目　　录

第一章 Java 智能卡的设计概要

1.1 项目背景与意义

本项目是 Java 智能卡的设计，Java 智能卡是在智能卡硬件系统的基础之上在卡片内通过软件构建的一个支持 Java 程序下载、安装和运行的软硬件系统。Java 智能卡充分利用了 Java 的平台无关性，使得 Java 技术“一次编写，随处运行”的思想在智能卡上得以实现。

Java 智能卡具有其独特的优点：

1. 跨平台应用。
   Java 编译器产生不依赖于平台的字节码，字节码在虚拟机中运行。它的代码独立于平台，可移植性好，可以跨平台运行。基于 Java 语言开发的智能卡应用程序可以运行在所有的 Java 智能卡上。
2. 安全性好。
   Java 智能卡继承了 Java 语言的安全特性，包括原子事务、应用防火墙等等，用来隔离 Applet 之间的非法访问，Java 语言的“解释性”执行的特点，使得它可以在执行时对代码进行彻底的检查，以防篡改、病毒和其它威胁，其安全程度远远超过了传统的预编译代码。
3. 开发效率高。
   Java 语言的面向对象特性使得编程极为灵活，可利用通用的 Java 开发工具进行应用的开发，开发人员通过大量的通用 API 和密码算法 API，使得编程应用变得简单。
4. 标准兼容。
   JAVA 卡技术以 ISO-7816 标准为基础，因此可以兼容按 ISO-7816 开发的所有智能卡系统和应用系统。Applet 不仅能在 JAVA 智能卡上相互执行，而且也能被现有的 IC 卡或智能卡读写设备所接受，因此 JAVA 智能卡具有很好的标准兼容性。
5. 支持一卡多用和重用。
   Java 智能卡允许多个应用程序安全地运行在同一张卡上，而且可以在不更换卡片的情况下，根据需要动态添加或删除卡中的应用。

Java 智能卡在全球各个领域都有着非常广泛的应用，比如 SIM 卡，银行卡，交通卡，身份识别，信息安全等等。通过这个 Java 智能卡开发项目，我们可以更好的学习 Java 智能卡的相关知识，通过自己动手编写代码，更加深刻的理解 Java 智能卡的各种优点以及对我们生活的影响。

1.2 Java 智能卡发展现状

Java 智能卡在全球智能卡市场的发展都呈现大幅增长势头。在通信方面，由于用户与运营商对数字网络技术的要求和信任以及安全意识的增强，促使 SIM 卡市场激增，特别是在中国、印度和拉美州等地区，而 SIM 卡市场的发展趋势则是以基于 Java 智能卡平台的高端 SIM 卡为主导。在金融方面，Java 智能卡的电子钱包功能非常适合人们日常的圈存消费，而全球的银行也保持增长势头，在中国，随着 EMV 迁移计划的开展，智能卡在银行领域必将得到大规模的应用。在交通领域，发卡量每年以两位数字增长，英国伦敦的 Oyster 卡和法国巴黎的 Navigo 卡都是比较成功的交通卡项目，但由于采用电子票务的非接触式无线射频技术标准尚不统一，实际上许多项目受到各自特定应用环境的限制。至于 Java 智能卡在身份识别方面的应用，虽处在起步阶段，但是发展速度极快，欧洲和亚洲的部分国家比如比利时和意大利，中国和泰国等已经推出身份证卡和医疗卡的项目。在生物识别市场，随着虹膜、指纹识别、掌纹扫描、声音分析等技术的发展，Java 智能卡由于其体积小、安全性高、存储容量大灯特点，成为携带个人生物信息的最佳载体。

Java 智能卡在许多领域都有着非常广泛的应用，市场也是涉及方方面面，但同时也存在着一些技术上的不足，因此具有非常诱人的发展前景，对于 Java 智能卡的开发学习项目也逐渐被越来越多人提上日程。

1.3 项目设计内容

本项目是基于 Java 智能卡的应用 Applet 开发，实现一个可以安装在 Java 卡片上的应用，该应用是实现电子钱包的功能。

1.4 项目设计的人员分工与进度

姓名	学号	分工
蔡荣裕	15352008	编写实验代码，完成实验报告
蔡一帆	15352011	编写实验代码，完成实验报告

第二章 电子钱包应用的设计过程

2.1 代码架构

部分功能代码已经给出，需要我们在此部分代码上进行代码设计，实现完整的电子钱包的功能。主要代码功能如下表-1：

文件	功能
BinaryFile.java*	读写二进制文件
condef.java*	定义指令常数以及部分错误常数
EPFile.java*	圈存初始化、圈存和消费初始化、消费功能，以及余额获取
KeyFile.java*	密钥相关操作
Papdu.java	APDU 的具体实现
PenCipher.java	COS 中安全管理部分（DES 的实现）
purse.java	处理 APDU 指令，并做出相应的响应
Randgenerator.java	产生随机数

