

# CCED 文件以假乱真巧加密

山东省机械工业学校计算机教研室 赵欣

CCED 是一种深受用户欢迎的字表处理软件。CCED3.0 以上版本新增了给用户文件加密的功能。然而美中不足的是其加密文件较容易被译。为什么能够破译呢? 一是因为加密算法是一定的, 二是因为加密文件本身又包含着破译所需的信息。如果我们把破译信息“藏起来”并且以假乱真, 别人就无法破译你的加密文件了。笔者在此介绍一种算法很简单而加密效果极好的方法。

为了说明加密原理我们必须了解 CCED 加密文件的结构。CCED 加密文件包括文件头和已加密正文两部分。以 CCED3.3 或 CCED4.0 版为例(CCED5.0 笔者未用过, 所以没提及), 其文件头第 1—34 个字节是加密文件标志, 实际上是一句英文信息, 警告不要修改此加密文件。第 35 个字节是译码表基值, 它是由加密时输入的加密口令经一定的运算而得到的。所以不同的口令产生的译码表基值不同, 而用该基值经运算而产生的加密正文内容也就不同, 从第 36 个字节开始存放经一定运算之后形成的密文加密口令。要给 CCED 的加密文件二次加密, 只需要修改第 35 个字节就可以了。把真正的基值以变量形式保存在一个内存变量文件中, 而在第 35 个字节上换上一个假基值。这样以假乱真之后, 虽然密文形式的加密口令没修改, 但是运行 CCED 再输入你当初加密时的口令, CCED 认为口令错, 拒绝将此文件调入内存, 所以别人即使能推算出你的加密口令也无济于事。另外, 已加密的正文部分也可以很好地保密, 因为译码基值是假, 而且与真基值毫无关系, 因此无法根据假基值去推算出真基值。所以即使别人用 DEBUG 或 PCTOOLS 显示文件, 也无法进行破译。而当自己需要解密时, 只需把自己保存的真基值写回第 35 个字节上, 文件就解密了, 再进入 CCED 输入你当初的加密口令就可将文件调入内存了。

这种加密和解密程序已在 FoxPro for Dos 2.5 下运行通过, 成功地为用户 CCED 加密文件完成了二次加密和解密。并且如果加密的是源程序文件, 待二次解密后, 照样可以正确地运行。由于 CCED 给文件加密时, 把文件属性改成只读文件了, 进入 PCTOOLS 可以看到文件属性显示 RA。而运行加密程序时, 需用 FoxPro 的低级文件函数对文件进行写操作, 因此, 在运行加密程序之前, 要先进入 PCTOOLS 用 Attrib 功

能把属性改为 A(表示归档)。

如果你在输入以下的加密程序或解密程序时输错了一句, 使程序运行时因出错而中止, 这时可能还没执行到关闭加密文件的语句, 这样你可在 FoxPro 命令窗口中发 = FCLOSE (FH)命令关闭此文件, 否则易造成文件数据混乱。因 FH 是全局变量, 所以程序中止时仍然存在。

\* 给 CCED 的加密文件二次加密 FPY.PRG

```
CLEAR
CLEAR ALL
SET TALK OFF
SET SAFE OFF
PUBLIC FH
ACCEPT "请输入二次加密的文件名(包括盘符路径和扩展名)"
TO FN1
ACCEPT "请输入 CCED 密钥文件名(包括盘符路径, 但不输入
扩展名)" TO FN2
FH = FOPEN(FN1, 2)
= FSEEK(FH, 34, 0)
RS = FREAD(FH, 1)
SAVE TO &FN2 ALL LIKE R * && 真译码表基值 RS 存盘
= FSEEK(FH, 2, 0) && 将文件的第 3 或第 4 个字符
FS = FREAD(FH, 1) && 作为假译码表基值
IF RS = FS && 写入文件第 35 个字节
FS = FREAD(FH, 1)
ENDIF
= FSEEK(FH, 34, 0)
= FWRITE(FH, FS, 1)
= FCLOSE(FH)
SET TALK ON
SET SAFE ON
RETURN
```

\* 给 CCED 二次加密的文件解密 SKEY.PRG

```
CLEAR
CLEAR ALL
SET TALK OFF
SET SAFE OFF
PUBLIC FH
ACCEPT "请输入解密的文件名(包括盘符路径和扩展名)"
TO FN1
ACCEPT "请输入" + FN1 + "的密文件名(包括盘符路径, 但不输
扩展名)" TO FN2
RESTORE FROM &FN2 ADD1 && 将真译码表基值调入内存
FH = FOPEN(FN1, 2)
= FSEEK(FH, 34, 0)
= FWRITE(FH, RS, 1) && RS 写回文件
= FCLOSE(FH)
SET TALK ON
SET SAFE ON
RETURN
```