RF-Applet Specification

Version 5.0

文档版本号：**1.0.5**

目录

01. 引言

01.1. 概要

RF-UIM 是用于手机的用户身份识别卡。RF-UIM 创新性地将无线射频（RF）模块集合到 UIM 卡内，从而彻底改变传统 UIM 卡只能和手机进行通讯的缺点，使 RF-UIM 可以随时和周围的设备进行交互，并且不影响 UIM 卡的原有功能。RF-UIM 可以广范应用于涉及电子支付、身份识别和信息传递等的各种应用领域，从而促进电信和银行、交通等领域的进一步融合。

RF-UIM 在测试卡或零售 DIY 卡中支持一卡多号（正式运营商卡不支持）,并且支持以 OTA 方式下载电话卡或删除电话卡。

01.2. 定义

UIM	用户识别模块（User Identity Model）。
RF	无线电射频技术（Radio Frequency）
RF-UIM	带 2.4G 无线电射频的手机 UIM 卡
RF-Pod	与 RF-UIM 配套使用的 Reader
Mifare 卡	NXP 公司的一种非接触式感应 IC 卡
ActiveC	插件形式的卡应用程序的
Applet	嵌入形式的卡应用程序
Apdu	应用协议数据单元
Mac	信息效验码
DES	一种数据加密算法标准(Data Encryption Standard)
RSA	一种非对称加密算法
CRC	循环冗余编码（Cyclic Redundancy Code）
PPS	常用的设备性能指标（Packet Per Second）
STK	STK 应用发展工具（UIM Tool Kit）
MCU	微控制器
OTA	Over the Air

02. 应用相关指令

注：所有指令操作前，需连接 RF 并且选择应用。

02.1. 修改应用状态

Describe：

修改应用状态

Detail:

Field	Len	Value	Note
Class	1	90	
Ins	1	38	
P1	1	00	
P2	1	XX	0x00 // 未使用 0x01 // 已被创建，还未安装好 0x02 // 应用已安装好,可以使用了 0x40 // 被停用 0x80 // 被锁,以后状态将不能被改变
Lc	1	08	
Data	8		长度 8，数据为用 Apdu 指令头用应用密钥生成的 Mac 数据

Return:

SW:

SW	Note
9A07	认证失败
9000	命令操作成功

Notes:

02.2. 修改应用 AID

Describe：

修改应用 AID

Detail:

Field	Len	Value	Note
Class	1	90	
Ins	1	38	
P1	1	01	
P2	1	00	
Lc	1	18	
Data	24		长度 24，数据为需要修改的 AID+用 Apdu 指令头用应用密钥生成的 Mac 数据

Return:

SW:

SW	Note
9A07	认证失败
9000	命令操作成功

Notes:

02.3. 修改应用名称

Describe：

修改应用名称

Detail:

Field	Len	Value	Note
Class	1	90	
Ins	1	38	
P1	1	02	
P2	1	00	
Lc	1	28	
Data	40		长度 40，数据为需要修改的应用名称+用 Apdu 指令头用应用密钥生成的 Mac

			数据

Return:

SW:

SW	Note
9A07	认证失败
9000	命令操作成功

Notes:

应用名称包括中午和英文，格式为 中文名称（长度+编码+内容）+英文名称（长度+编码+内容）Unicode 编码方式。

02.4. 修改应用密钥 1

Describe：

修改应用密钥

Detail:

Field	Len	Value	Note
Class	1	90	
Ins	1	38	
P1	1	03	
P2	1	00	
Lc	1	18	
Data	24		长度 24，数据为需要修改的应用密钥+用 Apdu 指令头用应用密钥生成的 Mac 数据

Return:

SW:

SW	Note
9A07	认证失败
9000	命令操作成功

Notes:

02.5. 修改应用密钥 2

Describe：

修改应用密钥

Detail:

Field	Len	Value	Note
Class	1	90	
Ins	1	38	
P1	1	06	
P2	1	00	
Lc	1	18	
Data	24		长度 24，数据为需要修改的应用密钥(3DES 加密)+用 Apdu 指令头用应用密钥生成的 Mac 数据

Return:

SW:

SW	Note
9A07	认证失败
9000	命令操作成功

Notes:

附录 A.

A.01. 取随机数

Describe：

取随机数

Detail:

	Len	Value	Note
Class	1	90	
Ins	1	E6	
P1	1	00	
P2	1	00	
Lc	1	XX	

Return:

Field	Len	Value	Note
Random	XX		随机数

SW:

SW	Note
9000	执行成功

Notes:

A.02．选择应用

Describe：

选择应用

Detail:

Field	Len	Value	Note
Class	1	A0	
Ins	1	A4	
P1	1	04	
P2	1	F0	
Lc	0	10	
Data	16		如：输入为字符串”Mifare App”，不足 16 位的补零

Return:

无

SW:

SW	Note
9000	命令操作成功

Notes:

