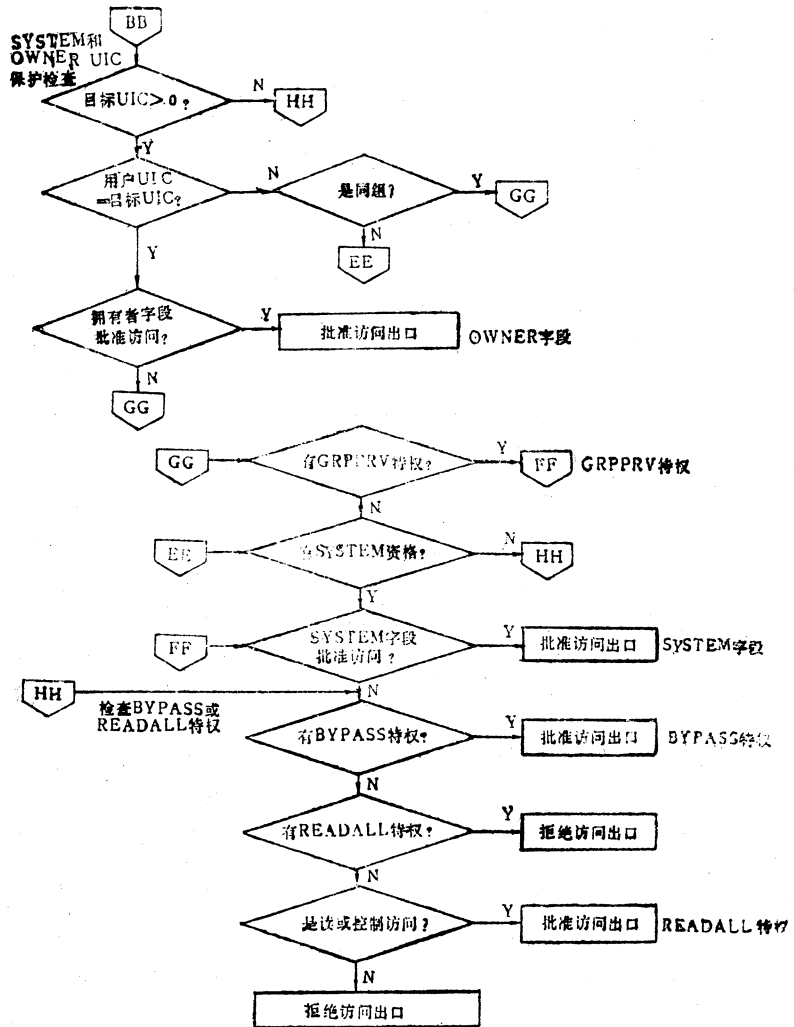
( IDENTIFIER=GRANGER, ACCESS=R )

( IDENTIFIER= [\*, \*], ACCESS=NONE )

② OWNER的UIC= [QAMGT, MORRIS]

③ 保护码 = ( S:RWED, O:RWE, G:R, W )

有如下五个问题，可以用来检验我们对ACL保护的理解决度：



1) MADISON用户希望打印然后删除ASSORANCE.MEM.

回答是否定的, 因为ACL只允许MADISON用户READ, 不允许DELETE.

2) 有SYSPRV特权的MORRIS用户希望删除它.

回答是肯定的, 因为SYSPRV特权有SYSTEM权力, 而SYSTEM有删除权.

3) UIC = [QAMGT, GRANGER] 并有GRPPRV特权的用户希望写文件.

回答是肯定的, 虽然ACL拒绝写, 但是GRPPRV特权有SYSTEM权力, 而SYSTEM有写访问权力.

4) UIC = [QAMGT, NONROE] 用户希望删除ASSURANCE.MEM文件.

回答是否定的, 因为MONROE用户既不是SYSTEM用户, 也不是OWNER用户, 所以只能与ACL的ACE = [\*, \*] 项相匹配, 而ACE = [\*, \*] 的项拒绝任何访问, 所以访问被拒绝.

5) 拥有QAMANAGEMENT标识符的WALLACE用户希望删除该文件.

回答是肯定的, 因为WALLACE用户持有QAMANAGEMENT标识符, 而后者拥有DELETE权力. 所以批准访问. 参考文献(略)

文献出处《小型微型计算机系统》1987年11期37—49页

# IBM-PC软磁盘文件的恢复方法

浙江大学 杨 长 生

**摘要** 本文介绍了恢复被删除的软磁盘文件的方法, 以及利用本文介绍的方法恢复被删除了的IBM-PC系统中的软磁盘文件的具体实现过程。

## 一、概 况

通常, 计算机用户都会把一些重要文件复制到其它的记录介质上作备份。这样, 一旦由于某种原因, 使用的文件丢失, 或者被破坏了, 还可以使用保存在其它记录介质上的那些文件副本。然而, 由于某种原因, 用户没有为某些重要文件保留副本, 加之操作错误, 而把某些重要文件删除了。那么, 往往会给用户带来重大的损失。因此, 在这些场合, 把已经被删除了的文件恢复起来是十分紧迫的。本文介绍了如何恢复被删除了的软磁盘文件信息的方法。

如何恢复被删除的软磁盘文件的信息, 与磁盘文件的管理系统, 文件的记录方式, 以及磁盘空间的管理方式密切相关。本文第二节介绍了五吋软磁盘的文件组织, 以及信息格式。第三节则介绍了IBM-PC软磁盘文件的管理方式, 以及文件在软磁盘上的记录方式。接着, 在第四节描述了恢复软磁盘上被删除了的文件信息的方法, 以及具体的操作过程。

## 二、五吋软磁盘的文件组织

记录文件是由操作系统中的文件管理系统来管理的。文件管理系统能把有关的数据组成文件并给于文件名称。

一般说来, 在IBM-PC软磁盘系统上存在三个基本文件: 文件目录, 文件分配表, 以及盘图文件。记录在软磁盘上的文件信息包括文件名, 文件的地址, 存贮方式, 用户名; 口令等等。这些信息也就构成了文件的分配表(或文件控制块)FAT(File Allocation Table)。而文件目录表又是由文件分配表FAT组成(如图1所示)。

文件目录的存放磁盘的某个固定位置, 供系统访问文件时检阅。用户需要使用文件中的信息时, 只要给出文件名, 用户名, 口令等信息。文件管理系统通过查找文件目录以及有关的文件分配表, 就能知道文件在软盘上的地址等参数。从而可以调用系统的一些子程序把文件的数据信息读至内存, 供用户使用。

盘图文件是磁盘中存在的一个很重要的文件。它是对盘区空间进行分配的依据。文件中记录了每个磁道上的扇区的当前使用情况。当需要把某个盘区分配给一个文件时, 为了保证充分利用磁盘空间, 并且不把原来的有用信息复盖掉, 就必须查阅盘图文件, 以决定是否可以对这些盘区进行分配。反之, 当把一个文件删除时, 文件管理系统也将修改盘图文件, 把曾经分配给这个文件的所有的盘区所在磁道上的使用情况重新记载, 以便这些盘区可以分配

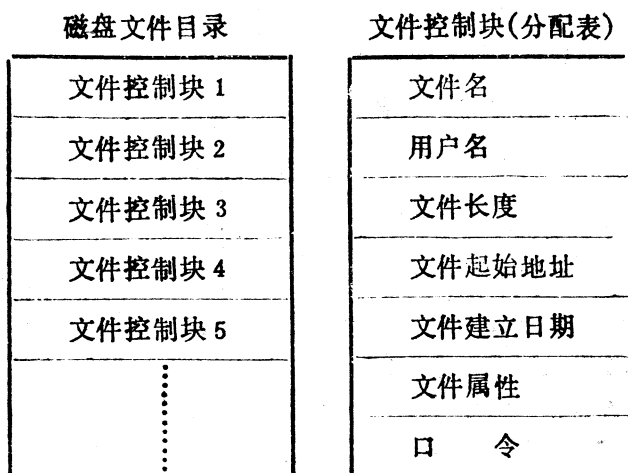


图 1 文件目录与文件分配表

给其它新文件记录信息。

总之，操作系统中的文件管理系统对软盘上文件的建立，修改，删除都是以文件目录表文件分配表和盘图文件作为依据的。一般说来，恢复软盘上的被删除了的文件信息，必须从文件目录表，文件分配表，以及盘图文件入手。

### 三、IBM-PC的DOS软盘文件管理

IBM-PC的磁盘操作系统(DOS)对软磁盘上文件的管理是以软盘上的文件目录表和文件分配表FTA为基础的。在系统对软盘格式化时，它的文件分配表与文件目录表被初始化了。下面将分别介绍IBM-PC的文件目录及文件分配表的结构。

#### 1. IBM-PC DOS磁盘的文件目录表

在IBM-PC DOS系统中，软盘的文件目录表存在软盘0面0道6扇区(单面软盘则为面0道5扇区)开始的盘区中。其目录数最多为112(单面软盘则为64个)。每条目录包含32个字节，其格式如下：

第0~7字节 为文件名。文件名最多为8个字符，并且是大写字母(即使在创建文件时，用户给出是小写字母，系统也自动把它转为大写字母)。文件名的第一个字节表示了文件的状态。如果一个文件被删除了，那么它的第一个字母被改写为小写字母“e”(即十六进制数E5)。

第8~10字节 为文件的扩展名。

第11字节 表明文件的属性。(如只读文件，系统文件，隐性文件等等)。

第12~21字节 为备用字节。

第22~25字节 指示了文件被建立的时间，即建立文件的年、月、日、时、分、秒。

第26~27字节 指明了文件信息在软盘上的起始记录块的地址。

第28~31字节 则说明了文件中所含信息的总字节数。

文件管理系统通过查找磁盘文件目录表，就可以知道文件的状态、属性、以及在软盘上记录的起始地址。

## 2. IBM-PC DOS的软盘文件分配表FAT

文件分配表FAT是文件管理系统用来给每个软盘文件分配盘区空间的表格。该表在磁盘上有两个文本，分别存放在磁盘的0面0道2扇区和0面0道4扇区。

在文件分配表中的一个项目对应于软磁盘上的一个记录簇\* (Cluster)。操作系统中的文件管理系统以簇为单位把盘区空间分配给文件。文件分配表中的头二个项目指出了文件目录在磁盘上的位置。从第三个项目开始则记录了磁盘上各簇的状况。不同的记录状况具有不同的意义。文件分配表的示意图为图2所示。文件分配表中从第三项开始(即软盘的第二簇)的以下项目,表示的意义如表1所示。

文件分配表

簇号	项目内容
0	文件目录区
1	文件目录区
2	使用情况或下一簇指针
3	使用情况或下一簇指针
⋮	⋮

表1 文件分配表中项目内容的意义

项目内容	意义
0 0 0	对应的簇可以使用
FF8~FFF	当前文件在盘中所用的最后一簇
FF7	对应的簇是坏的,不可使用
×××	除了上面三种内容外,任何十六进制数,表示下一簇号

图2 文件分配表示意图

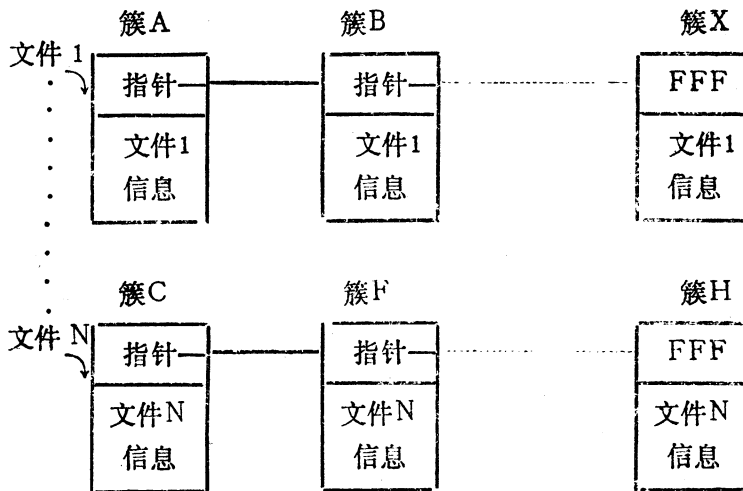


图3 文件记录的连接表结构

实际上, IBM-PC DOS系统中所用的软盘文件在软盘空间的映射是使用了一组连接表,如图3所示。文件1的记录所在的起始簇号则从文件目录中查获。该文件信息所在盘上的下一个簇号则从文件分配表中对应于簇号的项目内容给定。依此类推,直至文件分配表中项目内容为FF8~FFF之间的十六进制数为止。另外,文件分配表中的项目内容也指明了盘

• 双面软盘每一簇包含两个扇区,单面磁盘中每一簇为1个扇区

区的哪些簇是自由的，可以用来记录新文件的数据。除此之外系统中还存在一个自由空间连接表，如图4所示。当文件K被删除时，文件管理系统则把自由空间中的最后一个簇号的指针，修改为指向文件K的第一个簇号（即簇D），同时在软盘的文件目录表中把该文件名删去。实际上把文件名删去，只是把文件名的第一个字母改写英文的小写字母“e”。

从IBM-PC DOS的文件管理过程中，可以明白，软盘文件的删除过程，仅仅是在磁盘文件目录表中作删除标记；并且把曾经分配给该文件的盘区回收起来，作为自由存储空间，以待分配给其它新文件。因此，只要在文件目录表中被删除标记的文件名没有被新的文件名复盖，原来文件所在的信息空间没有被新文件占据，那么在文件目录表中恢复文件名，进而恢复文件信息是不难的，下一节，将介绍恢复被删除文件信息的详细过程。

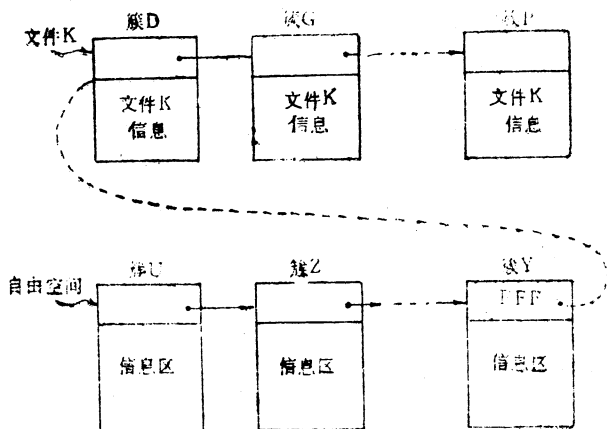


图4 自由空间连接表示意图

#### 四、文件信息的恢复过程

一般来说，恢复被删除的软盘文件的操作过程分为两个步骤：一是在软盘文件目录中恢复被删除了的文件名；二是把原来分配给该文件的磁盘空间，从软盘的自由空间连接表中分离出来。然而，要做到这些，往往要求用户比较熟悉IBM-PC DOS的文件管理系统。这对于一般用户来说也许有些困难。因此在下面我们介绍了一种简易的方法。

假定在软盘上有一个文件ABC.COM被删除了。现在打算恢复它的信息。则可作如下操作。

- 〈1〉用户调用DOS系统命令DEBUG，如下：A>DEBUG ↵ (假定当前使用A驱动器)
- 〈2〉在A驱动器中插入被删文件所在的软磁盘。
- 〈3〉在DEBUG命令的提示符“—”之下，作如下操作，从而把软盘的目录扇区信息读入内存。  
—L 100 0 0 8 ↵
- 〈4〉显示刚才所读入的信息。可作下面的操作：  
—D 0A00 1200 ↵

于是在计算机的显示屏幕上可出现类似上图5的信息。

```
08FF:0EF0 00 00 00 00 00 00 00 60-68 06 DC 00 00 06 00 00 ..... h.\.....
```

```

08FF:0F00 4D 55 53 40 43 20 20 20-42 41 53 20 00 00 00 00 MUSIC BAS.....
08FF:0F10 00 00 00 00 00 00 00 60-68 06 DE 00 00 22 00 00 ..... b, ^.....
08FF:0F20 44 4F 4E 4B 45 56 20 20-42 41 53 20 00 00 00 00 DONKEY BAS.....
08FF:0F30 00 00 00 00 00 00 00 60-68 06 E7 00 00 0E 00 00 ..... h, g .....
08FF:0F40 43 49 52 43 4C 45 20 20-42 41 53 20 00 00 00 00 CIRCLE BAS.....
08FF:0F50 00 00 00 00 00 00 00 60-68 06 EB 00 80 06 00 00 ..... h, k .....
08FF:0F60 50 49 45 43 48 41 52 54-42 41 53 20 00 00 00 00 PIECHART BAS...
08FF:0F70 00 00 00 00 00 00 00 60-68 06 ED 00 00 09 00 00 ..... h, m.....
08FF:0F80 53 50 41 43 45 20 20 20-42 41 53 20 00 00 00 00 SPACE BAS.....
08FF:0F90 00 00 00 00 00 00 00 60-68 06 F0 00 80 07 00 00 ..... h, p .....
08FF:0FA0 42 41 4C 4C 20 20 20 20-42 41 53 20 00 00 00 00 BALL BAS.....
08FF:0FB0 00 00 00 00 00 00 00 60-68 06 F2 00 00 08 00 00 ..... h, r .....
08FF:0FC0 43 4F 4D 4D 20 20 20 20-42 41 53 20 00 00 00 00 COMM BAS.....
08FF:0FD0 00 00 00 00 00 00 00 60-68 06 F4 00 00 11 00 00 ..... h, t .....
08FF:0FE0 47 57 42 41 53 49 43 20-45 58 45 20 00 00 00 00 GWBASICEXE .....
08FF:0FF0 00 00 00 00 00 00 9D 55-D5 06 F9 00 00 E0 00 00 ..... UU, Y.....
08FF:1000 52 53 20 20 20 20 20 20-43 4F 4D 20 00 00 00 00 RS COM.....
08FF:1010 00 00 00 00 00 00 DB66-16 07 31 01 A0 2F 00 00 ..... [f.. 1 .. / .....
08FF:1020 E5 42 43 20 20 20 20 20-43 4F 4D 20 00 00 00 00 eBC COM.....
08FF:1030 00 00 00 00 00 00 00 60-68 06 3D 20 00 00 00 00 ..... h, =.....
08FF:1040 00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6 VVVVVVVVVVVVVV
08FF:1015 F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6 VVVVVVVVVVVVVV
08FF:1060 00

