

# ROCKEY6 SMART 远程升级策略



## 摘要

在软件开发商使用加密锁保护软件时，由于加密锁做为一个实物，在发售和升级时不能通过 Internet，而只能通过邮寄等物理手段进行，而纯软件的发售和升级却完全可以在 Internet 上进行，因此使用加密锁进行远程升级的功能十分必要。

ROCKEY6 SMART 内部集成了三种远程升级：远程标志升级、安全文件从传输和远程模块管理。

远程标志升级是通过安全的方法修改用户端加密锁远程升级标记，使得用户端内部程序可以根据这个标记进行判断来控制执行不同的动作，或者结合后面两种方案来进行升级。

安全文件传输是通过将明文文件加密，提供给用户后，在用户端加密锁内部将密文还原成明文，在这个方案中，直接替换锁内部的任何文件，功能比较大，而且使用广泛，也是本文描述的重点。

远程模块管理是专门针对可执行文件，软件开发商通过安全的方式来控制用户端相应的一个或一组可执行文件的允许运行情况。

本文针对软件开发商如何控制进行远程升级的基本方法进行介绍，有关远程升级标志、安全文件传输、远程模块管理等理论性内容，请参考《用户手册》

## 一、远程标志升级

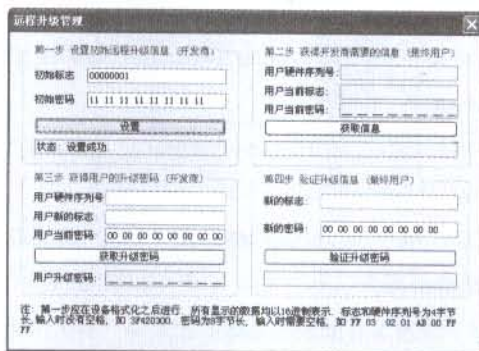
远程标志升级是 ROCKEY6 SMART 远程升级的最基本的方法，它主要是通过“一次一密”的升级方法，即每次升级数据只能使用一次。

远程标志的升级主要有两个作用，一是修改标志，将用户端加密锁的远程升级标志修改，以达到

控制程序执行的目的，二是临时提升一个远程升级密码验证的权限，供另外两种升级方案使用。下面我们就来看看 ROCKEY6 SMART 是如何进行远程标志升级的：

使用远程标志升级需要通过如下几个步骤：

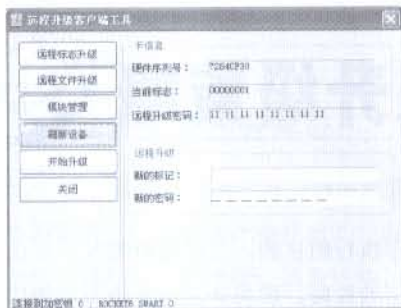
1. 开发商在出厂前，需要设置远程升级初始标志和初始密码。在这里，即使您暂时不打算使用此功能，我们也同样建议您设置一个初始的标志和密码，以为日后不时之需。如下图所示：



图一

这一步骤是在开发商在加密锁交给最终用户前做的初始化工作，这个工作只能做 1 次，必须对加密锁进行重新格式化之后才能重新设定基本远程升级密码。例如，在本例中，我们将远程升级标志和升级密码分别设置为：“0x00000001”和“11 11 11 11 11 11 11 11”，在实际使用中，您可以使用一个较为复杂的十六进制串来作为初始的远程升级密码。

2. 在用户使用过程中如需要升级，则向开发商提供硬件信息，这个信息包括硬件序列号、当前标志和当前密码，当然如果开发商已经记录了这些信息，就不必要求用户提供这些信息。



图二

在本例中，通过第二步，您的用户可以使用“远程升级客户端工具”获得该用户手上锁的相关信息，用户可以把这些信息通过各种方式提供给您，如果您打算在您自己的程序中集成这个功能，可以使用如下的代码片断来获得这些信息：

```
unsigned char RemotePass[8];
// 在这里进行打开锁操作
errcode = DIC_Command(hic, GET_REMOTE_INFO,
cmddata);
RemoteTag = DIC_Get(cmddata, REMOTE_TAG,
BY_VALUE, NULL);
DIC_Get(cmddata, REMOTE_PASS, BY_ARRAY,
remotePass);
// 获得当前加密锁的硬件 ID
errcode = DIC_Command(hic, GET_HARDWARE_INFO,
cmddata);
HardSerial = DIC_Get(cmddata, HARD_SERIAL,
BY_VALUE, NULL);
m_HardSerial.Format("%08X", HardSerial); //
格式化硬件序列号字符串
m_RemoteTag.Format("%08X", RemoteTag); // 格
式化远程升级标志字符串
// 格式话远程升级密码字符串
m_Password.Format("%02X %02X %02X %02X %02X
%02X %02X %02X", RemotePass [0],
RemotePass [1], RemotePass [2], RemotePass
```