XX 不能大于 0x10

A.03．Mac 实现原代码

```
// pRandom --  随机数
// pData    --  原信息指针
// nLen     --  原信息长度
// pKey     --  密钥
// pMac     --  输出 Mac 的指针
// mLen     --  输出 Mac 的长度

void Make_Mac(U1* pRandom,U1 * pData,U4 nLen,U1 * pKey,U1 * pMac,U1 mLen)
{
    U1 aMacBuffer[8];
    U4 nI;

    memcpy(aMacBuffer,pRandom,8);

    while (nLen>=8)
    {
        for (nI=0;nI<8;nI++) aMacBuffer[nI]^=*pData++;
      // 标准 Des 加密函数  参数说明：0：输出参数  1：输入参数  2：输入参数(密钥)
        Des08E(aMacBuffer, aMacBuffer, pKey);                    // Des 加密
        nLen-=8;
    }

    if (nLen==0)
    {
        for (nI=0;nI<8;nI++)          //  再加上  80 00 00 00 00 00 00 00
        {
            if (nI==0) aMacBuffer[nI]^=0x80; else aMacBuffer[nI]^=0x00;
        }
    }
    else
    {
        for (nI=0;nI<8;nI++)
        {
            if (nI<nLen)        //  再加上  80 00 00 ...直到 8 个字节
            {
                aMacBuffer[nI]^=*pData++;
            }
            else
            {
                if (nI==nLen) aMacBuffer[nI]^=0x80; else aMacBuffer[nI]^=0x00;
```

```
                }
            }
        }

    // 标准 TriDes(3DES)加密函数  参数说明：0：输出参数  1：输入参数  2：输入参数(密钥)
        Des08E_T (aMacBuffer, aMacBuffer, pKey);            // 用 TriDes 加密

        memcpy(pMac,aMacBuffer,mLen);
}
```

附录 B. 一些例子

以下指令均为发给卡的：所有指令均为 A0+长度+APDU 指令模式；

1。选择应用
指令：A0 15 A0 A4 04 f0 10 XXXX
数据：长度 为 16 数据为应用 ID 如：Mifare App 不足的补零

2。修改应用状态（使用前必须选择应用）
指令：
A0 0D 90 38 00 01 08 XXXX
数据：长度 8，为用 Apdu 指令头用应用密钥生成的 Mac 数据。

3。修改应用 Aid（使用前必须选择应用）
指令：
A0 1D 90 38 01 00 18 XXXX
数据：长度 24，为用 Apdu 指令头+16 位的要修改 Aid 和用应用密钥生成的 Mac。即为：16 位 Aid+8 位 Mac

4。修改应用名称（使用前必须选择应用）
指令：
A0 2D 90 38 02 00 28 XXXX
数据：长度 40，为用 Apdu 指令头+32 位的要修改 Stk 名称和用应用密钥生成的 Mac 数据。即为：32 位名称+8 位 Mac

4。修改应用密钥（使用前必须选择应用）
指令：
A0 1D 90 38 03 00 18 XXXX
数据：长度 24，为用 Apdu 指令头+16 位的要修改密钥和用应用密钥生成的 Mac 数据。即为：16 位密钥+8 位 Mac

注：

做 Mac 密钥生成方式：
1.有卡商提供的应用密钥。
2.取得 CardID。
连接 RF 后，用发给 Pos 指令：
80 05 90 B0 05 00 00
返回值为 64 字节数据，最后 8 位即为 CardID。

3.用相关算法算出对称密钥。
unsigned char aKey[16],aKey1[16],CardID[16];
aKey 为为卡商提供的应用密钥。

CardID 前八字节为卡 ID。

算法如下：

```
for (int nI = 0; nI < 8; nI++)
{
    CardID[nI+8]=~CardID[nI];
}
Des08E_T(aKey1,CardID,aKey);          //TriDes 加密(3DES)
Des08D_T(aKey1+8,CardID+8,aKey);      //TriDes 解密(3DES)
```

aKey1 即为最终密钥。

Des 算法为密钥长度为 16 的 ECB 算法。

修改应用密钥流程如下：

1.选择应用。

2.取随机数。

3.做 Mac。

参看 Mac 实现源码

pRandom:刚取到的随机数、

pData：90 38 03 00 18 00 00 00 00 00 00 00 01 01 00 00 00 00 00 00 00

nLen：16+5

pKey：参看上面 做 Mac 密钥生成方式

pMac：最后生成的 Mac。

pMac：Mac 长度。

调用 Make_Mac(U1* pRandom,U1 * pData,U4 nLen,U1 * pKey,U1 * pMac,U1 mLen)生成 Mac。

2.发送 4。修改应用密钥指令。指令是发向卡的。

例子：、

A0 ID 90 38 03 00 18 00 00 00 00 00 00 00 01 01 00 00 00 00 00 00 00 CD 34 8D FF C2 E4 8D F0

向卡发送：A0 ID

Apdu：90 38 03 00 18

Data 前 16 字节密钥+8 位 Mac：00 00 00 00 00 00 00 01 01 00 00 00 00 00 00 00 CD 34 8D FF C2 E4 8D F0

返回 9000 即为成功。

Mac 算法校验

pRandom: 01 01 01 01 01 01 01 01

pData：01 01 01 01 01 01 01 01 01 01 01

nLen：10

pKey：01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

pMac：A8 84 5C 2A 20 23 BB 12

pMac：8

调用 Make_Mac(U1* pRandom,U1 * pData,U4 nLen,U1 * pKey,U1 * pMac,U1 mLen)生成 Mac。

附录 C. 编制历史

版本号	更新时间	修改人	主要内容或重大修改
V1.0.0	2008-01-24	simon	编制规范
V1.0.1	2008-02-26	simon	修改
V1.0.2	2008-05-06	chenjie	修改
V1.0.3	2008-07-16	chenjie	修改
V1.0.4	2009-05-06	chejie	修改
V1.0.5	2009-12-09	huhq	修改