```

图5 屏幕显示的内存信息

在这张图左端的项目中可以查到被删除的文件ABC.COM的目录项已成为“eBC...COM”。在这一项目所在行的最左端，则列出了删被标记“e”所在的内存地址为08FF:1020。下一步将是把这一地址的内容改为“A”（ASSII编码为41）。

〈5〉修改作删除标记的地址的内容。操作如下：

—E 08FF:1020 41 ↵

〈6〉把已经改写过的内容，从内存写回到软盘的目录区中去。操作如下：

—W 100 0 0 8 ↵

然后，让系统返回操作系统DOS。操作如下：

—Q ↵

至此，软盘中被删除了的文件信息已经能由系统命令调用了，即被删除了的文件ABC.OAM的信息已经被恢复了。然而，由于在以上的操作中没有把分配给该文件ABC.COM的盘区空间从软盘的自由空间表中分离出来，所以一旦系统写入新的文件时，很可能把文件ABC.COM的信息复盖了。为了防止这种情况的发生，一个简单的办法就是把已经恢复了的文件ABC.COM，复制到另外一片磁盘上去。接着再把该文件从原来的软盘上删除掉。最后，把复制在另一片盘上的文件ABC.COM重新复制到原来的磁盘上去。经过以上的操作，磁盘上

的文件ABC.COM就不会被其它文件复盖了。

## 五、结 束 语

本文所述的方法，已经在IBM-PC机上试验过，实验证明，本文所述的方法是切实可行的。用户只要按照本文所述的操作步骤去做，无须了解机器的文件管理系统，就可以恢复被删除了的软盘文件信息。

由于IBM-PC DOS的软盘文件的管理方法是一种普遍应用的方法。因此本文所述的方法对于恢复记录在其它介质上的文件信息也有指导意义。例如，如果打算恢复被删除了的IBM-PC硬盘上的文件信息，只要找到硬盘上的文件目录则可采用与本文所述的类似方法，恢复它的信息。

文献出处：《小型微型计算机系统》1986年1期61~65页

# IBM PC软磁盘文件恢复技术

西北纺织学院 王 永 发

**摘要** 本文分析了IBMPC软磁盘文件删除后再行恢复的可能性，提出了恢复的基本方法，讨论了由后向前的恢复策略，并介绍了磁盘文件恢复的有效工具HF.COM。

## 一、被删除文件恢复的可能性

被删除的文件能否得到恢复？根据文件在磁盘上的组织形式以及IBM PC DOS执行删除命令的过程，可以得出结论：在文件被删除以后，只要该磁盘上没有拷贝入新的文件，被删除文件的恢复在多数情况下是可能的；即使该磁盘上拷贝入少量新的文件，被删除文件的恢复在某些情况下也还是可能的。

### 1. 磁盘文件的组织形式

文件在磁盘上是如何组织的呢？我们知道，磁盘分为软磁盘和硬磁盘两类，这里我们以软磁盘为例。磁盘盘面上分为若干个磁道（Track），而每一条磁道又分为若干个扇区（Sector）。IBMPC微型机使用的双面双密度（2S2D）磁盘每一面上有40个磁道（同心圆），用0~39来标志其磁道号。磁盘的最外圈为0号磁道，最里圈为39号磁道。DOS 2.00以上版本在格式化时，把磁盘分为9个扇区，双面磁盘上共有720个扇区，其逻辑编号为0~719，分布在磁盘的两面。每一个扇区长度为512个字节，每一个字节可存贮一个字符。

为了使磁盘能够接受DOS文件，所有的新磁盘在使用之前都必须进行格式化，即由格式化程序（FORMAT.COM）来划分扇区，同时还准备该磁盘的自举记录（引导程序）、



盘文件分配表 (FAT) 和根目录区, 格式化后, DOS的磁道分配如图 1 所示。

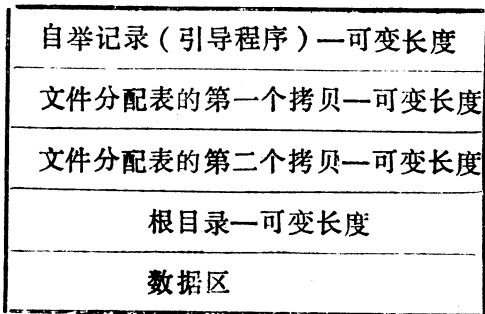


图 1 DOS磁盘分配

DOS以簇 (Cluster) 为单位给文件分配磁盘空间。双面软磁盘的一个簇包含两个连续的扇区。同一个文件的所有簇, 在文件分配表 (FAT) 中链接在一起。所以给一个文件分配的盘簇, 可以是一个连续的区域, 也可以跳过几个已被分配给其它文件的簇区, 在物理上并不一定连续。

自举记录为一引导程序, 用来把操作系统引入内存, 它存放在软磁盘的 0 号扇区。两个

文件分配表 (FAT) 是相同的。DOS给某个文件分配磁盘空间时, 就是使用这个文件分配表。每次分配一个簇, 直到某个文件存放完毕。一个文件分配表 (FAT) 存放在软磁盘的 1 号扇区和 2 号扇区, 另一个文件分配表 (FAT) 存放在 3 号扇区和 4 号扇区。

文件分配表 (FAT) 内的每一个表项目长度为 12 比特 (即 1.5 个字节)。每个表项目在文件分配表 (FAT) 中占一个表项目。文件分配表 (FAT) 的头两个表项目 (即前三个字节) 是表头, 用以说明磁盘的有关参数。对于双面 9 扇区的软磁盘, 第一个字节内容为 FDH; 第二、第三两个字节的内容总是 FFFFH, 由 DOS 设定。文件分配表 (FAT) 的表头部分如图 2 所示。

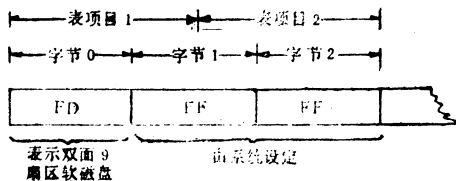


图 2 文件分配表 (FAT) 的表头

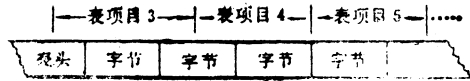


图 3 文件分配表 (FAT) 表项目 3 以后的逻辑排列

从文件分配表的第三个表项目开始, 每一个表项目用三位十六进制数作为一个簇的标志信息。这些表项目在逻辑上的排列如图 3 所示。

在图 3 中, 若表项目的内容为 000H, 则表示某一簇尚未用过, 或曾分配到这个簇区的文件已被删除。

若表项目的内容为 FF8H~FFFH, 则表示是某文件的最后簇。

若表项目的内容为其它位, 它指示出文件的 F 一个簇号。此文件的第一个簇号存放在根目录区内该文件对应的目录项中。

但是, 需要注意, 图 3 只是表示表项目的逻辑排列, 在物理上, 这些表项目并不是一个接一个连续存放下去, 而是以交错形式存放。要得到某一个表项目内的簇号, 还需要进行特殊的处理。

我们知道, 从文件的目录项中可以得到该文件在软磁盘上存放的起始簇号。为了确定这个文件的每一个后续簇, 须把每一个当前簇号乘以 1.5 (因为文件分配表 FAT 中每一个表项目长度为 1.5 个字节), 所得到的乘积可看作是在文件分配表 (FAT) 中从字节 0 开始的一个位移, 它将指向文件分配表 (FAT) 中的某一个字节, 从这个字节开始的连续两个字节 (即一个字) 对应于刚才所用的簇, 但却包含该文件 F 一个簇的簇号。文件分配表 (FAT)

就是利用这种方式把一个文件占用的所有簇盘链接在一起。

要得到这个簇号，可以把刚才计算出来的文件分配表（FAT）位移处的两个字节（即一个字）取出来，如果上一个簇的簇号是偶数，则取这个字的低12位，否则就取高12位。这样取得的12位数才是对应的逻辑表项目的内容。它可以是文件F一个簇的簇号，也可以是文件结束的标志（FF8H~FFFH）。

根目录区即磁盘文件目录存放区，它以目录项为单位，记载着属于根目录的所有文件的有关信息，它是DOS在磁盘上查找文件的依据。根目录区存放在磁盘的4扇区到11扇区，总长度为7个扇区。

每个目录项由32个字节组成。因为根目录区长7个扇区，每个扇区长512字节，由下式可知：

$$\frac{7 \text{ 扇区} / \text{根目录区} \times 512 \text{ 字节} / \text{扇区}}{32 \text{ 字节} / \text{目录项}} = 112 \text{ 目录项} / \text{根目录区}$$

根目录区可以容纳112个目录项。子目录作为一个文件存放在磁盘上，它的有关信息也由根目录区中的一个目录项来记载。由于软磁盘容量不大（360KB），存放的文件不可能太多，所以一般情况下不采用子目录方式。

每个目录项的32个字节，包括文件名、扩展名、文件属性、文件建立日期和时间，文件在磁盘上的起始簇号、文件长度等信息。一个目录项的组成情况如图4所示。

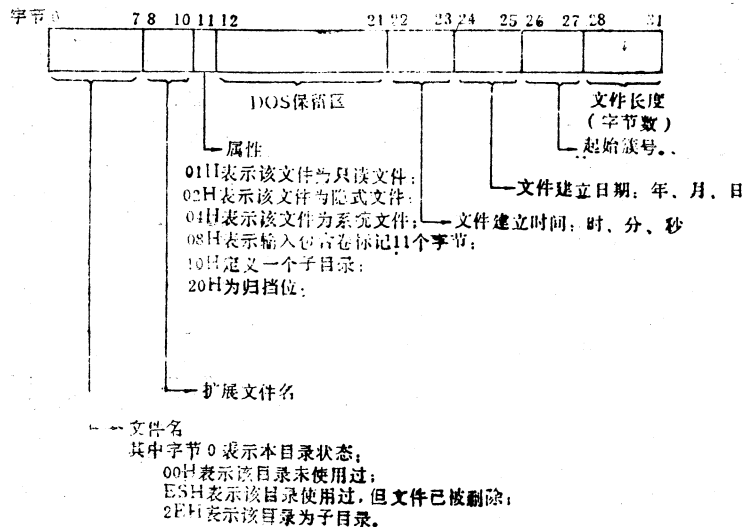


图4 目录项的组成

## 2. 文件删除命令的执行过程和被删除文件恢复的基本思想

DOS在执行文件删除命令ERASE或DEL时是否真的把指定文件的全部数据从磁盘上抹去了呢？不是。一个文件被删除后，它的数据并没有丢失，DOS只有在根目录区找到这个文件对应的目录项，把该目录项的第一个字节（即字节0）强行改变为E5H，以表示这个目录项曾经使用过，但文件已被删除，所以，在这个目录项中，除了丢失文件名的第一个字母以外，其它信息仍然存在；另外，DOS在文件分配表（FAT）中，把原分配给这个文件的簇号全部清0，以表示这些盘簇不再属于任何一个文件。以后，如果检查这个磁盘上的文件

时，DOS便不再列出这个被删除的文件；如果要拷贝入其它新的文件时，DOS就有可能把这个目录项和这个文件原先所占用的簇区重新分配给其他文件使用，从而彻底破坏原文件的数据。

由此我们容易想到，只要被删除文件的数据未遭到破坏，我们就有可能恢复这个文件。这里有两件事情要做。第一，在根目录区找到被删除文件对应的目录项，把DOS在删除这个文件时强行写入第一个字节（即字节0）的E5H再改回到原来文件的第一个字母的ASCII码，从而恢复该文件的目录项中的所有信息。第二，根据目录项中字节26~27中包含的起始簇号和字节28~31中包含的文件字节长度信息，利用DOS文件分配的算法重新建立该文件的文件分配表，填入磁盘文件分配表（FAT）中的相应位置，这就有可能把原文件所占据的簇区重新在文件分配表中链接起来，从而使被删除文件得到恢复。

## 二、被删除文件恢复的方法

显然，上面所提到的恢复目录项和重建文件分配表的操作超出了DOS常用命令的功能范围。我们可以利用DOS的功能调用或DEBUG程序读取和修改磁盘上指定扇区的有关信息，以达到恢复目录项和重建文件分配表的目的。这里，我们通过一个实例，来说明文件恢复的基本方法。

当一个磁盘被格式化后，它的文件分配表（FAT）和根目录区都处于初始化状态。文件分配表（FAT）的初始化状态如图5（a）所示，根目录区的初始状态如图5（b）所示。注意，图中只分别画出了它们的前32个字节。

(a)	FDFFF00	00000000	00000000	00000000
	00000000	00000000	00000000	00000000
(b)	00F6F6F6	F6F6F6F6	F6F6F6F6	F6F6F6F6
	F6F6F6F6	F6F6F6F6	F6F6F6F6	F6F6F6F6

图5 (a) 文件分配表（FAT）的初始化状态  
(b) 根目录区的初始化状态

当拷贝入一个文件F1.PRG（文件长度为5K字节）后，文件分配表（FAT）和根目录区第一目录项的状态分别如图6（a）和图6（b）所示。

(a)	FD FF FF 00	40 00 05 60	00 FF 0F 00	00 00 00 00
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
(b)	46 31 20 20	20 20 20 20	50 52 47 20	00 00 00 00
	00 00 00 00	00 00 D4 03	21 00 02 00	00 14 00 00

图6 (a) 拷贝F1.PRG以后，文件分配表（FAT）的状态  
(b) 拷贝F1.PRG以后，根目录区的状态

对比图 6 (b)和图 4 , 我们可以知道, 字节 0 ~ 7 是文件名, 即 F1. 字节 8 ~ 10 是扩展名, 即 PRG, 字节 26 ~ 27 是起始簇号代码 0200H, 由于高位字节在后, 故真正的簇号应为 0002H 即从第二簇区开始; 字节 28 ~ 31 为字节代码 00140000H, 同样, 由于高位字节和高位字在后, 所以真正字节长度为 00001400H 个字节, 亦即 5 K 字节。

这时, 我们用 ERASE F1.PRG 命令从磁盘上删除这个文件后, DOS 在文件分配表 (FAT) 和根目录区中所做的处理可从图 7 (a) 和图 7 (b) 容易地看出。

(a)	FD FF FF 00	00 00 00 00	00 00 00 00	00 00 00 00
	80 00 00 00	00 00 00 00	80 00 00 00	00 00 00 80

(b)	ES 31 20 20	20 20 20 20	50 52 47 20	00 00 00 00
	00 00 00 00	00 00 D4 03	21 00 02 00	00 14 00 00

图 7 (a) 删除 F1.PRG 以后, 文件分配表 (FAT) 的状态  
(b) 删除 F1.PRG 以后, 根目录区的状态

对比图 7 和图 6 , 我们可以看到, 文件被删除后, 在其目录项中, 只有第一个字节变为 E5H, 其余信息确实仍然存在, 但在文件分配表中, 有关文件 F1.PRG 的后续簇区分配信息却已完全丢失。

**恢复文件 F1.PRG 的操作如下:** (有关的 DEBUG 命令见附录)

1) 先查看根目录区 (从 5 号扇区到 11 号扇区), 找到被删除文件的目录项, 把第一个字节 (字节 0) 中的内容 E5H 恢复为 46H (字母 F 的 ASCII 代码)。

2) 由该目录项中, 可以知道文件 F1.PRG 在磁盘上存放的起始簇号为 0002H, 长度为 00001400H 字节, 即 5 K, 应占 5 个簇区 (每簇 1 K 字节)。

3) 由于这张软磁盘原来只有 F1.PRG 这一个文件, 它应分配的簇区是从 002H 开始的连续 5 个簇区, 即 002H, 003H, 004H, 005H, 006H。

4) 按照 DOS 在文件分配表 (FAT) 中簇号交错排放的算法, 把上述簇号折算成文件分配表 (FAT) 中的实际代码, 见图 8。

簇号	奇偶性	在 FAT 中的位数	位移处对应下一簇号 字位置	位移处对 应字内容	实际代码
002	偶	$2 \times 1.5 = 3$	字节 3.4	低 12 位	? 003 字节 3: 03 字节 4: ? 0
003	奇	$3 \times 1.5 = 4.5$	字节 4.5	高 12 位	004? 字节 4: 4? 字节 5: 00
004	偶	$4 \times 1.5 = 8$	字节 6.7	低 12 位	? 005 字节 6: 05 字节 7: ? 0
005	奇	$5 \times 1.5 = 7.5$	字节 7.8	高 12 位	006? 字节 7: 6? 字节 8: 00
006	偶	$6 \times 1.5 = 9$	字节 9.10	低 12 位	? FFF 字节 9: FF 字节 10: ? F

图 8 由簇号推算文件分配表 (FAT) 中相应字节的实际代码

因为文件F1.PRG占用5个簇区, 006H是最后一个簇区, 所以图8中最后一个表项目中应当是文件结束标记FFFH。

5) 把上面推算出来的文件分配表的实际代码从文件分配表(FAL)的第四个字节(即字节3)开始, 连续送入文件分配表(FAT)。

经1)~5)处理后的根目录区和文件分配表(FAT)的状态如图9(a)和图9(b)所示。

(a)	FD FF FF 03	40 00 05 60	00 FF 0F 00	00 00 00 00
	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

(b)	46 31 20 20	20 20 20 20	50 52 47 20	00 00 00 00
	00 00 00 00	00 00 D4 03	21 00 02 00	00 14 00 00

图9 (a) 恢复F1.PRG后, 文件分配表(FAT)的状态  
(b) 恢复F1.PRG后, 根目录区的状态

对比图9和图6, 我们可以看到, 在删除F1.PRG以前和恢复F1.PRG以后, 文件分配表(FAT)和根目录区的状态完全相同。

至此, 文件F1.PRG已经完全得到恢复。

### 三、文件恢复策略的讨论

由以上的分析可以看到, 如果磁盘上仅有一个文件被删除, 这个文件的恢复是肯定的, 可是, 在实际工作中, 一张磁盘经过长期使用, 文件的删除、拷贝的活动多次发生, 使磁盘的文件分配表(FAT)呈现出十分复杂的情况。