表-1 代码功能表

**为已给出的代码*

实验中我们需要完整实现 Papdu、PenCipher、purse、Randgenerator，四部分代码。其中给出给出的 EPFIle 中依赖 PenChipher 和 Randgenerator 中函数主要是 DES 获取过程密钥和随机数部分。

2.2 APDU 辅助类的实现

命令头必写部分				命令选择部分		
CLA	INS	P1	P2	LC	Data	LE

APDU 命令：

- CLA：指令的类
- INS：指令编码
- P1、P2：参数 1、参数 2

- LC：数据段长度
- Data：发送的数据
- LE：所期待的响应字节数

APDU 中除了包含 CLA、INS、P1、P2、LC、LE 必要部分，还包含了数据段部分，所以对于 APDU 的辅助类我们需要定义对应部分的变量。同时对于 APDU 指令中我们需要判断这个指令是否包含数据，此时在此辅助类中添加判断函数。

```
public class Papdu {
    public byte cla, ins, p1, p2;
    public short lc, le;
    public byte[] pdata;
    /**
     * 构造函数
     */
    public Papdu(){
        pdata = JCSystem.makeTransientByteArray((short)255, JCSystem.CLEAR_ON_DESELECT);
    }
    /**
     * 判断APDU命令是否包含数据
     * @return 是否包含数据
     */
    public boolean APDUContainData(){
        switch(ins){
        case condef.INS_CREATE_FILE:
        case condef.INS_WRITE_KEY:
        case condef.INS_WRITE_BIN:
        case condef.INS_INIT_TRANS:
        case condef.INS_LOAD:
        case condef.INS_PURCHASE:
        case condef.INS_GET_SESPK:
        case condef.INS_GET_MAC:
            return true;
        }
        return false;
    }
}
```

代码-1　Papdu 实现代码

2.3 处理 APDU 指令

2.3.1 获取 APU 缓冲区内容

从 APDU 缓冲区数组引用并将之赋值给辅助类，使用 apdu.getBuffer()取得 APDU 对象的通信缓冲区数组，然后将这部分数据存于 APDU 辅助类中，此处需要而外判断是否含有数据，如有数据需要将一起赋值。