[3],

RemotePass [4], RemotePass [5], RemotePass [6], RemotePass [7]);

```
// 关闭锁操作
```

3. 开发商生成远程升级密码，这里需要注意的是，如果远程升级标志的最高位为1（即第一个字符大于“8”），表示用户升级密码是和硬件序列号相关的，这样，即使所有用户标志都相同，但是生成的升级密码也全不相同，否则在生成新的升级密码时，和您输入的“用户硬件序列号”无关。

4. 用户通过新的远程升级标志和新的密码来进行升级，升级之后用户端加密锁内部的远程升级标志被修改为新的标志。这一过程可以使用我们提供的“客户端远程升级工具”来进行，也可以在程序中通过如下的代码即可完成：

```
// 设置升级标志为开发商生成的新的远程升级标志 (0x02)
```

```
DIC_Set(cmddata, REMOTE_TAG, BY_VALUE, 0x02,
NULL);
```

```
// 为了简便说明，这里将新的密码全部设置为 0x22
```

```
memset(RemotePass, 0x22, 8);
```

```
DIC_Set(cmddata, REMOTE_PASS, BY_ARRAY, 0,
RemotePass);
```

```
// 进行验证远程升级标志和远程升级密码操作。
```

```
iRet=DIC_Command(Hic, CHECK_REMOTE_INFO,
&rInfo);
```

通过以上的步骤，您就成功将您的用户手上的加密锁的标志替换为您指定的标志了，如果您单独使用本方案来实现升级的目的，可以在出厂前将算法预置在加密锁内，通过内部程序采用如下的方法进行判断：

```
dword dwTag;
```

```
// 获得远程升级标记
```

```

get_remote_tag(&dwTag);
if(dwTag & 1)
    ;//Do something
esle if(...)
    ;//Do something else
else
    ...

```

这样，想必各位开发人员比我们会更清楚如何将自己的哪些算法放入到加密锁中了。

## 二、安全文件传输

在安全文件传输这一方案中，它是可以和第一种方案结合使用的，在 ROCKEY6 SMART 中，生成密文操作是和远程升级第三步“获得升级密码”中获得的密码相关的，而生成明文文件是和用户进行远程升级时的验证升级密码相关的，因此，软件开发商进行生成密文文件前做了“获得升级密码”操作，那么该密文文件就是和这个升级密码相关的，在用户进行生成明文文件操作的时候，就需要这个新的标志和新的密码才能完成升级。

这个操作的基本步骤如下：

(1) 开发商首先用自己手上的加密锁将文件生成“密文文件”。

(2) 把这个密文文件（同时可能还需要远程升级标志和远程升级密码）发送给用户。

(3) 用户使用“远程升级客户端工具”或软件开发商提供的程序把“密文文件”送入加密锁，由加密锁在锁内还原成明文文件。

整个安装文件传输方案的实施步骤可以参看图八。

假设您所使用的可执行文件名称和 ID 分别为 0x1000 和 0x0001，结合前面远程标志升级的内容，您在升级这个文件使用安全文件传输就有如下三个方案：

### 1. 所有用户完全相同

通过安全文件传输的特征得知，当您插入加密锁

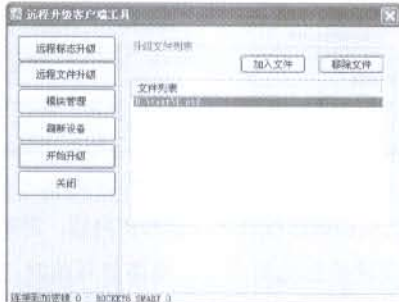
后不进行生成新的远程升级密码操作，生成的密文文件就可以看作是通用的，那么任何属于您的用户不需要验证远程升级密码（也不能验证，如果此时是验证过的，那么需要重新拔插一下锁）即可完成升级。

通过这种方式非常简单，您不需要考虑远程升级的问题，当进行升级的时候，您可以使用 IDE 将您待升级的可执行文件通过“生成密文文件”操作进行加密，将加密后的密文文件（即生成密文文件操作的输出文件）发送给用户使用。



图三 生成密文文件

用户获得这个密文文件之后，就需要通过“生成明文文件”操作进行解密，解密后这个文件就以明文的形式保存在锁内，用户就可以正常使用升级后的软件了，用户可以使用“远程升级客户端工具”来完成升级。