我们把文件分配表(FAT)设想成一条长链, 这条长链即是在文件在磁盘上分布的一个映象。每个磁盘文件在磁盘上占有一定的位置, 反映在这条长链上即占有一定的区段, 见图10。

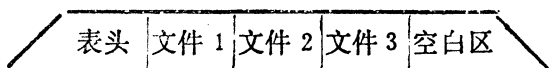


图10 文件在文件分配表中的分布

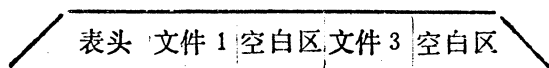


图11 删除文件之后的文件分配表

如果各个文件占用的区段互不交叉, 象图10表示的情况一样, 那么这些文件被删除后, 能够恢复是肯定的。因为, 每一个文件都有它自己的起始簇号, 相当于在文件分配表中的入口地址, 一经入口, 它的后续所有簇区一直顺序链接在一起, 直到该文件结束。这时, 当某一文件被删除后, 它所对应的区段变为空白。例如, 在图10所示的情况下, 我们删去文件2, 则文件分配表变为图11的情形。

这时，我们如果需要对文件 2 进行恢复，可以按照上一节的步骤，肯定能够成功。

我们考虑一下稍微复杂的情况。例如在图 11 所示的情况下，我们先拷贝一个比文件 2 较短的文件 4，由于文件 2 已被删除，DOS 令将文件 2 原先占据的簇区起始部分分配给文件 4，从而破坏了文件 2 的信息，文件分配表形成图 12 所示的情形。

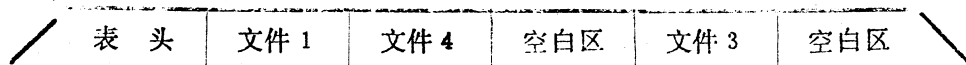


图 12 拷贝文件后的文件分配表

这时，再拷贝入文件 2。显然，在图 12 中，文件 4 与文件 3 之间的空白区域比文件 2 的长度要小。根据 DOS 给文件分配磁盘空间的算法，文件 2 将首先占据这一空白区域，其不足部分，将安排在文件 3 以后。于是，文件 2 占据了两个不连续区域，在文件分配表 (FAT) 中互相链接在一起，如图 13 所示。

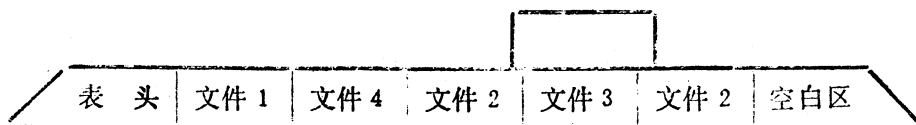


图 13 拷贝文件 2 后的文件分配表

在这种情况下，如果这四个文件全部被删除，能否将它们全部正确恢复呢？由图 13 可知，恢复文件 1、文件 4 和文件 3 是可能的，也是容易按照上节所述的方法实现的。但在恢复文件 2 时，就不仅要考虑文件 2 的起始簇号和字节长度，同时要考虑文件 3 的起始口号和字节长度。在为文件 2 分配簇区时，要扣除文件 3 占据的簇区，否则，文件 2 便不可能得到正确的恢复。由此我们得出结论，在进行文件恢复时，应该首先比较它们的起始簇号，由起始簇号较大的文件开始，采取由后向前的策略是合理的。因为，只有当后面的文件占据的磁盘簇区确定以后，恢复前面的文件时，才能正确地扣除分配给后面文件的簇区，从而正确地恢复前面的文件。

这种恢复策略适合于更加复杂的情况。如图 14 所示，当文件 2 被分配得更加零碎时，也可按照由后向前的顺序，使之得到正常恢复。在图 14 中，文件 2 被分割成三个部分，在恢复文件 2 时，要考虑扣除文件 5 和文件 3 所占据的簇区。

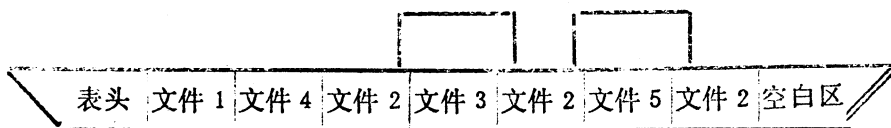


图 14 文件被分割成三部分的情形

实践证明，这种由后向前的恢复策略在大多数情况下，能够正确地恢复被删除文件。但是，应当指出，由于磁盘文件分配表 (FAT) 中有时会出现极为复杂的情况，例如图 15 所示的两个（或多个）文件在磁盘上交错存放，这将使得由后向前的恢复策略遇到困难。

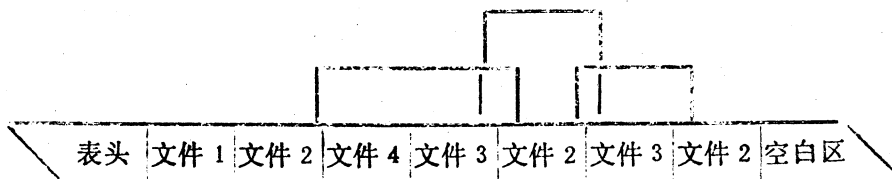


图 15 文件 2 和文件 3 交错存放的文件分配表

在图15中,我们看到,文件2被文件4和文件3分割成三部分,文件3又被文件2分割成两部分。这种情形使我们无法采取由后向前的恢复策略,因为文件2与文件3相比,不存在某一个文件具有恢复的优先性,因而呈现出互相依赖、互相制约的状态。在恢复文件2时,需要扣除文件4和文件3所占据的簇区;而要确定文件3所占据的簇区,又需要文件2的位置能够确定下来。显然,这是不可能的。在这种情况下,只有按照文件的起始簇号去搜索后续簇区,在文件可读性较强的情况下,我们仍有可能搜索到它所占据的簇区,再把它填入该文件相应的文件分配表,仍然可能恢复这个文件。如果文件的可读性较差,那么这些文件的恢复是很困难的。

**综上所述,在恢复一批被删除文件时,应当依循下述步骤:**

1) 查看目录区。列出目录区中记载的所有文件,包括被删除的文件和未被删除的文件,特别是各个文件的文件名、起始簇号和字节长度。这里需要注意,当一个文件被删除后,它所占据的簇区一经分配给其它文件,这个文件即使在根目录区某一个目录项中还残留着部分信息,除了第一个字节(字节0)已被修改成E5H以外,它的起始簇号和字节长度也全部清零。这样的文件自然无法恢复,不需加以考虑。

2) 从起始簇号最大者开始恢复。从其目录项中取出其起始簇号和字节长度,对应文件分配表中的相应位置,连续往后排放。遇到有被未删除文件占据的簇区时,则跳过这些簇区继续排放。这样,一般能够把原文件所占据的簇区正确地链接起来,从而使该文件得到恢复。

3) 已经恢复的文件与未被删除的文件一样对待。重复以上过程,自后向前逐个恢复,直到所有被删除文件全部得到恢复为止。

## 四、磁盘文件恢复的有效工具——HF.COM

使用DOS磁盘扇区读写功能调用,或运行DEBUG程序,需要对DOS有比较深入的了解。对于一般用户,建议使用磁盘文件恢复的有效工具——HF.COM。HF是“恢复”一词拼音的缩写(HuiFu)。HF.COM是XBey Soft-Tools(西北纺院软件工具)中的程序模块,能帮助用户方便地恢复被删除文件。它采用中文菜单提示方式,用户只需按照菜单提示键入相应的数码或字符,即可以进行文件的恢复工作。

HF.COM不仅能够用来恢复软磁盘文件,也能够用来恢复硬磁盘文件。程序内部已经考虑了由后向前的恢复原则,所以用户在使用时,无须过问文件的起始簇号和恢复顺序问题。此外,HF.COM还允许用户在磁盘上的任数位置修改数据,因此,它不仅是文件恢复的工具,同时又是软件开发的工具。HF.COM易学易用,是程序设计人员欢迎的软件工具。

但是,正如上文所述,HF.COM虽能帮助用户恢复绝大多数被删除文件,但并不能保证百分之百地正确恢复每一个被删除文件。对于磁盘文件分布极为复杂的状态,如何寻求正确的恢复策略,正是我们正在追求的目标之一。

附录 利用DEBUG程序进行磁盘绝对扇区读写的命令

1. 把装有DEBUG.COM程序的磁盘插入驱动器A,在DOS状态下,设当前驱动器为A,键入: A)DEBUG ↵

2. 把装有被删除文件的磁盘插入驱动器A。
3. 在DEBUG提示符“-”后键入:

-L 100 0 0 10 ↵

即把软盘A从0扇区开始的16个扇区的信息读入内存08FF:0100开始的区域。

(不同系统下,段地址08FF可能不同。)

4. 目录区从08FF:0B00单元开始,文件分配表第一个拷贝从08FF:0300单元开始,文件分配表第二个拷贝从08FF:0700单元开始,可用命令D显示这些单元的内容,用命令E改写。

键入: -D 0B00 0B90 ↵

则可显示08FF:0B00~08FF:0B90这一存储区域的内容。

键入: -E 08FF:0B00 46 ↵

则用数据46H代换08FF:0B00单元的数据。

5. 把在内存中所作的修改,写入磁盘的相应位置。

键入: -W 100 0 0 10 ↵

6. 修改完毕,要退出DEBUG。

键入: -Q ↵

文献出处:《电脑应用技术》1988年1期21—27转9页

## IBM—PC磁盘文件的恢复及文件属性的修改

张晋浙

计算机在使用过程中有时会因为操作人员的错误或偶然的疏忽,将一些没有保留备份的文件删除掉。一般来说,要将IBM—PC上已被删除的磁盘文件完整的全部恢复是相当困难的,有时甚至是不可能的。但在某些特定条件下,只需要在未进行任何新的写操作之前,则有可能全部或部分地将已被删除的文件恢复。本文在详细分析了IBM—PC磁盘文件的目录表和文件分配表FAT (File Allocation Table) 的结构与使用方法后,介绍了恢复被删除文件的方法。最后还介绍了如何修改一个文件的属性,用户可根据需要将自己应用程序中的有关文件属性置为只读或隐藏,以起到某种保护作用。

另外,本文中出现的数字如未特别说明一般均为十进制数,十六进制数则根据惯例在其后附以大写英文字母H或指明为十六进制。由于在分析及使用过程中经常要进行十进制与十六进制间的换算,建议最好准备一只具有十进制与十六进制转换动能的计算器(如SHARP EL—514)或一份换算对照表。



# 一、IBM-PC磁盘文件管理

## 1、DOS如何划分磁盘空间

当使用FORMAT命令对一个磁盘进行格式化时，除了对磁盘划分磁道和扇区外，还要在规定的磁道和扇区建立三个重需的区域。第一个区域在磁盘第一面的0道0扇区，称为引导记录。它的作用是在计算机启动时将系统盘上的两个系统文件IBM BIO.COM 和 IBM DOS.COM装入内存，在引导失败或对非系统盘则给出提示，告诉用户重新启动。第二个区域是文件分配表FAT，它的作用非常重要，所以每个磁盘上都有两份完全相同的文件分配表。在双面9扇区软盘上，每个FAT长度为两个扇区，起始相对扇区号分别为1和3。因为在DEBUG命令中要求使用相对扇区号（或称逻辑扇区号），故本文所指的扇区号均为相对扇区号，而不再指明是磁盘上的哪一面第几磁道几扇区。第一个区域是根目录，紧跟在第二个文件分配表之后，在双面9扇区软盘上是05H~0BH共7个扇区，从0CH开始是用户可使用的数据扇区。

对于其它型式的磁盘如8扇区软盘或硬盘，其空间分配方式请参见表1。表1中还给出了一些今后要用到的其它有关数据。硬盘的有关数据与分区大小、数目有关，本文仅讨论只有一个分区的10M硬盘。

## 2、文件目录表

文件目录表是存放文件目录的地方，每个文件的目录由32个字节组成，包含了有关该文件的重要信息，各字节的意义如下：

第0~7字节：文件名，第0个字节为以下值时有其特殊意义。

00H表示该目录项从未使用过，DOS在有关的操作中遇到00H后即停止向下继续检索。E5H表示该文件已被删除，DOS跳过该目录项继续向下检索。2EH表示目录表为一子目录表。除此之外出现的其它值为文件名第一个字符的ASCII代码。

第8~10字节：文件名后缀。

第11字节：表示文件属性，意义如下：

表1 与磁盘空间分配有关的数据

磁盘形式	FAT 起始扇区	每个 FAT长度	目录扇区	目录扇区数	目录项数	扇区数/ 每串	数据区 起始扇区
单面8扇区	1, 2	1	3~6	4	64	1	7
双面8扇区	1, 2	1	3~9	7	112	2	0AH
单面9扇区	1, 3	2	5~8	4	64	1	9
双面9扇区	1, 3	2	5~0BH	7	112	2	0CH
10M硬盘	1, 9	8	11H~30H	20H	512	8	31H

01 H只读文件，拒绝COPY、ERASE命令。

02 H隐藏文件，拒绝COPY、ERASE、DIR命令。

04 H系统文件，拒绝COPY、ERASE、DIR命令。

08H表示该目录项前11个字符为磁盘卷标，在FAT中没有入口，只能存在于根目录中，

10 H表示该文件名为一子目录的父名。

20H档案位，在建立或修改文件时，该值加到文件属性中表示关闭，对硬盘上文件使用BACKUP命令后，该位被打开，从文件属性中减去该值。档案位主要和BACKUP命令中的M开关及RESTORE命令中的P开关有关。

第12~21字节：保留未用。

第22~23字节：文件生成或最后一次更新的时间。

第24~25字节：文件生成或最后一次更新的日期。

第26~27字节：文件的起始串号，也是指向FAT的入口地址。第26字节是低位字节，第27字节是高位字节。当第0字节为2EH时表示该子目录表所在的串号，当第0字节与第1字节均为2EH时表示子目录表的父目录所在的串号（如果父目录在根目录中则串号为0000H）。

第28~31字节：表示文件长度，以字节为单位。第28字节为低位字节，第31字节为高位字节。

### 3. 文件分配表FAT

文件分配表FAT是磁盘上一个相当重要的区域，其中包含以下4个内容的信息。

(1) 磁盘性质(型式)

(2) 扇区封锁信息表。

(3) 扇区使用情况。

(4) 指示一个文件的后继串。

FAT中每一个表项的值由一个12位二进制数(三位十六进制数)组成，占1.5个字节。

第1~2字节总是FFH，第0字节(首字节)表示磁盘的型式，意义如下：

FEH 单面每道8扇区软盘

FFH 双面每道8扇区软盘

FCH 单面每道9扇区软盘

FDM 双面每道9扇区软盘

F8H 硬盘

从第3字节开始的第2表项表示磁盘上数据区各串的情况，数据区一定是从002H串开始的。第3~5字节为第2~3表项，表示磁盘上002H串和003H串的情况。表项号、串号、磁盘上的串域是一一对应的。应注意表项的值在字节中的位置，例如第4字节的一半属于第2表项，而另一半属于第3表项。具体分配时其高4位应属于第3表项的低4位，而其低4位则是第2表项的高4位。也就是说当第3~5字节的值分别为45H、30H、12H时，则第2表项的值为045H，第3表项的值为123H。为了不致搞错，建议在使用FAT时最好每三个字节用记号隔开，对中间位置的字节均按上述处理。

磁盘在格式化时发现损坏的扇区便在对应的表项中写入FF7H，表示该扇区所在串不

能使用。当表项的值为000H时表示对应的串是空的，可以使用。FF8H~FFFH表示该串为文件中最后一串，FF0H~FF6H表示保留的串。其它在表项中出现的值指示文件中的下一个串的串号，也是该文件在FAT中的下一个入口。

## 二、DOS如何使用文件目录表和文件分配表FAT

DOS在分配磁盘空间时是以串为最小单位进行的，而不是以扇区为单位。对于不同形式的磁盘，每串所拥有的扇区数目是不一样的（见表1）。对一个长度仅为1个字节的文件来说。在双面盘上要占用1024字节的空间，而在硬盘上就要占用4096字节的空间。读文件时，DOS首先在目录中找到该文件的目录项，根据目录项中第26~27字节的起始串号值求出文件的起始相对扇区号和和在FAT中的入口。起始扇区号与起始串号之间的关系如下（十进制）：

磁盘型式      起始扇区号S与起始串号C的关系

单面8扇区	$S = C + 5$
双面8扇区	$S = 2C + 6$
单面9扇区	$S = C + 7$
双面9扇区	$S = 2C + 8$
10M硬盘	$S = 8C + 33$

文件分配表中的入口地址（或称偏移）即为字节的顺序（从0开始编号）。具体求法如下（十进制）：

1、将串号乘以1.5。

2、若所得结果为一整数，比如说是6。则将FAT中第6个字节的8位二进制数作为 $b_7 \sim b_0$ 位，将相邻的下一字节的低4位作为 $b_{11} \sim b_8$ 位组成一个12位二进制数，这个12位的二进制数即表示对应串的状态。若所得结果是一小数为0.5的数，比如是7.5。则FAT中第7个字节的高4位作为 $b_3 \sim b_0$ 位，将下邻的下一个字节作为 $b_{11} \sim b_4$ 位组成一个12位二进制数。

写文件时过程正好相反，DOS首先在目录中检索是否有同名文件，若无则将该文件名写入第一个找到的作过删除标记的目录中，否则开辟一新的未用目录项。然后顺序检索FAT中的每个表项，找到第一个值为000H的表项后，将文件开始写入该表项对应的串中，同时将该串号写入目录项中第26~27字节。如果文件未写完，则继续向后寻找下一个空串，并将其串号写入上一个表项中。如果文件写完，则将该表项值置为FFFH。