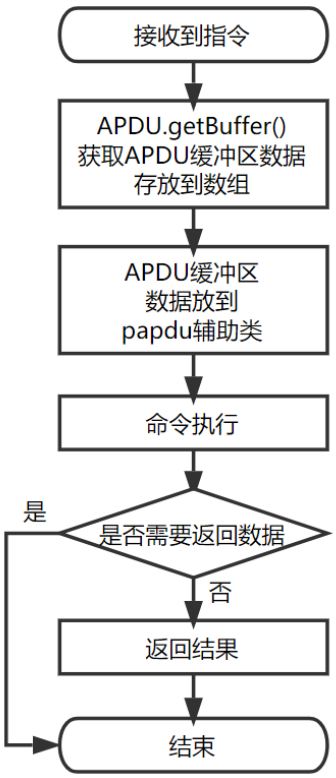


图-1 命令执行流程

函数说明：

- APDU.getBuffer() 返回 APDU 缓冲区数组
- APDU. setIncomingAndReceive()基本接收命令，设置通信传 输模式为终端到卡片
- Util.arrayCopyNonAtomic()对源数据指定部分进行拷贝
- APDU.setOutgoingAndSend()将 apdu 置为卡片到终端模式并设置返回数据

```
//取APDU缓冲区数组引用并将之赋给新建数组
byte[] buf= apdu.getBuffer();
//取APDU缓冲区中数据放到变量papdu
short lc = apdu.setIncomingAndReceive(); //接收数据
papdu.cla = buf[ISO7816.OFFSET_CLA];
papdu.ins = buf[ISO7816.OFFSET_INS];
papdu.p1 = buf[ISO7816.OFFSET_P1];
papdu.p2 = buf[ISO7816.OFFSET_P2];
Util.arrayCopyNonAtomic(buf, ISO7816.OFFSET_CDATA, papdu.pdata, (short)0, lc);
//判断命令APDU是否包含数据段
boolean hasData = (papdu.APDUContainData());
if(hasData){//有数据获取数据长度 le赋值
    papdu.lc = buf[ISO7816.OFFSET_LC];
    papdu.le = buf[ISO7816.OFFSET_CDATA+lc];
}
else{//无数据不需要lc 缓冲区此时为le
    papdu.le = buf[ISO7816.OFFSET_LC];
    papdu.lc = 0;
}
boolean flag = handleEvent();//是否成功处理了命令
if(flag && papdu.le!=0){//需要返回数据
    Util.arrayCopyNonAtomic(papdu.pdata, (short)0, buf, (short)0, (short)papdu.le);
    apdu.setOutgoingAndSend((short)0, (short)papdu.le);
}
```

代码-2 APDU 命令处理执行

2.3.2 处理执行命令

COS 中需要我们事先的有 8 个基础命令同时还用两个命令用于计算过程密钥和 MAC 与 TAC。

命令	常数名	作用
0xE0	INS_CREATE_FILE	文件建立
0xD4	INS_WRITE_KEY	写入密钥
0xD6	INS_WRITE_BIN	写入二进制
0x50	INS_INIT_TRANS	初始化圈存和初始化消费
0x52	INS_LOAD	圈存
0x54	INS_PURCHASE	消费
0x5C	INS_GET_BALANCE	查询余额
0xB0	INS_READ_BIN	读取二进制
0x60	INS_GET_SESPK	计算过程密钥
0x62	INS_GET_MAC	计算 MAC 或 TAC

我们需要对着 10 个命令分别进行处理，注意的一点是初始化圈存和初始化消费是两个部分需要分开处理，如果命令处理成功返回 true，失败返回 false，对于不存在的命令直接抛出异常。

```
private boolean handleEvent(){
    switch(papdu.ins){
        case condef.INS_CREATE_FILE:      return create_file();
        case condef.INS_WRITE_KEY:        return write_key();
        case condef.INS_WRITE_BIN:        return write_bin();
        case condef.INS_INIT_TRANS:
            if(papdu.p1 == (byte)0x00)    return init_load();
            if(papdu.p1 == (byte)0x01)    return init_purchase();
            ISOException.throwIt(ISO7816.SW_WRONG_P1P2); //不存在的操作
        case condef.INS_LOAD:            return load();
        case condef.INS_PURCHASE:        return purchase();
        case condef.INS_GET_BALANCE:     return get_balance();
        case condef.INS_READ_BIN:        return read_bin();
        case condef.INS_GET_SESPK:       return get_sespk();
        case condef.INS_GET_MAC:         return get_mac();
    }
    ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
    return false;
}
```

代码-3 命令的执行

2.4 文件系统

文件系统主要是密钥文件、持卡人基本文件以及应用基本文件。与之相关的是 BinaryFile、EPFile、KeyFile 这三个文件。这三个文件已经给出，我们只需要完善，然后使用即可。

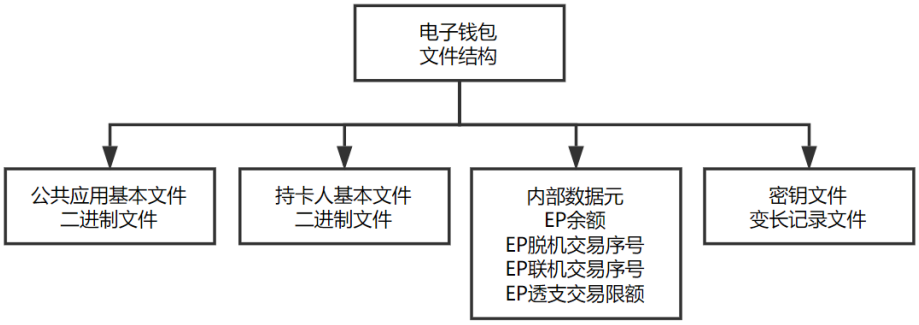


图-2 电子钱包文件结构

公共应用基本文件与持卡人基本文件，主要是保存发卡方信息和持卡人信息。

密钥文件主要存储各种密钥，TAC、消费、圈存密钥。

EP 文件为电子钱包文件，存储电子钱包数据。其中 EPFile 需要有 PenChipher 和 Randgenerator 的支持，PenChipher 主要是实现计算过程密钥和 MAC 与 TAC。Randgenerator 为随机数生成。