图四 用户使用工具进行升级

如果您打算将这个升级功能集成在您的软件中，

您需要首先了解 PLAINTEXT\_FILE 这个宏，这个宏就是 ROCKEY6 SMART 用来进行生成明文文件操作的，把这个宏传递给 DIC\_Command，并在 cmddata 参数处填写您的密文文件的内容，执行完这个命令后，将生成一个明文文件在锁中。如下代码所示：

```
memcpy(cmddata, c_exefile, exesize);
errcode = DIC_Command(hic, PLAINTEXT_FILE,
cmddata);
```

（注：c\_exefile 是存放密文文件内容的缓冲区，exesize 是密文数据的大小，hic 是已经打开的并没有经过超级密码验证的 ROCKEY6 SMART 句柄。）

## 2. 部分用户相同

在进行远程升级标志和升级密钥的初始设置时，将部分用户的远程升级标志和升级密码相同并且保证远程升级标志的最高位为 0，这样，开发商的可以对某一批用户进行升级，使这一批用户的远程升级文件相同。

采用这种方案时，您需要再所有的加密锁出厂前将相同的一类锁统一进行一次批量设置远程升级初始标志和密码，为了说明方便，我们假设初始标志和密码全为“1”。



图五 批量烧制时设置初始标志和密码

设置完初始远程升级标志和密码后，您就可以将这些锁发送给您的用户了。当需要升级时，您需要通过以下的步骤进行：

通过 IDE 远程升级管理第三步，输入一个新的标志（这里假设为 22222222），和当前的密码，点

击“获取升级密码”按钮，获得新的密码，您可以将这个新的标志和密码保存起来。



图六

转到 IDE 真实设备主界面，使用生成密文文件的功能，生成密文文件，这时，生成的密文文件就和远程升级信息相关了，现在，您就可以把新的标志和密码以及刚才生成的密文文件一起发送给你的用户了。

用户拿到这些升级数据之后，就可以使用客户端远程升级工具进行升级了，首先在远程升级标志一栏填入新的标志和密码，然后点击开始升级进行远程标志升级。



图七

进行完远程升级后，再在远程文件升级一栏中加入待升级的文件，再次点击开始升级生成明文文件。这样整个升级过程就完成了。

## 3. 所有用户全不相同

在进行远程升级标志和远程升级密钥设置时，可以将远程升级标志的最高位设置为 1，即远程升级标志大于 0x80000000，在升级时生成远程升级密码是

和用户加密锁的硬件序列号相关的，这样就保证了所有用户升级文件均不一样，可以看出，所有用户全不相同与部分相同实现方式大致相同，所不同的是远程升级标志最高位为1，另外就是，开发商需要给每一个用户提供唯一的升级文件，由于用户较多，开发商就不能通过工具手动为用户升级了，就需要通过程序来实现。在加密锁出厂前进行初始化时，如果没有其他地方使用远程升级标志，您可以采用第一种方案中的方式通过IDE的批量烧制工具将所有的锁的远程升级标志和密码置为相同即可。

在进行升级时，开发商从用户获得用户的基本信息（如图二所示），软件开发商就可以根据这些信息来进行生成密文操作。

在编写代码之前，您首先需要了解CRYPTOTEXT\_FILE宏，ROCKEY6 SMART是由这个命令来完成生成密文文件操作的。它所需要的命令缓冲区是，前面是一个DICST\_File，后面紧跟着是文件的数据，使用的方法和创建一个文件并写入数据是相同的，只是由CRYPTOTEXT\_FILE宏一次完成，并将生成的密文数据存放到cmddata参数指向的缓冲区中，您可以将这个缓冲区内的数据复制出来，并保存为一个文件。

进行远程升级标志获取操作：

```
// 设置新的升级标志为 0x02
DIC_Set(cmddata, UPGRADE_REMOTE_TAG,
BY_VALUE, 0x02, NULL);

// 填入实际的用户加密锁硬件序列号（这里假设为 0x7A85728C）
DIC_Set(cmddata, UPGRADE_HARD_SERIAL,
BY_VALUE, 0x7A85728C, NULL);

memset(buffer, 0x02, 8); // 设置新的密码全为 0x22
DIC_Set(cmddata, UPGRADE_REMOTE_PASS,
BY_ARRAY, 0, buffer);

errcode = DIC_Command(hic,
GET_UPGRADE_REMOTE_PASS, cmddata);
```

```
// 把新获得的远程升级密码放到 RemotePass 中
memcpy(RemotePass, cmddata, 8);
```

```
DIC_Set(cmddata, FILL, 512, 0, NULL); //
清空
```

```
DIC_Set(cmddata, FILE_ID, BY_VALUE, 0x0001,
NULL); // 文件 ID 0001
```

```
DIC_Set(cmddata, FILE_CLASS, BY_VALUE, 0xff,
NULL); // 文件类别
```