一个文件可以是放在互相衔接的几个连续串中，也可能是断续放在几个不相邻的串中，中间隔着其它文件占用的串或空串。但不论哪种情形，一个文件所占用的串和串号一定是从小到大顺序递增的，除非该文件不是采用正常的拷贝命令和编制手段建立的。当用COPY命令将一些文件拷贝到一张空盘上时，所有的文件一定是放在互相衔接的连续串中的。一但对文件进行了多次增删和修改后，这种情形一般就不存在了。有时在某个文件中间还会出现释放的空串，这些情况都给恢复文件带来了极大的困难，甚至使恢复成为不可能。

## 三、文件恢复

文件恢复的操作是使用汇编语言调试程序DEBUG来进行的。这要使用其中的四条命令，

装入命令L、显示命令D、键入命令E和写入命令W，因此必须事先阅读DOS操作系统手册中的有关细节。

### 1、DOS在删除文件时干些什么

当一个文件被删除时，DOS首先将该文件目录项的第0个字节置为E5H（注意在DEBUG中用D命令显示目录项时，对应E5H的ASCII显示为小写字母e。但E5H并非e的ASCII代码，e的ASCII代码应为65H）。然后将文件占用的串在文件分配表FAT中对应的表项值全部置为000H，表示这些串已被释放，可以在以后的写操作中重新使用。除此之外，DOS不再进行任何其它操作，也就是说此时磁盘上被删除文件本身并没有遭到任何破坏。如此看来要恢复被删除文件是有可能的，但只有在一定的前提下才行。

### 2、文件恢复的条件

当文件的长度为一个串时，具体的说就是在单面软盘上文件长度小于512字节，双面软盘上小于1024字节，硬盘上小于4096字节时，不论删除的文件数目多少，所有文件都可完整的全部恢复。这是因为文件的起始串号在目录中未被破坏，第一个串总是找得到的。

如果文件长度超过一个串，但存放在相邻的连续扇区中时，也可完整地全部恢复。如果文件长度超过一个串，且又存放在不连续的扇区中，那么可能有以下几种情况：

（1）只删除了一个文件，且在删除该文件之前全部文件占用的串中间不存在空串，或空串的串号小于该文件的起始串号，此时该文件可以恢复。否则会因为分不清哪几个空串是原先该文件占用过的，使文件不能恢复或不能正确恢复。

（2）删除了一个以上的文件，此时若被删除的几个文件占用的串是互相穿插在一起的则无法恢复，这种情形就和存在空串是一样的。只有当被删除文件的数目及其占用的串与所有文件的总数目及其占用的串相比其百分比较小时，其中一部及全部或许有恢复的可能，主要取决于文件在磁盘上的相互位置。

（3）文件全部被删除。这种情形最糟糕。此时除了目录表中各文件的起始串外，文件分配表已成为一片空白，无法判断各文件的后续串号，所有文件均无法恢复。

不论何种情况，由于文件的起始串号保留在目录中，因而第一个串总是可以恢复的。

### 3、文件恢复的方法

文件恢复主要分两部分，恢复文件目录表和恢复文件分配表FAT。下面以双面9扇区软盘为例说明整个过程（其它形式的软盘及硬盘只是各命令中使用的有关参数，如驱动器号、相对扇区号不同而已）。

假定软盘上原先有4个文件，被删除的文件名为FILEHIND.COM，图1、图2分别为该盘的目录表和文件分配表FAT。将软盘插入A驱动器，当前驱动器为C。首先恢复被删除文件的目录项，步骤如下：

```
C>DEBUG <回车>
-I 100 0 5 7 <回车>
```

命令中第二个参数0表示A驱动器（对B、C驱动器分别为1、2），第三个参数5是目录起始相对扇区号，第四个参数7为目录扇区数目，均为十六进制数。此时目录扇区中的

内容已全部装入主机内存。

—D 〈回车〉

此时显示开关的4个目录项，如图1所示。屏幕左边是十六进制代码，右边是ASCII字符（注意当代码值大于7FH时的不对应关系，因为代码二进制值的最高位在显示时不起作用）。无参数的D命令每次显示四分之一扇区的内容，一次可显示4个目录项。找到需要恢复的文件目录项时，修改其第0字节值E5H。本例中待恢复文件名的第一个字符原先是F（ASCII代码46H），可用E命令恢复。

—E 100 〈回车〉

此时屏幕显示

—E 0B0A: 0100 E 5

键入46，回车

—W 100 0 5 7 〈回车〉

注意执行写命令W时，W后的四个参数必须和原先装入命令L中的完全一样，否则可能使软盘上的所有信息被彻底破坏。对于硬盘，这一点更要小心。必要时可将目录扇区和FAT扇区的数据备份到另一张盘上去，以防不测。到此目录项已恢复好了，接下去恢复文件分配表FAT中有关的表项值。对长度小于一个串的文件，这一步非常容易，只要在FAT中找到该文件的入口将表项值置为FFFH就行了。

如果文件长度大于一个串，且当文件是存放在相邻连续的扇区中时或虽不连续但符合2中的条件（1）时，FAT的恢复也是很容易的。首先根据文件的字节数求出文件所需的串数目。对双面9扇区软盘为文件字节数除1024，有余数时加1。然后根据目录项中文件的起始串在FAT中找到对应的入口，再向后检索值为000H的表项，求出该表项对应的串号，填入上一个表项，直到表项数目和文件应有的串数目相等时为止，在最后一个表项中填入FFFH。先在纸上演算好，再用E命令一一修改。本例中其步骤如下，假定文件FILEHINO.COM是符合2中条件（1）的。

先在图1目录表中得到文件的起始串号为002H，文件长度为000037C9H，求出文件所占的串数目为14D（D表示十进制），FAT中入口为3D。于是可在图2文件分配表FAT中以字节为单位从0开始数到3即为002H串第2表项所在。继续向后寻找发现第4表项、第7表项以及第8~第18表项值均为000H，根据先前假定，这几个串应该是文件FILEHIND.COM原先占用的串。于是可将第4表项对应的串号004H填入第2表项，第7表项对应的串填入第4表项，一直到最后第18表项的值填入FFFH为止，FAT已被恢复，整个文件也完全被恢复了。图3、图4分别为恢复后的目录表和文件分配表FAT。一个已被删除的文件是否符合2中所说的条件（1），只要数一下该文件在FAT起始入口以后的空串是否与该文件所需的串数目相等即可。一般说来，如果不相等则肯定不符合条件（1），相等则表示符合。但也可能有例外，比如说该文件上最末尾几个串原先是在所有文件已使用的串后面，而且在其中还可能夹有空串。如果在不久前进行过大量的删除文件操作，这种情况往往会发生。

当文件很大时，恢复FAT是一件相当繁琐的工作，稍不注意很容易将数的位数放错。因此，对一个连续存放的长度较大的文件，可采用如下的替换方法来恢复。

L 100 0 5 7

-D 100 1 7 F

```

0B0A: 0100 E5 49 4C 45 48 49 4E 44-43 4F 4D 20 00 00 00 00 eILEHINDCOM...
0B0A: 0110 00 00 00 00 00 00 20 70-E4 06 02 00 C9 37 00 00 ..... pd...I7..
0B0A: 0120 42 41 4C 4C 20 20 20 20-42 41 53 20 00 00 00 00 BALL BAS...
0B0A: 0130 00 00 00 00 00 00 71 09-21 00 03 00 03 00 00 00 .....a,!.....
0B0A: 0140 42 41 53 49 43 41 20 20-43 4F 4D 20 00 00 00 00 BASICA COM...
0B0A: 0150 00 00 00 00 00 00 60-54 07 13 00 00 66 00 00 .....<T...f...
0B0A: 0160 41 52 54 20 20 20 20 20-42 41 53 20 00 00 00 00 ART BAS...
0B0A: 0170 00 00 00 00 00 00 60-68 06 05 00 80 07 00 00 .....<h.....

```

图 1 文件删除后的目录表

L 100 0 3 2

-D 100 1 7 F

```

0B0A:0100 FD FF FF 00 F0 FF 00 60-00 FF 0F 00 00 00 00 00 } ...p.'.....
0B0A: 0110 00 00 00 00 00 00 00 00-00 00 00 00 40 01 15 60 .....@.<
0B0A: 0120 01 17 60 01 19 A0 01 1B-C0 01 1D E0 01 1F 00 02 .....@.<...
0B0A: 0130 21 20 02 23 40 02 25 60-02 27 80 02 29 A0 02 2B !#@.....).+
0B0A: 0140 C0 02 FF 0F 00 00 00 00-00 00 00 00 00 00 00 00 @.....
0B0A: 0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B0A: 0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B0A: 0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

```

图 2 文件删除后的FAT

L 100 0 5 7

-D 100 17F

```

0B0A: 0100 46 49 4C 45 48 49 4E 44-43 4F 4D 20 00 00 00 00 FILEHINDCOM...
0B0A: 0110 00 00 00 00 00 00 20 70-E4 06 02 00 C9 37 00 00 .....pd...I7..
0B0A: 0120 42 41 4C 4C 20 20 20 20-42 41 53 20 00 00 00 00 BALL BAS...
0B0A: 0130 00 00 00 00 00 00 71 09-21 00 03 00 03 00 00 00 .....a,!.....
0B0A: 0140 42 41 53 49 43 41 20 20-43 4F 4D 20 00 00 00 00 BASICA COM...
0B0A: 0150 00 00 00 00 00 00 60-54 07 13 00 00 66 00 00 .....<T...f..
0B0A: 0160 41 52 54 20 20 20 20 20-42 41 53 20 00 00 00 00 ART BAS...
0B0A: 0170 00 00 00 00 00 00 60-68 06 05 00 80 07 00 00 .....<h.....

```

图 3 文件恢复后的目录表

L 100 0 3 2

-D 100 17F

```

0B0A: 0100 FD FF FF 04 F0 FF 07 60-00 FF 8F 00 09 A0 00 0B } ...p.....
0B0A: 0110 C0 00 0D E0 00 0F 00 01-11 20 01 FF 4F 01 15 60 @.....D.<
0B0A: 0120 01 17 80 01 19 A0 01 1B-C0 01 1D E0 01 1F 00 02 .....@...

```

```

0B0A: 0130 21 20 02 23 40 02 25 60-02 27 80 02 29 A0 02 2B ! .# @... ) . +
0B0A: 0140 C0 02 FF 0F 00 00 00 00-00 00 00 00 00 00 00 00 @.....
0B0A: 0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0B0A: 0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0B0A: 0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....

```

图 4 文件恢复后的FAT

首先修改待恢复文件的目录项，记下起始串号，求出文件的起始扇区号，所需串长度。任意找一个串长度与待恢复文件相等或更大的文件并将其拷贝在一张空盘上。在DEBUG中将待恢复文件装入内存，然后写入已拷在空盘上的那个文件所在扇区。修改该文件目录项中长度字节与待恢复文件一样。如果该文件串长度大于待恢复文件，则还要修改FAT中相应的文件结束串。最后将该文件用RENAME命令重新命名为待恢复文件名，至此恢复即告完成，空盘上的文件已经是恢复后的文件了。这样做可不必逐个去恢复FAT中的每个表项，工作量可大大减少。

#### 4、子目录中文件的恢复

子目录中的文件其恢复方法与根目录中的文件基本一样。唯一不同之处是首先要找到子目录的目录表，因此必须对子目录表在磁盘上的存放方式作一充分了解。

当用MD命令建立一个子目录时，其过程与建立一个文件是类似的。首先在根目录中（如果是第二层子目录则在相应的父目录中）建立该子目录的名字，其格式与文件名一样。然后将FAT中检索到的第一个空串作为该子目录的目录扇区，起始串号保留在目录项中。同时起始串的表项值立即被置为FFFH，也就是说DOS一开始只为子目录扇区提供一个串的空间，在双面软盘上为2个扇区，可放32个目录项。在第一个串中DOS自己还要用去二个，剩下30个可供用户使用。当用户在该子目录中的文件数超过30时，DOS再次寻找一个空串作为该子目录下一个扇区，并将此串串号放入起始串表项中。这样的扩充可一直进行下去，直到磁盘空间被用完为止。所以说，子目录中的文件数目是有限制的。从第2个串开始，每串中的32个目录项用户都可使用。

因此，当恢复子目录中被删除的文件时，必须根据该文件在子目录中目录排到的次序找出相应的目录扇区才行。对前30个文件，可在根目录中找到起始串号求出扇区号，对后面的文件则要在FAT中去寻找相应的串号。

### 四、文件属性的修改

如果用户手中有IBM—PC DOS补充盘上的FILEHIND.COM文件，用它来修改文件属性是相当方便的。但该文件进行的修改只能局限于软盘，并且不能修改子目录名本身的属性，有时对子目录中的文件也无法修改。如果希望修改硬盘上文件属性或希望将其一子目录名隐去，还得借助于DEBUG。本文开头磁盘文件管理一节已对目录项中有关文件属性的部分作了介绍，因此修改文件属性是一件很容易的事。文件属性可以相加使用，例如希望将一个原先属性为20H（档案位关闭）的文件同时具有隐藏、只读属性，可将目录项中第11字节值修

改为20H+01H+02H=23H即可。应注意08H(卷标)与其它属性相加是无意义的,10H(子目录)与20H(档案位)相加则引起混乱。除此之外可任意相加,但对用户文件一般不要使用04H(系统文件)属性。当一个文件置只读属性后,拒绝COPY、ERASE命令,置隐藏属性后,拒绝DIR、COPY、ERASE命令。DOS 2.10以下版本不能执行隐藏的用户文件,但可进入隐藏的子目录。进入dBASE III后所有的隐藏文件均可执行,并可用其中的删除命令删除。用户可根据需要修改一个文件的属性以起到某种保护作用,例如对dBASE III中的内存变量存储文件可置成只读,从而可禁止修改、再存、删除等操作,以保证其中数据不被破坏。

文献出处:《计算机世界》月刊1987年1期12~17页

## UNIX 系统中误删文件的恢复

宋运祁

如果不慎误删了UNIX系统中的文件,只要再没有人在系统中建立新的文件,是可以恢复的。这是因为用rm命令删除文件时,虽然已将该文件的i节点(inode)中的信息基本清除,但该文件所占用的各个磁盘块(block)上的文件内容并没有被破坏。只要能把这些盘块号找到,就可以重建其i节点,把被删的文件恢复出来。

当文件被删除时,该文件的i节点中的文件所占用的盘块号被依次补充到文件系统的空块表(free block list)的顶部,它位于文件系统专用块(Super block)即2号块中。恢复被删文件的关键在于从空块表顶部找到这些块号,并将其填入任一个空i节点中,重建此i节点。下面以DUAL 83/80 68000 UNIX 7版系统为例(假定被删文件是在/e目录之下)说明恢复被删文件的具体步骤:

1. 首先打Sync命令使内存中的专用块(Super block)副本刷新磁盘上的专用块。
2. 用od命令查看被删文件所在的文件系统设备(本例为/dev/smdoe)的专用块(磁盘地址0×200起)。

```
% od -x /dev/smdoe 200 : more
200 03bc 0000 5c58 0012 0000 1b5e 0000 1b5d
210 0000 1b5c 0000 1b5b 0000 1b5a 0000 1b59
220 0000 1b58 0000 1b57 0000 1b56 0000 1b55
230 0000 1b54 0000 1b53 0000 1752 0000 1551
240 0000 15a4 0000 176c 0000 1932 0000 1d05
  :   :   :   :   :   :   :   :   :
  :   :   :   :   :   :   :   :   :
2f0 ... .. ... .. ... .. ... ..
```

其中从200地址起的十六进制数12(即十进制数18表示以下的从00001b5e起(每4个字



节为一个块号)共有18个空块号。第十八个空块号00001d05就是被删文件的第一个盘块号。如果该文件不止一个盘块,则其前面的0000 1932就是该文件占用的第二个盘块号。依此类推。要估计被删文件有多大,占多少盘块,不妨多估计些。多了可删掉多余的,少了就再找不回来了。假如估计最多不超过6块(3072字节即 $0 \times 600$ ),则由后到前依次记下6个盘块号如下:

```
0000 1d05 0000 1932 0000 176c 0000 15a4 0000 1551 0000 1752
```

[注]如果估计的盘块数等于或大于专用块中的空块数(例如本例中估计有20个盘块),则读取此空块表中的17个空块号,第17个块号为0000 1b5d,另一个0000 165e不能直接读取,其余的三个块号要到0000165e号块中去找。为此将165e十六进制数乘以2得到2cbc再在其后加两个零化成此号盘块在磁盘上的地址2cbc00,然后用od命令od-x/dev/smdoe 2cbc00读此块内容。头两字节如0032表示以下有50个块号,第50个块号就作为该文件的第十八块,第49个作为十九块,依此类推。

以上每个块号占4字节。将每个块号的最高字节00删去,使每三个字节表示一个空块号如下(因为在i节点中,每个块号只占3字节):

```
001 d 05 001932 00176c 0015a4 001551 001752
```

为了便于用adb命令,将这些块号填入i节点中,将以上块号改为2字节一段,并将前面的00略去,得到:

```
1d 500 1932 17 6c00 15a4 15 5100 1752*
```

```
001d 0500 1932 0017 6c00 15a4 0015 5100 1752
```

3. 从专用块(Super block)的空i节点(free inode list)中任取一个空i节点号。其中2d0地址的十六进制数0020表示以下共有32个空i节点号(每个占2字节)。可以取第一个即009e号空i节点。为了算出此号i节点在磁盘上的首地址,可先将它化为二进制数1001 11 10然后与二进制数1111相加得到1010 1101,再在其后加6个0得到1010 1101 000 000,化为十六进制数2b40。

4. 以超级用户(super user)用adb命令将以上找到的被删文件的盘块号写入到磁盘地址 $0 \times 2b40$ 起的 $0 \times 9e$ 号空i节点中,并重建该i节点的其它有关信息。

```
# adb -w /dev/smdoe
```

```
a. out file=/dev/smdoe-warning: not in a. out format
cannot open core
```

```
ready
```

```
2b40? w81a4 1000 c00 1d 500 1932
```

```
17 6c00 15a4 15 5100 1752
```