// 文件属性

```
DIC_Set(cmddata, FILE_ATTRIBUTE, BY_VALUE,
FILEATTR_EXEC, NULL);
```

// 文件大小

```
DIC_Set(cmddata, FILE_SIZE, BY_VALUE,
READDATA_SIZE, NULL);
```

// 可执行文件名 1000

```
DIC_Set(cmddata, FILE_NAME, BY_ARRAY, 0,
"1000");
```

```
DIC_Set(cmddata, FILE_DATA, BY_ARRAY |
READDATA_SIZE, 0, (char*)g_progReadData); //
文件内容
```

// 生成明文文件操作

```
errcode = DIC_Command(hic, CRYPTOTEXT_FILE,
cmddata);
```

```
exesize = DIC_Get(cmddata, FILE_DATA,
BY_ARRAY, c_exeFile);
```

（注：cmddata、buffer 是个足够大的缓冲区，RemotePass 是个 8 字节的数组，c\_exeFile 用于存放返回的密文，hic 是已经打开并且经过超级密码验证的锁的句柄，而 READDATA\_SIZE 是明文数据的大小。）

开发商就可以在用户端的程序中使用如下代码段：

```
// 设置升级标志为开发商生成的新的远程升级标志 (0x02)
```

```
DIC_Set(cmddata, REMOTE_TAG, BY_VALUE, 0x02,
```

NULL);

// 为了简便说明, 这里将新的密码全部设置为

0x02

```
memset(RemotePass, 0x22, 8);
```

```
DIC_Set(cmddata, REMOTE_PASS, BY_ARRAY, 0,
```

```
RemotePass);
```

// 进行验证远程升级标志和远程升级密码操作。

```
iRet=DIC_Command(hic, CHECK_REMOTE_INFO, &rInfo);
```

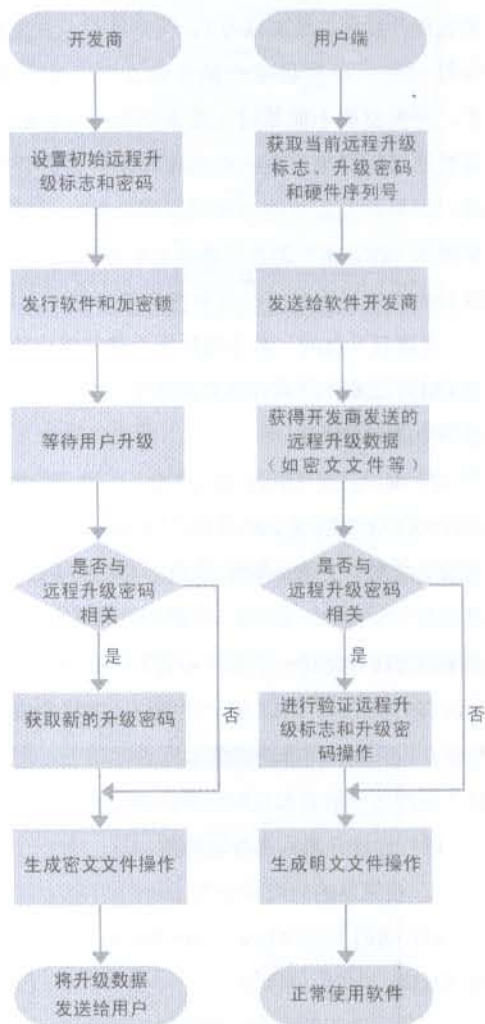
```
memcpy(cmddata, c_exefile, exesize);
```

```
errcode = DIC_Command(hic, PLAINTEXT_FILE, cmddata);
```

(注: `cmddata` 是个足够大的缓冲区, `RemotePass` 是8字节的数组, `c_exefile` 是存放密文文件内容的缓冲区, `exesize` 是密文数据的大小, `hic` 是已经打开的 ROCKY6 SMART 句柄)

再以上三个方案中, 需要注意的是, 如果您采用的是第二种和第三种方式, 那么, 安全文件传输和远程标志升级是同时进行的, 首先软件开发商根据用户端加密锁的信息通过自己的锁设置新的标志和新的密码, 在获取新的密码后, 由于新的升级密码信息已经保存, 再进行生成密文操作, 这时, 将这个新的密码和标志以及密文文件, 一起发送给你的用户, 那么用户再使用新的标志和密码验证通过后就可以进行生成明文的操作了。开发商的获取新的密码和生成密文操作是一步进行的, 中间不能有拔锁或重打开等操作, 同样客户的验证新的标志和密码的操作也是一步进行

的。根据前面的描述, 整个远程升级过程如下图所示:



图八