打CTRL-D退出adb命令

说明: 81a4为i节点的mode字表示是普通文件,读写权为rw-r--r--。1为链接数,第一个0表示文件主号为root;第二个0表示文件主组号即与root同组,第三个0与c00表示此文件字节数为3072(因为有6个盘块 $\times 512$ 字节)接着是此文件的盘块号(每个块号占3字节)。

5. 用fsck检查此文件系统(/dev/smdoe)修正由于以上人工修补造成的不一致。遇到问号,一律打Y。

```

# fsck/dev/smdoe
• • phase 1— Check Blocks and sizes
• • phase 2— Check Pathnames
• • Phase 3— Check Connectivity
• • Phase 4— Check Reference Counts
UNREF FILE I=158 MTIME=Jan
      10:0 1970
RECONNECT ? Y      [打Y]
FREE INODE COUNT WRONG IN
      SUPERBLK
FIX ? Y      [打Y]
• • Phase 5— Check Free List
EXCESSING DUP BLKS IN FREE LIST
CONTINUE ? Y      [打Y]
6 DUP BLKS IN FREE LIST
BAD FREE LIST
SALVAGE ? Y      [打Y]
• • Phase 6— Salvage Free List
402 files 2000 blocks 20845 free
• • • FILE SYSTEM WAS MODIFIED
• • •

```

这样就已将误删的文件恢复了。文件名定为000158（以i节点号为名）并放在/e/lost+found目录之下。只要用mv命令将它改向适当的名字转到适当的目录之下，并检查文件内容删除后面多余的原非本文件的内容即成。

必须注意，此文件系统的根目录下一定要有lost+found目录。如果没有的话，以上用fsck命令就无法将文件抢救回来，还要采用更麻烦的人工改变某一目录的方法。顺便指出，为了防止文件丢失，每个文件系统的根目录之下必须用mklost+found命令建立lost+found目录。以83/80为例。在/usr/e/f/g以及/目录之下都应当有lost+found目录。

文献出处：《计算机世界》月刊1987年8期23—24页

## 文件的属性及其修改应用

丹东汽车制造厂 牟耀东

在计算机实际应用当中，往往有大量的数据及应用程序需要加以保护和保密，本文介绍一种编制容易，但对文件保密性及保护性均能起作用的实用程序，它适用于广大的计算机应用人员，以及初学者。

## 一、文件属性

在IBM—PC DOS2.00操作系统支持下，文件有多种属性，通过属性的划分，可对文件进行对应管理及控制使用。文件的单独属性可分为：

01H：标志文件为只读，当使用功能调用3DH企图打开该文件供输出时，会导致返回一个错误代码，此属性可以与以下属性结合使用。

02H：隐含文件，该属性文件拒绝通常的目录检索。

04H：系统文件，该属性文件拒绝通常的目录检索。

08H：该项的前11个字符包含卷标号，不含其它有用信息，并只存在于根目录中。

10H：该项定义一个子目录，并拒绝通常的目录检索。

20H：归档位，每当文件已经写完并关闭时就置此位，此位可和其它属性一起使用。

由属性的特点可见，只要根据文件应用的范围及要求，进行相应属性的修改即可达到保护、保密文件的目的。由于文件属性的修改较繁琐，故采用程序来完成。

## 二、程序简介

1. 要求硬件环境为IBM—PC及其兼容机，操作系统为CC—DOS2.00或以上版本，另需动态调试程序DEBUG.COM。

2. 程序主要利用了DOS的功能调用，通过设置相关功能号由软件中断INT21来实现，因而占用内存少（小于256字节），灵活性高，实用性强。

程序起始地址为DS:0100H，执行时需要将功能号置于AH并根据所进行的调用要求执行一些必要的寄存器操作，软中断结束后送回相关信息，控制权仍交给属性修改程序。其整个属性修改过程共由7次软中断实现（修改失败则8次）。

需要指出的是，调用功能号0AH进行软中断完成键盘输入包括驱动器、路径及文件名的字符串，其最后是以回车符（0DH）来结束的。为此需将0DH改为00H才能使输入的信息有效，因而程序设置了判断字符串长度，找出回车符并以00H代换的功能。

由于本程序在汉字CC—DOS支持下编制而成，所有提示均采用汉字显示。

### 3. 存贮区分配

DS:0100~DS:0101 程序区。

DS:0102~DS:0144 数据及键盘输入缓冲区。

DS:0145~DS:0195 程序区。

DS:0196~DS:01FF 数据缓冲区。

### 4. 所使用的软件功能调用

01H：完成键盘输入一个字符，屏幕回显及将此字符置于AL。

09H：屏幕显示字符串，DS:DX指向字符串缓冲区首址。

0AH：键盘输入字符串，DS:DX指向输入数据缓冲区首址，其第一字节为此缓冲区容纳字符个数，第二字节为接收的字符个数。本程序设定缓冲区长度为13H。

43H：置或取文件属性。由于程序设定为改变文件属性，因而置AL内容为01H。DS:DX

指向包含驱动器、路径及文件名的ASCII字符串首址。

### 三、程序编制

1. 启动DEBUG.COM, 并于DS:0100H地址使用汇编命令“A”打入如清单(附1)所示程序。具体操作为:

-A0100↓则表示由DS:0100地址始输入汇编语言程序。

2. 程序中提示符(中文)的输入, 利用伪操作来实现较优。如:

DS:01A0DB“××××……\$”表示从地址01A0H始输入汉字如引号内所示, 结尾必须用ASCII字符“\$”(24H)结束。

3. 键盘输入缓冲区初始时要按规定设置数据, 存放键入字符区内必须置00H, 可利用输入命令“E”来进行, 如:

-E0132↓即表示从当前地址DS:0132H开始输入实际的十六进制数值。

4. 程序入毕, 用命名命令“N”命名为“ANYFILE1.COM”, 具体操作如:

-NANYFILE1.COM↓即可。

用寄存器修改命令“R”将BX置为0000H, CX置0100H, 如:

=RBX↓

BX:0100

:0000↓ 此项为输入值

同理可完成对CX的输入。最后操作如:

-W↓ 即完成全部编制与存贮操作, 此时的“ANYFILE1.COM”文件已写入当前磁盘中。

### 四、有关说明

此程序是在CC-DOS2.00支持下编制的, 同样也可在MS-DOS2.00支持下编制与执行。

2. 文件属性修改可以是单独的某个属性, 也可改为数个属性的复合, 使其保密性, 安全性更高。

### 五、程序使用

1. 作为长期保存文件使用, 防止意外情况删除、修改等操作, 可将文件属性置为“只读”。启动此程序, 输入包括驱动器、路径及文件名的字符串后, 选择属性“1”输入, 若修改成功, 屏幕显示“继续?(Y/N)”, 按非“Y”任意键可退出。

2. 作为保密使用

A: 作为文件保密使用, 可采取将文件属性置为“隐含”、“系统”等方式, 也可组合使用, 使一个文件兼具多种属性。如在选择属性输入时, 输入数字3则被修改文件兼具“只读”、“隐含”属性。经过保密的文件, 不能采用通常方式列出文件名及其内容。此方法只

适用于保密状态保存文件，常规应用时需将属性改回。

B：文件既可保密，又可使用的二种方式：

①为克服根目录中隐含文件需经属性修改才能使用的问题，采用建子目录的方法将有关文件归属其内，之后对子目录属性进行保密处理，使其在根目录的通常检索下不被列出。其处理结果见附2。由于子目录“GY”经过隐性处理，因而根目录检索不被列出，但指出路径名后可列出在此子目录下存在的5个文件。

由于对根目录中的子目录采用了加密处理，采用“TREE”命令检查其所在驱动器上的全部目录通道也不能发现子目录“GY”。

应用时，只要将子目录做为当前目录即可对其中文件加以使用。

②采用如①所述方式处理的子目录，通常方式无法检索，但如果已知子目录名采用通常方式还是能列出文件名及其内容的。为进一步加强保密性，对子目录下的文件继续进行隐性处理，则可进一步加强保密性，起到二级及多级保密作用（即建立多级子目录，进行多级加密）。经过此种方式加密的文件，在包含命令文件的子目录为当前目录条件下，也是可以执行的。以执行DBASE III命令文件为例：

1. 所有的程序文件（即文件扩展名为“PRG”）均在子目录“GY”下。并将此磁盘作为B驱动器磁盘。

2. DBASE系统盘为A盘，其驱动器，路径的选择可于批处理文件“AUTOEXEC.BAT”中进行，由批处理自动完成，批处理程序清单见附3。

3. 进入DBASE系统后其参数的选择由CONFIG.DB完成，程序如下：

```
A>TYPE CONFIG.DB
```

```
PATH=A;
```

```
DEFAULT=B;
```

```
COMMAND=DO MAINM
```

经实际应用验证，采用二级或多级保密处理有如下主要优点：

a：程序在子目录中可执行（不包括扩展名为“.COM”及“.EXE”等DOS命令文件），数据文件在子目录中可存取（数据文件不能置为“只读”属性）。

b：由于采用了二级或多级保密，其保密程度较高，破译较难。

c：采用通常方式的拷贝（盘拷贝除外）、删除命令均无效。

## 六、结 语

以上程序编制与应用已经过半年以上时间验证，效果好，执行容易。目前已在IBM-PC/XT及其兼容机DS-PC机上应用。

### 附 1

```
4F7F:0130 00 00 13 00 00 00 00 00 00—00 00 00 00 00 00 00 00
```

```
4F7F:0141 00 00 00 00 00
```

```
4F7F:0100 EB43 JMP 0145
```

```
4F7F:0102 DB, 修改文件属性, 入文件名 $:
```

```
4F7F:0145 B409 MOV AH,09
```

4F7F : 0147	BA0201	MOV	DX,0102
4F7F : 014A	CD21	INT	21
4F7F : 014C	B40A	MOV	AH,0A
4F7F : 014E	BA3201	MOV	DX,0132
4F7F : 0151	CD21	INT	21
4F7F : 0153	BB3301	MOV	BX,0133
4F7F : 0156	021E3301	ADD	BL, ( 0133 )
4F7F : 015A	43	INC	BX
4F7F : 015B	B80000	MOV	AX,0000
4F7F : 015E	8907	MOV	( BX ), AX
4F7F : 0160	B409	MOV	AH,09
4F7F : 0162	BAA001	MOV	DX,01A0
4F7F : 0165	CD21	INT	21
4F7F : 0167	B401	MOV	AH,01
4F7F : 0169	CD21	INT	21
4F7F : 016B	B400	MOV	AH,00
4F7F : 016D	2C30	SUB	AL,30
4F7F : 016F	89C1	MOV	CX,AX
4F7F : 0171	B80143	MOV	AX,4301
4F7F : 0174	BA3401	MOV	DX,0134
4F7F : 0177	CD21	INT	21
4F7F : 0179	2DFFFF	SUB	AX,FFFF
4F7F : 017C	7407	JZ	0185
4F7F : 017E	B409	MOV	AH,09
4F7F : 0180	BAE001	MOV	DX,01E0
4F7F : 0183	CD21	INT	21
4F7F : 0185	B409	MOV	AH,09
4F7F : 0187	BAEA01	MOV	DX,01EA
4F7F : 018A	CD21	INT	21
4F7F : 018C	B401	MOV	AH,01
4F7F : 018E	C021	INT	21
4F7F : 0190	2C59	SUB	AL,59
4F7F : 0192	74B1	JZ	0145
4F7F : 0194	CD20	INT	20
4F7F : 01A0	DB' 输入文件属性: ( INPUT FILE ATTRIBUTE ) 0 ;		
	:读写,1:只读,2:隐含,4:系统: \$'		
4F7F : 01E0	DB' 修改失败! \$'		
4F7F : 01EA	DB' 继续? ( Y/N ) \$'		

附 2

```
B>DIR/W $GY
VOLUME IN DRIVE B IS
MYDDATEFILE
DIRECTORY OF B: $GY
    . . . AUTOEXEC BAT
MBILLT PRG  OVERBILL PRG 5 FILE(S)  135168BYTES FREE
B>CD $
B>DIR/W
    VOLUME IN DRIVE B IS MYDDATEFILE DIRECTORY OF B: $
CONFIG  DB  MBILLT PRG
ANYFILE1.COM 3FILE(S)  135168 BYTES FREE
```

附 3

```
A) TYPE AUTOEXEC, BAT
ECHO OFF
CLS
HDINIT
CLS
ECHO PLEASE WAIT....
FILE1
CCCC
ECHO DBASE III 系统盘放入A驱动器, B驱动器为命令文件
BAUSE
B:
CD $GY
A:
DATE
DBASE
ECHO ON
```

文献出处:《电脑与微电子技术》1987年2期28—31页

# 数据库系统中恢复管理的设计与实现

徐 杰 张 滨

摘要本文讨论了数据库设计中恢复管理的主要问题，描述了一般的恢复管理原则，吸取了SQL/DS恢复管理的优点，提出了两种类型的恢复——处理失败和用户逻辑错误恢复的处理方法，提出了恢复系统（Recovery System）的设计思想，同时，在安全数据库管理系统（SDBMS）中实现了该恢复系统（RS）。

## 一、引言

对于数据库中的数据要求保证其正确性和一致性。但是，由于数据库的环境所致，数据库常常会遭受破坏，它可能来自计算机系统的硬件和软件故障，如存贮设备故障和操作系统本身异常终止或数据库管理系统（DBMS）本身异常终止；同时数据库也会受到大自然和人为的破坏，如雷电引起的瞬时停电或操作员及用户的操作失误以及人为故意对数据库数据的修改。这样都可能使数据库的数据处于不正确和不一致状态。在数据库的数据遭到破坏时，如何迅速、正确地把数据库的数据恢复到可用状态——即数据库的恢复管理，是极其重要的，它是评价数据库系统性能优劣的一个重要准则。

由于数据库中的数据不能保证百分之百的正确性，总存在被破坏的可能性，因此，数据库系统就一定要保证：一旦数据出错，要立即修正错误的数据库，确保其正确、可靠。这样，恢复系统在安全数据库系统中就非常重要。安全数据库管理系统（SecureDBMS——SDBMS）是保证数据安全的核心，恢复系统是SDBMS保证数据库数据安全的一部分。

## 二、恢复的一般原则

这里所描述的恢复原则，是一个好的恢复系统必备的条件，也是本恢复系统RS追求的设计目标。

一般原则应是：

1. 系统失败时损失应最小，重新启动系统时用户无需要重新登记事务或数据，也不必从头开始执行原来的事务。
2. 撤消（Undo）或重做（Redo）的应尽可能是单个事务，不应造成在恢复时总是重新存储整个数据库或整个文件。
3. 恢复速度要快。因为恢复期间数据库中总会有部分数据用户不用，故可利用周期性地整理恢复数据的方法缩短恢复的时间。



4. 应尽量减少恢复时的人工干预工作，亦即恢复工作应尽量自动化，不要由用户保存恢复数据和决定恢复过程，这样既费时又容易出错。

5. 确保恢复数据的安全可靠。

因此，本恢复系统RS设计的目标就是一个高效、快速、安全可靠的恢复管理系统。

### 三、恢复类型

虽然许多失败是不可预知的，但是通过对计算机系统的分析，总结失败的原因，本文只从主要的实现方面进行考虑：

·处理失败

由于各种原因，系统和应用在功能上失败，包括：

i) 应用失败

ii) 系统失败

iii) 介质故障

·用户逻辑错误

在系统或应用执行已经定义的所请求的功能时，存在有许多错误，而且请求本身也可能不正确，因此，用户或应用程序不能保证其说明功能的完全正确性。

本文给出的恢复系统主要针对这两类问题进行恢复管理。处理失败恢复和用户逻辑错误恢复有本质的区别。因为本系统中处理失败是被立即检测出来的，而且修正动作（恢复处理）是检测到失败后立即执行的；而用户逻辑恢复存在有许多问题，因为它不能立即被检测出来而只能由用户自己发现，处理方法是：如果在变换（即对数据库进行的变换）提交之前检测到错误，恢复处理简单地退出变换即可；若在变换之后才检测出用户逻辑错误，则必须通过建立后备数据副本来处理这种情况。因此，应当尽量避免用户逻辑错误，而避免的方法是通过合理的应用设计和合理的数据控制。

### 四、数据库恢复的支撑环境和恢复处理过程

#### （一）首先引进几个概念

·逻辑工作单元：类似于事务（Transaction）的概念，它是由一组用以完成一个逻辑应用功能的操作组成。逻辑工作单元既可是单个也可是一组命令，对DB进行操纵是以逻辑工作单元为原子构成的。

·检验点（Checkpoint）：检验点实际是一个时间参照点，它表明该点这一时刻所有完成的DB修改都提交到DB中去了。检验点包括事务检验点和系统检验点两种，事务检验点即系统允许用户在其事务的适当位置上设置检验点，以便被撤消再运行时不必从头开始，如图4.1所示。系统检验点即系统本身按一定时间间隔设置检验点，使系统停止后能够从最近的一个系统检验点重新启动。

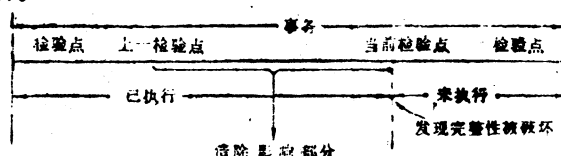


图4.1 检验点与事务

·恢复日志：记载了对DB修改的全部情况，设立恢复日志是为了DB的恢复而保留详细的数据，因此，对每一操作都要记录被操作数据的前象（Before image即操作前的状态）和后象（After image即操作后的状态）。

·数据库数据集的恢复：当数据库集遭到破坏时，必须将其恢复到某一正确状态。在这种情况下，数据库的恢复通常是根据最新的数据库后备副本和运行日志来实现的。

## (二) 支撑环境

恢复系统RS提供如下实用程序，它们是进行DB恢复的必备条件，它们是：

- 数据库回退 (Backout)

回退是指应用异常终止时撤消 (Undo) 该应用对数据库所做的修改，将DB从一种不一致状态恢复到该应用执行前的一致状态的过程。RS设计的数据库回退是动态的。

- 数据库数据集映象副本 (Database data set image copy)

它提供给恢复系统生成数据库数据集转贮映象的功能。

- 数据库变换累聚 (Database Change accumulate)

- 数据库数据集恢复 (Database data set recovery)

- 日志打印实用程序 (Log print utility)

它提供了打印恢复日志内容的功能。

## (三) 恢复处理过程

为了实现本恢复系统的目标，根据上述提供的恢复管理支撑手段，RS的恢复处理过程如图4.2所示，它也是恢复系统处理各类失败的基本思想，其处理方法简述如下：

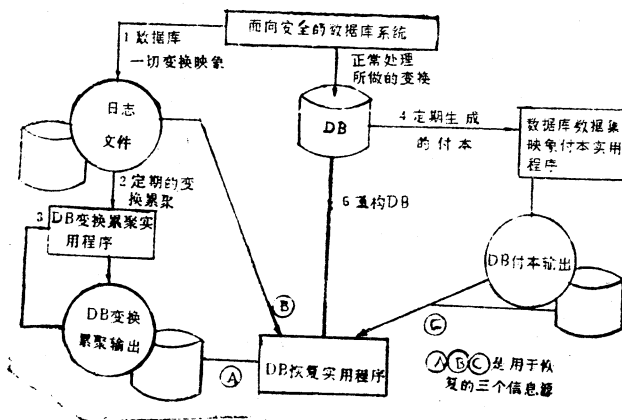


图4.2 恢复处理示意图

1. 把变换的数据状态记载到日志中。
2. 选择出日志文件中的变换的数据库日志记录，并且以数据库和文件次序分类日志记录，选择和分类是数据库变换累聚实用程序运行时生成变换累聚文件的一部分工作。
3. 归并已分类和已选择的先前累积变换，仅仅保持最新的数据，归并是数据库变换累聚实用程序运行时生成变换累聚文件的一部分工作。
4. 使用数据库数据集映象副本实用程序临时地转贮数据集以提供一个后备副本。
5. 数据库数据集恢复实用程序读②，③，④信息，恢复数据到可用状态。

## 五、两类恢复问题的处理机理

前面已描述了恢复的类型有：处理失败的错误和用户逻辑错误恢复两大类，而处理失败的恢复可分三种情形：

1. 应用失败的恢复
2. 系统失败的恢复
3. 介质失败的恢复

下面就各个问题描述其恢复处理机理。

### (一) 处理失败的恢复

#### 1. 应用失败的恢复

当一个修改数据库的应用发生异常终止时，数据库中的数据一般处于不一致状态，此时，必须将该应用所做的全部DB修改撤消，为此，必须在数据库系统中使用运行日志功能去登记应用对DB的修改。

如果该应用是一个批处理程序，那么在该程序异常结束后应立即执行数据库恢复实用程序。如果该应用是联机处理环境下的一个事务或逻辑工作单元，则由动态事务恢复功能完成数据库的恢复。因此，本系统对应用失败的恢复是动态实现的。

#### 2. 系统失败的恢复

当操作系统，数据通信系统或数据库管理系统等异常终止时，正在其上运行的数据库应用将随之而异常终止。如果已正常结束的数据库应用所作的数据库修改还没有写回数据库，则会因为I/O缓冲区的内容的丢失而无法写回数据库中去；而那些尚未正常终止的数据库应用所做的数据库修改部分，若在系统故障前已将它们写回数据库中，则需将这些部分予以撤消。因此，在系统故障后需要对数据库进行恢复处理，对这种故障的恢复主要依靠日志文件。

在RS中，规定修改过的数据写入数据库的时间一般在检验点或宿主机停机时刻，因此在系统失败后，不但要撤消（Undo）由此而异常结束的逻辑工作单元所做的那些先写入DB的数据修改部分，而且还要重新（redo）将已完成的逻辑工作单元所做的尚未写入数据库的数据修改部分写入。也可利用重启动功能在系统重启动时使用运行日志自动实现对数据库的恢复（正常停机时重启动也完成此恢复功能）。

#### 3. 介质故障的恢复

在数据库存贮介质发生故障后，除了要排除硬件故障外，还要将其上面的数据库数据集恢复到可用状态——正确的和一致状态。因此，在数据库系统的日常运行中应定期地对数据库进行转贮（dump），所谓转贮就是定期地将整个数据库中的数据复制到磁盘或磁带上保存起来的过程。RS对数据库存贮介质故障是以数据集为单位进行恢复的。在排除硬件故障之后，数据库数据集恢复实用程序以最新的一次数据库数据映象后备输出文件、数据库交换累聚文件和恢复日志的信息为依据，将已损坏的数据集文件恢复到故障前的状态。

同时,考虑一种特殊情况,即数据库恢复日志文件被破坏。这时恢复处理比较复杂,为此采用双重运行日志功能,这样如果一个运行日志文件发生介质故障,RS能继续使用另一个运行日志副本;若两个副本的介质皆出现故障,只要能从这两个副本文件中拼凑出一个完整的运行日志副本,RS可继续处理介质故障的恢复。不过设置双重运行日志进行介质故障恢复,其控制十分复杂且增加系统开销。

## (二) 用户逻辑错误的恢复

这类错误的恢复依赖于发现这种错误的时间,如果这种错误是在应用执行期间,(即对DB的修改提交之前)被检测到的话,恢复处理就变得较简单了,此时只要强制该应用异常终止,然后对数据库进行恢复处理即可;如果这种错误是在修改被提交之后才发现的话,对数据库的恢复处理就相当复杂了,通常可以人工地撤销错误的数据修改,也可以利用数据库的后备副本将其恢复到某一时刻的状态,然后重新进入这个时刻之后对数据库所做的所有有效修改。一个比较简单的实现办法是用静态转贮的方法用数据库档案文件或后备副本对整个数据库进行恢复,并且重做其后有效的修改。

## 六、恢复系统RS的实现

恢复系统RS是安全数据库管理系统(SDBMS)的一部分,从图6.1中可看出两者的关系。

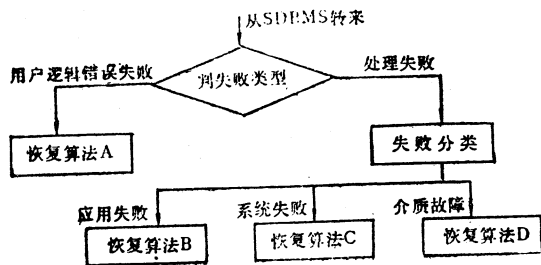


图6.1 SDBMS总图

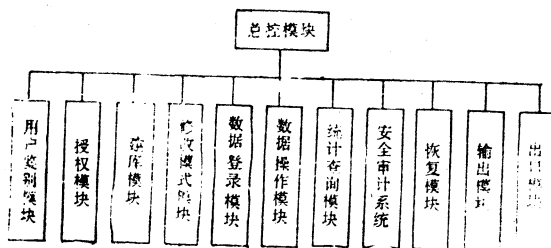


图6.2 RS总框

### (一) SDBMS简介

SDBMS是一个面向统计数据库的安全数据库管理系统,它保证统计数据库的统计查询(如Sum,Count,Average等)不泄露个体信息。它给用户提屏交互的,自含式会话语言,使用户接口简单灵活。整个DB对用户是透明的。本系统不仅安全性程度高,而且还给用户丰富的统计查询信息和数据操作能力。总的思想是层层设防,最后审计,如果有“脏数据”进入系统,DBA则要恢复系统执行恢复处理

采用的安全措施有七种:①用户鉴别;②授权;③访问控制;④在概念级使用安全性约束和完整性约束;⑤统计查询的推理控制;⑥安全审计;⑦恢复。

本SDBMS是在IBM PC/XT机上开发的,使用的是PASCAL语言。

### (二) RS的实现

图6.2给出了RS的总控框图,本RS的设计与实现严格按模块结构进行,使系统层次分明,正确程度高。RS按不同的失败类型分别采取不同的处理算法。

· 恢复算法A: 处理用户逻辑错误;

- 恢复算法B：处理应用失败；
- 恢复算法C：处理系统失败；
- 恢复算法D：处理介质故障；

下面分别描述这些算法，其机理在第五部分已描述，这里不再赘述。

(1) 恢复算法A

算法A是在用户产生逻辑错误之后，对数据进行恢复处理，框图如6.3图示。

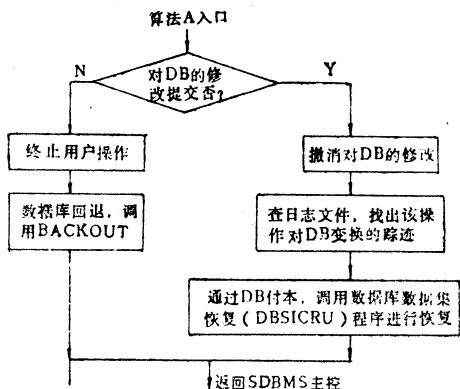


图6.3 恢复算法A框图

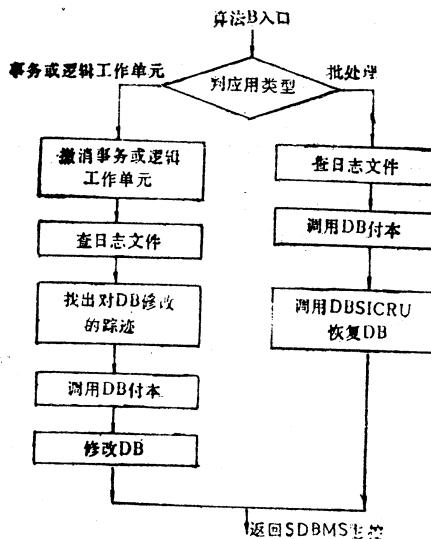


图6.4 恢复算法B框图

(2) 恢复算法B、C、D

算法B是应用失败的恢复，它在两种情况下工作：一是联机情况下的事务或逻辑工作单元的应用，二是批处理的应用，算法框图如6.4所示。

算法C是处理系统失败的情况，框图如6.5所示。算法D是处理介质故障，框图如图6.6。

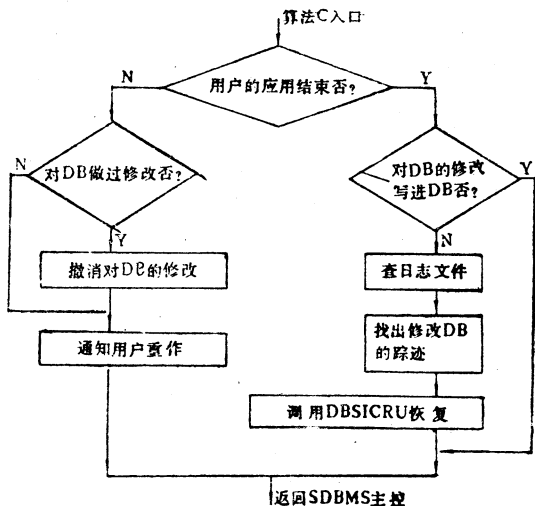


图6.5 恢复算法C框图

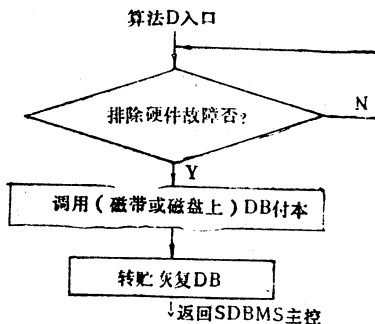


图6.6 恢复算法D框图

## 七、结束语

本文对恢复系统RS的设计与实现作了一些探讨,该系统吸取了SQL/DS恢复管理功能的优点,具有快速、有效地恢复数据库的功能,能够满足面向安全数据库设计的要求。不过在具体实现时,恢复系统功能的强弱、效率高低以及开销大小是应权衡评价的指标,研制出一个实用的恢复系统是最终宗旨。本恢复系统RS设计也存在一些问题,如没有考虑数据库重组问题,因此,也有待今后的进一步加强和完善。

### 主要参考文献略

文献出处:《新浪潮》1988年1期20—24页

## 一种文件加、解密方法的实现

李 革

**摘要**本文介绍一种通过软件来实现IBM-PC及其兼容机磁盘文件加、解密的方法。这种方法十分简便,但其保密性能相当好。读者如有兴趣不妨一试。

### 一、概 述

在IBM-PC及其兼容机上用过WORDSTAR的人都会有这种体会,即产生的文本(可以是信件、公文等)对任何人均是公开的。若不愿公开其文本资料,则只有尽量保管好存有文件资料的磁盘。如果有一种软件能十分简单地给文本加、解密,那将会是令人高兴的。本人利用自己过去常在BASIC中使用的一种加、解密原理开发了一个应用程序。通过实践,证明是可行的。

本文将讨论这种方法的实现。笔者认为,此法也适应于其它类型的磁盘文件。

### 二、设计原理

我们知道,不论是ASCII码还是BINARY类型的磁盘文件调入机器内存以后,总会占用一定的内存空间。如果我们通过某种约定获得一串不重复而且是无规律的数,使得这个内存区域内每一存贮单元中的数值发生变化,那么我们就无法读懂这个磁盘文件的内容,从而达到加密的目的。如果也能利用这串同样的数据文件的内容得到恢复(即解密),那么通过这串数就完成了加、解密工作。注意,用于加、解密的这串数其变化范围必须相当大,并且

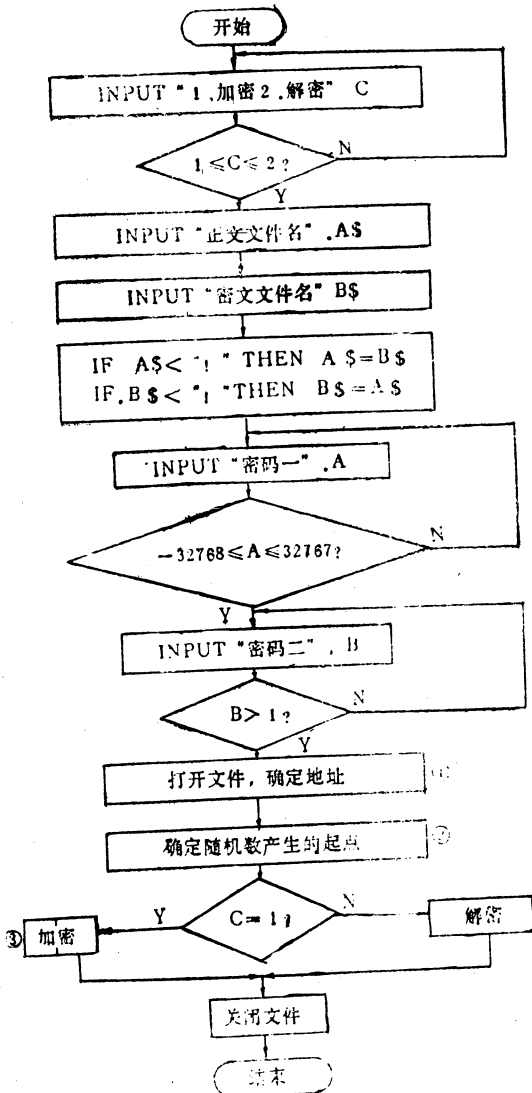
通过约定产生的数串种类必须繁多，这样别人才难以破译。

在IBM—PC及其兼容机中，随机函数的命令RND ( X ) 就能够满足上述约定。这个命令可以建立一串随机数值，并且能够从任何一个预置点开始建立起。

RND函数能产生一串准随机的数值，也就是说，虽然它们看起来是随机的，但是数值是按一种严格的方法产生出来的。所以谁掌握了它的种子数值，谁就可以重新产生出一串相同的“随机”数。

当加密程序产生出一个随机数值后，它就到普通文件缓冲区中，找出未加密文件的第一个字节，然后将这个字节的内容与随机数相加，并取相加结果以256为模的余数，所以这样得到的数值不可能大于255，把这个最后的数值存入与普通文件缓冲区相对应的密码缓冲区中，解密则正好是逆过程，程序从装有密码文件信息的缓冲区中得到信息，然后按一一对应的方式减去上一次加入的随机数，再存入普通文件的缓冲区中。

### 三、实现方法



实现的方法有多种，为了更好更简明地说明上述原理，我们用IBM—PC编译 BASIC来讨论一种较易实现的方案。当然，这并不一定是最优的，例如：如用IBM—PC宏汇编来实现的话，速度将更快。

程序框图及分析如下：

其中①②③④将列出源程序进行分析，其余部分已十分简单明了，读者可自行分析。

①打开文件并得到文件缓冲区中存取文件信息资料的第一个内存单元的地址。

```
OPEN "R", # 1, A$, 128
```

```
OPEN "R", # 2, B$, 128
```

```
AD 1 = VARPTR (# 1) + 188
```

```
AD 2 = VARPTR (# 2) + 188
```

A \$、B \$分别为正文、密文件名，文件名可以相同。

②确定随机数产生的起点

```
RANDOMIZE A
```

```
FOR X = 1 TO B
```

```
    A = RND
```

```
NEXT
```

其中：A和B为密码。A为-32768至32767中的一个整数。B为大于1的一个数。注意：这里B值按理论是可以取得很大的，但越大耗费的时间也就越多，建议你一般取到1500左右

的整数也就可以了，因为这样已可以近十亿种不同的随机数序列中挑选出一种来作为加、解密信息。这两个密码中只要一个不对，那么破译出来的文件是读不懂的。

### ③加密

```
I=1
A=1
WHILE A
  GET# 1, I
  FOR X=0 TO 127
    TEXT=PEEK(AD 1+X)+INT(RND*255)
    IF TEXT>255 THEN TEXT=TEXT-256
    POKE(AD 2+X), TEXT
  NEXT
  PUT# 2, I
  IF I>=(LOF(1)/128) THEN
    A=0 I=I+1
  WEND
```

加密过程是通过GET读入128字节的文件资料，再对读入的128字节内容进行加密转换，然后通过PUT写入磁盘并由表达式 $I \geq \text{LOF}(1)/128$ 来判断文件是否已经读完。如果表达式成立，则说明文件已经读完，所有资料已被加密，即可以退出循环来关闭文件，这时加密过程就告结束。

### ④解密

```
I=1
A=1
WHILE A
  GET # 2, I
  FOR X=0 TO 127
    TEXT=PEEK(AD 2+X)-INT(RND*255)
    IF TEXT<0 THEN TEXT=TEXT+256
    POKE(AD 1+X), TEXT
  NEXT
  PUT# 1, I
  IF I>=(LOT(2)/128) THEN
    A=0
    I=I+1
  WEND
```

从中可以看出，解密完全是加密的逆过程。

## 四、编程和使用的二点建议

1. 若编程时提示信息用中文的话，则最好对编译程序(这里以IBM-PC BASIC CO-



MPILER 1.0版为例)作如下改动:

```
A>DEBUG BASCOM.LIB
-E   A 3 FF EB φ 3
-E   A 6 φD EBφ 3
-W
-Q

A>REN BASRUN.EXE BASRUN
A>DEBUG BASRUN
-E 2983 EB φ 3
-E 2AB7 EB φ 3
-W
-Q

A>REN BASRUN BASRUN.EXE
```

这样经编译后的程序当运行结束后,就不会再强行进入西文状态了。

2. 编译时不要考虑程序运行完一次后又重来一次运行的情况,因为编译后的程序不退出系统又运行第二次的话,尽管有时密码输入正确,也有可能出译码不正确的现象。另外,两个密码必须记住,哪怕差一点也不能解密成功。例如:密码为1000和1001,若输入1000和1000将只能译出一些乱七八糟的东西。

## 五、结束语

作者用BASIC编译的加、解密程序对WORDSTAR EDLIN、CDBASE等多种应用程序产生的文本文件进行加、解密实验,均得到了令人满意的效果。如果加密一个10多K字节的文本文件,一分钟内即可完成。

文献出处《新浪潮》1987年6期60—62页

## 也谈APPLE程序的保密

广 西 桂 宁

看了今年第1期《电脑与微电子技术》上刊登的张启峰同志的文章,受益不浅,关于其中文件名保密的问题,也谈谈笔者的一点看法。

所谓文件名保密,主要是利用不显示的控制字符(CTRL+字母键)加入文件名中,以达到保密文件的目的。解密的方法就是运行相关的程序后,这些隐蔽的字符就以反白的形式显示出来。但通过观察,发现有些控制字符还是不能显示出来,如CTRL-H(记为 $\hat{H}$ )。如能将其放入文件名中,就还能起到文件名保密的目的。由于 $\hat{H}$ 键的功能是退回一格,因

此是无法直接加入文件名中，只能通过程序来进行。具体方法如下：

一、将要保密的文件装入内存。其清单如图1所示。

图一：程序清单

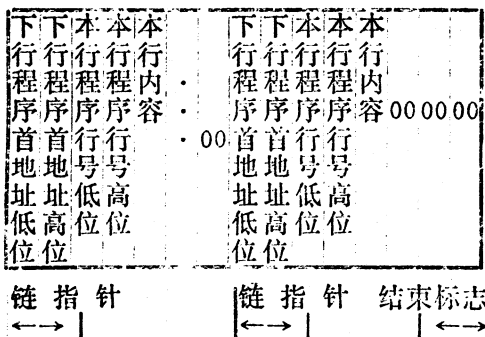
```

0 GOTO 10
1 A$="PA A"; PRINT CHR$(4)"SAVE"A$
2 END
9 REM PROGRAM
10 INPUT "T, S"; TK, SC
20 INPUT "READ(1) OR WRITE(2); "; RW
30 POKE 45121, RW
40 POKE 45975, TK; POKE 4597, SC
50 CALL 45111
60 POKE 72, 0
70 POKE 45121, 2
80 END
  
```

二、在程序的前面加入一小段程序(即图1中的0~2句)。其中0句的作用是DUN命令，不会产生存盘的操作。初始文件名为“PA A”。

三、使用CALL-151进入监控状。由于APPLESOFTBASIC程序在内存中是以链式结构(如图二所示)从\$801单元开始存放的。据此可知(参见图三)程序的第2句(即图1

图二：程序链式结构示意图



图三：监控状态下程序有效示意图

0800-	00	09	08	00	00	AB	31	30
0808-	00	25	08	01	00	41	24	D0
0810-	22	50	41	20	41	22	3A	BA
0818-	E7	28	34	29	22	53	41	56
0820-	5	22	41	24	00	2B	08	02
0828-	00	89	00	39	08	09	00	B2
0830-	20	50	52	4F	47	52	41	4D

0838-	00	4A	08	0A	00	84	22	54
0840-	2C	53	22	3B	54	4B	2C	53
0848-	43	00	69	08	14	00	84	22
0850-	52	45	41	44	28	31	29	20
0858-	4F	52	20	57	52	49	54	45
0860-	28	32	29	3A	22	3B	52	57
0868-	00	77	08	1E	00	B9	34	35
0870-	31	32	31	2C	52	57	00	8E
0878-	08	28	00	B9	34	35	39	37
0880-	35	2C	54	4B	3A	B9	34	35
0888-	39	37	2C	52	43	00	99	08
0890-	32	00	8C	34	53	31	31	31
0808-	00	A3	08	3C	00	B9	37	32
08A0-	2C	30	00	B0	08	46	00	B9
08A8-	34	35	31	32	31	2C	32	00
08B0-	B6	08	50	00	80	00	00	00

中的1句)的链指针为\$809,从\$809开始往下查找,就会发现字符串A\$就放在811~\$814中,其中空格键(代码为\$20,有关的代码值可参见苹果使用手册)就放在\$813单元中,

将该单元的内容改为 \$08 (H的代码)。

四、退回系统状态。这时的程序清单如图4所示。注意A\$已改为“PAHA”。

图四：程序清单（已加入H在A\$中）

```
0 GOTO 10
  A$="PA"; PRINT CHR$(4)"SAVE" A$
2 END
9 REM PROGRAM
10 INPUT "T, S"; TK, SC
20 INPUT "READ(1) OR WRITE(2): "; RW
30 POKE 45121, RW
40 POKE 45975, TK; POKE 4597
50 CALL 45111
60 POKE 72, 0
70 POKE 45121, 2
80 END
```

五、输入RUN1。这时文件就存入盘中，显示的文件为“PA”。即使运行了相关的解密程序后，显示的文件名仍是“PA”。“其真实文件名为PAHA”。

使用此法的优点在于：

一、由于H链的特点，只能通过程序来装入保密的文件，装入的方法与存盘的方法类似，只需将LOAD取代SAVE就行了。只有通过程序才能装入程序，这就更有利于保密。

二、由于H是退格键。它不仅取代相应的字符（如本例中的空格键），而且还将取代字符的前一个字符盖过（如本例中的第二个字符A）。因此，破密者要想装入保密的文件，不仅要知道H在文件名中的位置，还需要知道H字符的前一个字符是什么。而这两个字符都是不显示的，这无疑给破密者增加了困难，从而达到了保密的目的。

文献出处：《电脑与微电子技术》1987年2期24—25页

## 文件加密与解密

山西 裴兴龙

随着计算机网络技术的发展，信息在网上传输的安全成了一单独课题。为了保证明文文件（未加密）在传输过程中不被窃收（破译）需将明文文件加密成密文文件后再传输。密文文件再经过相应解密还原为明文文件。即文件的加密与解密。

本文采用运算法加密。是用BASIC编写了一个简单的加密、解密程序。它可以加密一切由EDLIN、Worstar、生成的文件、各种原程序、DBASZ—Ⅰ、DBDSZ—Ⅱ生成的ASCⅡ文件（•TXT、•PRG/CMD、•MEM•FRM等）、可以运行在一切微机上的。

```

10 REM《这是加密程序》
20 CLS:KEY OFF
30 INPUT "请键入加密的文件名(d:filename.ext):"; FILEINT$
40 INPUT "请键入加密后文件名(d:filename.ext):"; FILEOUT$
50 INPUT "请键入密码:只能是字符型"; G$
60 G$=MID$(G$,1,1)
70 OPEN FILEINT$ FOR INPUT AS#1
80 OPEN FILEOUT$ FOR OUTPUT AS#2
90 CLS:X$=HEX$(ASC(G$))
100 LINE INPUT #1, A$:RR=LEN(A$)
110 IF RR>127 THEN X$=A$:GOTO 180
120 CLS:LOCATE 5,35:PRINT "正在加密 稍候"
130 FOR I=1 TO RR
140 B$=MID$(A$,I,1)
150 E$=HEX$(ASC(B$)XOR 51)
155 IF LEN(E$)=1 THEN E$="0"+E$
160 X$=X$+E$
170 NEXT I
180 PRINT #2, X$
190 IF EOF(1) THEN 210
200 X$=HEX$(ASC(G$)):GOTO 100
210 CLOSE #1:CLOSE #2:SYSTEM

10 REM《这是解密程序》
20 CLS:KEY OFF
30 INPUT "请键入解密的文件名(d:filename.ext):"; FILEINT$
40 INPUT "请键入解密后文件名(d:filename.ext):"; FILEOUT$
50 INPUT "请键入密码:"; G$
60 G$=MID$(G$,1,1)
70 OPEN FILEINT$ FOR INPUT AS #1
80 OPEN FILEOUT$ FOR OUTPUT AS #2
90 CLS:X$=HEX$(ASC(G$))
100 LINE INPUT #1, A$:RR=LEN(A$)
105 IF MID$(A$,1,2)<>X$ THEN X$=A$:GOTO 180
120 CLS:LOCATE 5,35:PRINT "正在解密 稍候"
130 FOR I=1 TO RR STEP 2
140 B$=MID$(A$,I,2)
145 IF I=1 THEN 220
150 E$=CHR$(VAL("&h"+B$)XOR &H33)
160 X$=X$+E$

```

```

170 NEXT I
175 N=LEN(X$):X$=MID$(X$,4,N-3)
180 PRINT #2,X$
190 IF EOF(1) THEN 210
200 X$=HEX$(ASC(G$)):GOTO 100
210 CLOSE #1:CLOSE #2:SYSTEM
220 IF X$<>B$ THEN 210 ELSE GOTO 150

```

文献出处：《软件报》1988年2月20日

## PC—1500机程序的恢复

北 京 曹采发

BASIC语言程序用NEW指令清除后，如启动机器语言程序，能够自动寻找起始地址和终了地址，达到恢复BASIC语言程序的目的。

这个机器语言是浮动的，放在任何地址中都能起作用。但为了安全起见，一般的放在用户区最低地址之前，如8K模块则放在3808H至3816H地址中，16K模块则放在0008H至0016H地址中，即使按NEW指令也不会被破坏。这要不动用备用区地址，那么这个机器语言程序将永存于机内，供随时调用，十分方便、可靠、实用。

一、RAM区的地址分配（见下表）。

机 型 加 模 块	RAM区 最低地址	备用区地址	用户区 最低地址	RAM区 最高地址
PC-1500+CE-151	4000H	4008H~40C4H	40C5H	57FFH
PC-1500+CE-155	3800H	3808H~38C4H	38C5H	5FFFH
PC-1500+CE-161	0000H	0008H~00C4H	00C5H	47FFH
PC-1500A+CE-161	0000H	0008H~00C4H	00C5H	57FFH
PC-1501+CE-161	0000H	0008H~00C4H	00C5H	57FFH

二、机器语言程序的输入：

机器语言程序：

F4 78 65 69 00 B5

FF 64 27 99 04 F6

78 67 9A

如8K模块。

POKE &3808, &F4', &78 &9A

如16K模块：

POKE &0008, &F4, &78 &9A

三、启动:

如BASIC语言程序用NEW指令清除, 欲想恢复:

8K模块用CALL &3808 ✓

16K模块用CALL &40008 ✓

在 (PRO) 状态下, 按  $\downarrow$  键即显示源程序, 达到了恢复BASIC语言程序的目的。

文献出处: 《软件报》1988年3月5日

## LASER机程序中部分程序段的删除方法

秦 皇 岛 李 琪

LASER机BASIC设有DELETE (删除) 命令, 便于剪辑程序, 如下简便方法可删除大段程序行。

方式 1: 执行如下三步:

( 1 ) 0 POKE31480, 182:D < Line1 > - < Line2 > :END ✓

( 2 ) RUN ✓

( 3 ) 0 ✓

用LIST命令可见 < Line1 > (指欲删除程序段的首行号, 但最好不是全程序的首行号) 至 < Line2 > (指 < Line1 > 后面的任一行号) 之间的程序段已被删去。

方式 2:

< 1 > 0 POKE 31480, 182:D < Line2 > :END ✓

< 2 > RUN ✓

再用LIST列出清单, 可见到 < Line2 > 前面所有程序行 (包括本方式功能行 0) 均被删去。

反复使用这些方法, 可随意方便地删除任意大段大段的程序段。(行 0 中字符 "D" 可换为除 ":" 外的任意文字字符)

10 A= 1 : B= 2 : C= 3

20 PRINTA;

30 PRINTB;

40 PRINTC;

50 END

0POKE31480, 182 : D20—30 : END

RUN

READY

例 1

```

0
LIST
10 A=1 : B2 : C=3
40 PRINT C
50 END
READY
10 PRINT 1 ;
20 PRINT 2 ;
30 PRINT 3 ;
40 PRINT 4 ;          例 2
50 END
READY
0 POKE 31480, 182 : D-30 : END
RUN
READY
LIST
40 PRINT 4,
50 END
READY
COPY

```

文件出处：《软件报》1987年8月16日

## 用PC—TOOLS工具盘拷贝加密盘

西 安 汪佳莹

中国科学院希望电脑公司为IBM-PC个人计算机设计的汉字操作系统H-DOS同电子部六所研制的CC-DOS相比有其独到之处，尤其在汉字压缩字库技术等方面是出类拔萃，具体优点在以前的《软件报》上已有专门的介绍，这里就不再重复了。

H-DOS采用了专门的加密技术，使得IBM-PC上流行的COPYWRITE、COPYIIPC等高级拷贝工具在其面前无能为力。

笔者，无意中利用，PC—Tools' 工具盘中的COPY命令将H-DOS软盘复制成功。以前我们在希望公司购买的HCAD仅有一份，无法制作备份软件，使用时总是提心吊胆，现在利用'PC—Tools'软盘将含有H-DOS的HCAD软盘作了备份软件使用起来再不必为没有备份而发愁了。

文献出处：《软件报》1988年4月23日

## 二字节文件解除各种版本BASIC的“P”

关于BASIC中的加“P”命令的去密方法过去已谈及许多，看来主要是解二个问题：

- 1、各种版本的BASIC程序在缓冲区的首地址。
- 2、加“P”命令后首字节为FE，现想介绍一种编制容易，使用方便的仅有2个字节的文件，以解决上述的问题，极为容易地解除“P”命令。编制 jPM.BAS如下：在DOS环境下

```
>debug✓  
—E CS:0100 FF 01✓  
—N jPM.BAS✓  
—R CX✓  
  0000  
  :0002✓  
—W  
—Q
```

现在在盘中已形成一个二字节的jPM.PAS文件，使用时，只需在BASIC状态下：

```
Load“文件名”✓  
这文件名指加过“P”命令的  
Load“jPM.BAS”✓  
List✓
```

即列出加过P命令的文件清单，若要保存则用SAVE时不加P即可。

如果BASIC版本不同（即首地址不同），可用搜索命令S查找程序首地址中的“FE”。找到后即用地地址作为首地址作以上步骤。

文献出处：《计算机世界》1987年12月8日

## 恢复内存中（或加密）BASIC程序的一种简易方法

石家庄 陈 凯

BASIC应用程序被装入内存时，其首标志是固定的。对于删除（即NEW）和加密的BASIC程序，一般都是对首标志进行了修改。笔者使用DEBUG调试程序建立了一个3字节的文件达到了恢复内存中（或被加密程序不能显示）BASIC程序的目的。建立文件的方法步骤是：

```
C) DEBUG ↓
```



```

-E100 ↓
3DBC: 0100 00.FF 00, 00.10 ↓
-RCX ↓
CX 0000
: 3 ↓
-N TEST.BAS ↓
-W
Writing 0003 bytes
-Q ↓
C)

```

使用3字节的TEST.BAS文件时，只要把TEST.BAS文件装入（即LOAD）内存，就能将已经删除（即NEW）或经过加密的BASIC程序恢复到内存并可显示文件清单。

文献出处：《软件报》1988年5月14日

## IBM—PC/XT磁盘文件的一种保护方法

武汉 陆学斌

在使用“DEL”或“ERASE”等文件具有破坏性的指令时，有时会误将您珍贵的文件删除掉，这种差错在无写保护功能的C盘上最易发生。这里介绍一种简单方法使您避免上述差错，当然您如果不想让别人修改或删除您的某些文件时，也可使用这一方法。

我们知道，文件有四种属性：普通、隐含、只读、系统。通过改变文件的属性即把普通文件变成只读文件即可达到保护的目的。

例：A盘上有一名“BASICA.COM”的文件，把它变成只读文件的步骤如下所示。其中，单引号内的是文件名字。如需改变别的文件属性，只需改动单引号内的文件名字即可。如果想改变C盘内的某文件属性，只需在C)下重复上述步骤。

最后说明一点，如果您想把已变成只读的文件删除掉，需要再次改变该文件的属性，具体的步骤与上面所述一样，只不过把MOV CX, 01改成MOV CX, 00即告完成。

```

A) DEBUG ↵
- A100 ↵
09CD:0100 DB 'BASICA.COM' , 0, 0 ↵
09CD:010C MOV AX, 4301 ↵
09CD:010F MOV CX, 01 ↵
09CD:0112 MOV DX, 100 ↵
09CD:0115 INT 21 ↵
09CD:0117 INT 20 ↵
09CD:0119 ^C
- G=100 ↵

```

文献出处：《软件报》1988年5月14日

# 探讨关于修复dBASE III中数据文件的方法

武汉计算机应用开发所 王 述

在使用dBASE III时,由于某种原因可能会发现数据文件被破坏的现象。下面从两个方面加以探讨。

一、打开库后发现记录个数比上次所建立记录个数少若干。一般可采取这样的步骤。

- ① USE 文件名
- ② GO Bott
- ③ Skip

可以重复的步骤③,若能用List显示出下面的记录则修复成功;若屏幕仍显示: End of file encountered 则可用,

- ① GO Bott
- ② DELE
- ③ PACK

若能用List显示出下面记录则修复成功;仍不成功转二。对于上面情况可能伴随着这样一种情况:打开库索引,发现其记录比库中记录少许多,则可采用下面步骤解决。

- ① USE 库+索引
- ② Reindex

二、打开库后发现大量的记录被丢失并伴随产生紊乱信息。产生这种情况多半是正在对库文件执行操作时突然掉电。对此采用一中方法不能奏效时,可采用下述步骤,这里以一实例加以说明。

笔者编制的人事信息管理子系统中有一个履历信息库UL.dBF,原先该库中有275条记录,掉电后再打开发现只有25条记录,为此,找一个格式化好的空白盘,将UL.dBF拷贝上去。

```
C>Recover A:UL.dBL
C>REN A:File 0001 . REC A:UL
C>debug A:ULD 80, 190
-D1A0, FFF
```

发现第26条记录外(OA 00开始)有一特殊字符(·)而其上下相应位置都是空格,去掉该字符:

```
-E0A01
-0A01 . 1A . 20
-D0A00 (发现 '·' 字符消去了)
-W
-Q
C>Ren A:UL A:UL.dBF
```

C>dBASE

· USE A · UL

· LIST

此时275条记录全部显示出来，至此修复成功。

文献出处：《计算机世界》1987年8月23日

## APPLE I 机的dBASE II 命令文件保密

广州 陈文

APPLE I 用户编辑dBASE II 命令文件，一般都是为了更有效地管理某些数据库内的数据。我们有必要也对管理这些数据库的命令文件作保密工作，防止别人盗取库内需要保密的数据。下面，是由本人设计并已在APPLE I 微电脑上调试运行通过的程序。

在程序中，第3、4语句是分别给变量CHEN和WEN赋以6个空格字符；第5语句是给变量N1赋以数值13；第6语句是给变量N2赋以数值6；第7语句是产生一个本身由用户拟定给程序的密码“□□1966”（其中“□”代表空格字符）并把该值赋给变量CHEN；第8语句与第9语句是要求试图调用命令文件XYZ的用户输入密码；第11语句是核对用户输入的密码是否正确，如果正确，则执行命令文件XYZ，否则，就退出本程序并返回到dBASE II 系统状态。

最后，还需一提的是该程序的设计思想：1. 提高密码设计技巧；2. 检验密码是否正确的条件语句不应明示密码。由于程序本身拟定的密码由第7语句产生，产生的密码是“□□1966”这样，我们平时可以仅记密数字“1966”而实际输入密码时，我们得先按下两次空格键，接着键入“1966”这四个数字。另外，在第11语句进行密码核对时，并没有明显列示密码。由此，无疑更有效地保护了命令文件本身。

本保密程序经适当修改，可以在IBM PC及其兼容机上运行。

```
1 SET TALK OFF
2 ERASE
3 STORE "      " TO CHEN
4 STORE "      " TO WEN
5 STORE INT (&.78) TO N1
6 STORE (LEN (WEN) TO N2
7 STORE ("□□1987□□8□□2□□1966□12□□5□□",
  N1, N2) TO CHEN
8 @ 11, 11 SAY "PLEABEGET TO CIPHER. "GET WEN PI CTURE
  "XXXXXX"
9 READ
```

```

10 SETEXACT ON
11 IF WEN = "&CHEN"
12 DO XYZ
13 ELSE
14 ERASE
15 @ 11, 11 SAY "YOU CAN'T TO DO THIS PROGRAM,"
16 CANCEL
17 END IF
18 SET TALK ON

```

文献出处：《软件报》1988年3月19日

## 如何修改磁盘文件的卷名

江 苏 曹 勇

今年初我们对1987年统计数据进行了整理、归档处理。本着节约的原则，整理出几十张没有保留价值的软盘（把其中不要的文件删掉），为了把指定的卷名写到软盘上去（这是档案管理技术的要求），按常规方法需要用FORMAT/V命令重新格式化，但这种方法既费机时又缩短软盘驱动器的寿命。下面介绍一种方法不需要重新格式化软盘而达到同样的目的。

使用调试程序DEBUG来对卷名进行修改，具体步骤如下：

1. 调用DEBUG程序，在DEBUG命令提示符下用装入命令L把软盘上逻辑扇区号为5的目录区内容装入微机内存；
2. 用修改内存命令E，把对应的内存单元修改成所要求的卷名（十六进制ASCII码），规定西文卷名的字符数≤11个，汉字≤5个，不足11个用ASCII码20H填满，位于第十二个字符处键入08（不能漏掉）；
3. 用写盘命令W将修改过的内容写到软盘的原来位置上，退出DOS。至此，完成整个修改过程。

这里举的2个实例是在PC兼容机M24机上进行的，使用西文MS-DOS 2.1及中文CC-DOS 2.1操作系统，完成卷名修改的。

### 1. 软盘上没有DOS系统

要求将原卷名“86工业数据”改成“泗洪统计局”。在DEBUG提示符下键入汉字需要把汉字转换成十六进制内码，具体转换法可参照有关书籍。

经查表（国际GB 2312—80）并且转换后“泗洪统计局”的汉字内码为E3 F4 BA E9 CD B3 BC C6 BE D6。

A>C, DEBUG←

—L 0 0 5 1←; A盘扇区号为5的内容装入内存首址0 0 H处

—E 0 0 E 3 F 4 BA E 9 CD B 3 BC C 6 BE D 6 20 0 8←; 修改内存单元(注意:每次都从首址0 0 H处修改)

—W 0 0 5 1←; 将修改过的内容存盘

—Q←; 退出DEBUG, 返回DOS

## 2. 软盘中有DOS系统

修改方法及步骤与上述相同, 内容相异之处是把E命令右边的首地址修改成60H。

硬盘卷名通常不需要修改, 确需修改可仿照上几例也能实现, 只是硬盘的卷名位于DOS分区内逻辑扇区号为11的位置上, 由读者自己去完成, 不再赘述。

几点说明: ①. 使用西文DOS的DEBUG程序在提示符下是不能显示汉字的, 但把卷名在内存相应的单元修改成汉字的十六进制内码存盘后, 在CC—DOS下即可显示汉字, 结果相同; ②. 对硬盘进行这项工作一定要谨慎行之, 万一误操作就会丢失文件, 建议你在软盘上熟悉DEBUG命令之后再着手这项工作; ③. 选择FORMAT命令的其它参数(如/1, /8, /B等)格式化过的软盘, 其卷名修改方法读者可以仿照上述去完成。

文献出处: 《软件报》1988年4月23日

# 大批量文件在硬盘上的保护方法

沪建一公司 葛自伟

对于大批量文件一般都集中在一个子目录中使用。其保护方法, 虽然可以象单个文件一样, 用不可显示的区位码写子目录名或把目录项中第11字节(文件属性)置为秘密文件(12 H)不被显示出来, 但是还容易被熟悉的人改名或解除加密。笔者采用修改目录项中第26、27字节的方法来进行保护, 使别人无法进入子目录进行有关操作。

具体步骤如下:

(1) 在根目录下建立一个子目录。

C: MD CHINESE

(2) 把大批文件拷入此子目录。

(3) 进入DEBUG, 打印出此目录项内容。

(4) 建立修改及恢复文件KEYW、KEYWU。

A: type keyw

L 0 2 10 2

e2d8 00 00 00 00

w 0 2 10 2

Q

A: type keywu

```
L 0 2 10 2
e2d8 21 0A 16 00
w 0 2 10 2
Q
```

(5) 建立修改及恢复批处理文件。

```
A : type keyw.bat
ECHO OFF
TYPE A : KEYWIDEBUG
CLS
A : type keywu.bat
ECHO OFF
TYPE A : KEYWUIDEBUG
CLS
```

至此，只要每次在使用子目录后退回主根，执行A，KEYW，再热启动，就把该子目录锁上了。当要使用时可在主根下执行A，KEYWU，再热启动即可恢复，进入子目录。执行前应该意把DEBUG.COM文件拷在主根下或在A盘上。

DEBUG-86 version 2.11

>L 0 2 10 2

>d280

```
0DC2 : 0280 43 4F 4E 46 49 47 20 20-53 59 53 20 00 00 00 00 CONFIG SYS....
0DC2 : 0290 00 00 00 00 00 00 20 04-21 0E 14 J0 45 00 00 00 ..... !...E...
0DC2 : 02A0 A9 A1 A9 A2 A9 A3 20 20-20 20 20 10 00 00 00 00 )!)")# .....
0DC2 : 02B0 00 00 00 00 00 00 C5 04-21 0A 2B 03 00 00 00 00 .....E.!+.....
0DC2 : 02C0 43 48 49 4E 45 53 45 20-20 20 20 10 00 00 00 00 CHINESE .....
0DC2 : 02D0 00 00 00 00 00 00 08 00-21 0A 16 00 00 00 00 00 .....!.....
0DC2 : 02E0 4D 50 20 20 20 20 20 20-20 20 20 10 00 00 00 00 MP
0DC2 : 02F0 00 00 00 00 00 00 2D 00-21 0A 52 01 00 00 00 00 ..... !.R.....
```

>a

文献出处：《计算机世界》1987年9月23日

# COMX机数据保护妙法

福建省仙游一中 陈开雄

如果用户的存储空间同时包含有程序和数据,那么对程序的任何编辑都会把数据抹去(详见《软件报》87年第10期)。解决的办法通常用DSAVE和DLOAD指令,但这得用到录音机,很不方便。其实我们可以用先把程序空间后面的数据空间往后移,等程序编辑完后,再把数据空间重新连接在程序空间后面的办法来解决。我编了一个子程序,请在输入你的程序之前,先输入这个子程序。那么,当你进行程序编辑时,先键入RUN6000✓,接着子程序会问你要把数据空间往后移多少页(每页256个字节,其值要视新输入的程序部份长度来定)。接下去,子程序就进行将数据空间往后移工作,如果数据较多,可能得运行较长一段时间。等程序编辑完后,请键入RUN6500✓,则数据恢复了。

在子程序中,用到了D、X等简单变量,那么它们的值会不会因程序的编辑而失去呢?不会的,因为A、B、...、Z等26个简单变量的值是贮存在4300至4368这一段内存里,每个占4个字节,而带足码的简单变量、字符串、数组是贮存在程序空间后面的。

```
6000 D = PEEK (@4283) * 256 + PEEK (@4284)
6010 X = PEEK (@4292) * 256 + PEEK (@4293)
6020 Y = PEEK (@4294) * 256 + PEEK (@4295)
6030 Z = PEEK (@4299) * 256 + PEEK (@429A)
6040 INPUT N: FOR I = D TO Z: POKE (I + N * 256, PEEK (I)):
      NEXT: END
6500 E = PEEK (@4283) * 256 + PEEK (@4284) - 1: D = D + N * 256
6510 POKE (@4292, INT ((X + E) / 256)): POKE (@4293, MOD (X + E,
      256))
6520 POKE (@4294, INT ((Y + E) / 256)): POKE (@4295, MOD (Y + E,
      256))
6530 POKE (@4299, INT ((Z + E) / 256)): POKE (@429A, MOD (Z + E,
      256))
6540 FOR I = D TO Z + N * 256: POKE (I - N * 256 + E, PEEK (I)):
      NEXT: END
```

文献出处:《软件报》1988年6月4日

# APPLE—II DOS 3.3 磁盘文件的删除与恢复

饶朝学

在DOS 3.3下，倘若你不小心，把一个文件误删掉了，不要紧，只要你暂不往该磁盘上写入新文件，被误删的文件就可以轻而易举地取回来。下面，我们先来看看DOS 3.3如何存取一个文件。

我们知道，DOS 3.3的磁盘格式是这样的：每片磁盘划分为35个轨道（TRACK），每个轨道上又分为16个段落（SECTOR），每个段落可存放256个字节的数据。DOS存取文件将首先从第17（\$11）轨道开始操作。在第17轨道上除第0段落存放了该磁盘的内容表（VTOC）外，其余15个段落作为文件目录段落使用，存有文件的索引进入点。第17轨道的段落组织如图1所示。

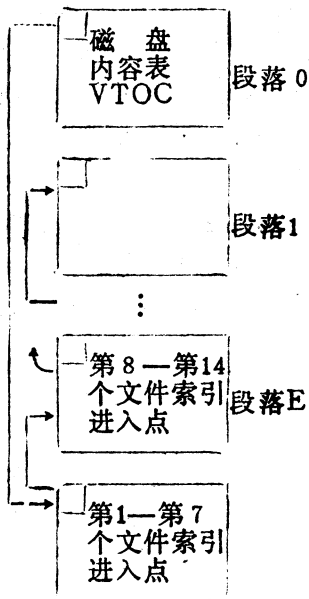


图1

各个目录段落的内容如下：

字节	内容
00	未使用
01	下一目录段落之磁道号
02	下一目录段落之段落号
03~0A	未用
0B~2D	第一个文件的索引进入点
2E~50	第二个文件的索引进入点

上下类推。

其中，索引进入点的格式是这样的：

相对字节	内容
00	该文件存放第一个轨道/段落表的轨道号
01	该文件存放第一个轨道/段落表的段落号

02	文件类型标志
03~20	文件名
21~22	以段落表示的文件长度

所谓轨道/段落是这个意义：每一个文件都要用一个段落来记录该文件所占用的段落及其磁道号，而记录这些情况的表则称为磁道/段落表。

我们再来看看DOS寻找一个文件的过程（图2）。

首先，DOS会逐个访问目录段落，直到找到文件名相符的文件为止，然后，根据该文件的索引进入点，找到其磁道/段落表，从中得到该文件存放数据的磁道及段落号。

当使用DELETE命令删去一个文件时，DOS将在目录段落中，把该文件的索引进入点



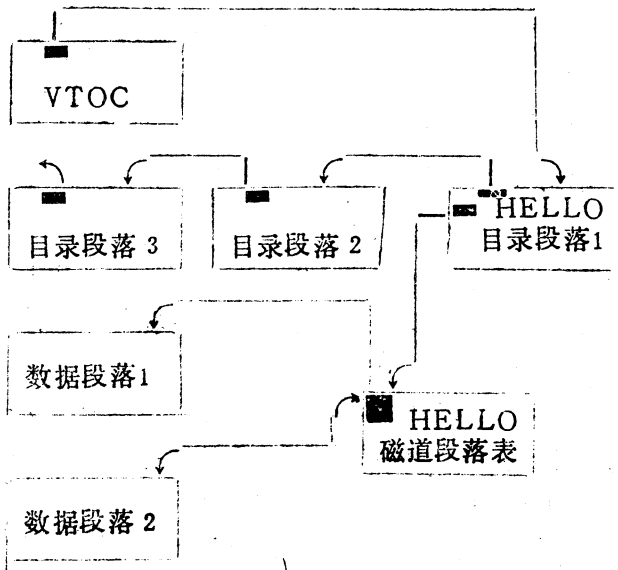


图 2

第二步的将索引进入点中的FF标志改回来。为此，我们可以调用DOS的驱动程序（RWTS），将该目录段落读入内存，经修改后再写回磁盘。调用RWTS前，须填写参数表，使RWTS能正确地工作。限于篇幅，在此不详述，有兴趣的读者，可以参阅所附程序中的程序（\$805F），该子程序将把目录段落读入内存\$8100~81FF的缓冲区，由子程序（\$8094）修改后，再写回磁盘。至此，误删的文件已可取回。

本文的最后，给出一个可以完成以上操作的程序。该程序会要求你输入待恢复文件、在CATALOG命令列示文件目录时所处的序号（十进制）。另外必须注意的是，此程序运行后，尽管文件已经取回，但该文件所占用的空间已经在DELETE命令释放了。为此，需将该文件用SAVE命令再存盘，以免被以的新文件所破坏。

#### 机器码程序

• 800.80C6

08-0020	38 FC 20 E4 FB 20 BB	8068-	A0 04 91 00 A5 03 A0 00
8008-	80 A2 00 20 75 FD A0 00	8070-	91 00 A0 08 A9 00 91 00
0810-	20 A7 FF A5 3E A0 0F F8	8078-	C8 A9 81 91 00 A5 04 A0
8018-	C9 08 90 08 38 E9 07 D8	8080-	0C 91 00 A9 00 A0 03 91
8020-	88 4C 17 80 D8 98 85 03	8088-	00 20 E3 03 20 D9 03 A9
8028-	A9 11 85 02 A9 01 85 04	8090-	00 85 48 60 A2 00 BD 00
8030-	20 5F 80 20 94 80 A9 02	8098-	81 C9 FF 80 09 E8 8A C9
8038-	85 04 20 5F 80 A9 CF 20	80A0-	FF B0 17 4C 96 80 8A A
8040-	ED FD A9 CB 20 ED FD 60	80A8-	18 69 20 AABD 00 81 998
8048-	A0 A0 D2 C5 C2 CD 05 CE	80B0-	00 81 A9 A0 9D 00 81 4C
8050-	A0 C5 CC C9 C6 A0 C5 CB	80BB-	90 80 60 A2 17 BD 47 80
8058-	D4 A0 D2 C5 D4 CE C5 20	80C0-	20 ED FD CA D0 F7 60
8060-	E3 03 84 00 85 01 A5 02		

文献出处：《电子与电脑》1987年12期7—8页

第一个字节置成FF，而把该字节所记录的磁道/段落表磁道号转抄至索引进入点第20字节，另外，DQS还将该文件所占用的空间释放，这一点记录在VTOC表中。

了解了DOS文件的管理，要恢复误删的文件就不难了。

首先，必须确定待恢复的文件的索引进入点在哪一个目录段落。用CAT-ALOG命令把磁盘上文件列示出来，我们即可知道该文件原来排在第几位，根据图1，可查得它的索引进入点的目录段落号。