2.4.1 文件创建

在电子钱包应用中，初始化指令为 create_file 命令。当发送初始化指令后，电子钱包已建立起相应的电子钱包文件，我们需要创建上述 4 个文件，文件创建命令报文如下表

CLA	INS	P1	P2	Lc	Data
80	E0	00	00	文件信息长度	文件控制信息

对于创建不同文件的 P2 值如下表

密钥文件	应用基本文件	持卡人基本信息文件	电子钱包文件
00	16	17	18

Data 部分值如下表

	命令报文数据域					
文件类型	BYTE 1	BYTE 2-3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
持卡人基本文件	39	0037	FF	FF	FF	FF
应用基本文件	38	001e	FF	FF	FF	FF
电子钱包文件	2F	FFFF	FF	FF	FF	FF
密钥文件	3F	FFFF	FF	FF	FF	FF

创建文件时只需要判断 APDU 命令是否正确，同时判断文件是否已存在，如满足命令正确且不存在则创建，其他情况直接抛出异常，提示相应错误。

```
//文件系统
private KeyFile keyfile;      //密钥文件
private BinaryFile cardfile; //应用基本文件
private BinaryFile personfile;     //持卡人基本文件
 private EPFile epfile;            //电子钱包文件
 private boolean create_file() {
   switch(papdu.pdata[0]){
       case condef.KEY_FILE:       return KEY_file(); //创建密钥文件
       case condef.CARD_FILE:      return CARD_file(); //创建应用基本文件
       case condef.PERSON_FILE:    return PERSON_file(); //创建持卡人基本文件
       case condef.EP_FILE:        return EP_file(); //创建电子钱包文件
       default:
           ISOException.throwIt(ISO7816.SW_FUNC_NOT_SUPPORTED);
   }
   return true;
 }
  //创建密钥文件，其他文件不在展示
private boolean KEY_file() {
   if(papdu.cla != (byte)0x80)
       ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);//CLA错误
   if(papdu.p1 != (byte)0x00 || papdu.p2 != (byte)0x00)
       ISOException.throwIt(ISO7816.SW_WRONG_P1P2);//文件类型错误
   if(papdu.lc != (byte)0x07)
       ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);//错误的长度
   if(keyfile != null)
       ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);//文件已存在
   keyfile = new KeyFile();
   return true;
 }
```

代码-4 创建文件系统

2.4.2 读写文件

2.4.2.1 写入密钥 write key

写入密钥部分我们需要需要写入三种密钥：消费密钥、圈存密钥、TAC 密钥，APDU 命令如下表：

CLA	INS	P1	P2	Lc	Data
80	D4	00(增加)/01(修改)	密钥标识	数据域长度(15)	数据

Data 数据如下

DATA					
3e/3f/34	使用权	更改权	密钥版号	算法标识	16 字节密钥

增加或修改密钥的时候同样需要判断参数是否正确，同时还要判断是否存在密钥文件，如不符合则抛出异常，符合则调用 KeyFile 中的创建密钥函数。

```
private boolean write_key(){
    if(papdu.cla != (byte)0x80)
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);//CLA错误
    if(papdu.p1 != (byte)0x00 && papdu.p1 != (byte)0x01)
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);//参数类型错误
    if(papdu.lc != (byte)0x15)
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);//错误的长度
    if(keyfile == null)//密钥文件不存在
        ISOException.throwIt(ISO7816. SW_FILE_NOT_FOUND);
    if(papdu.p1 == (byte)0x00 && keyfile.recNum == keyfile.size)//密钥已满
        ISOException.throwIt(ISO7816.SW_FILE_FULL);
    keyfile.addkey(papdu.p2, papdu.lc, papdu.pdata);
    return true;
}
```

代码-5 增加、修改密钥文件

2.4.2.2 写二进制文件 write binary

写二进制文件主要是写入发卡人和持卡人信息，Keyfile 中已经实现 write_bineary，主要就是传入数据储存到数组中，APDU 报文如下：

CLA	INS	P1	P2	Lc	Data
00	D6	文件标识符，持卡人基本文件为 0x17 应用基本文件为 0x16	欲写文件的偏移量	数据域长度	数据

此处和写入密钥一样需要判断指令是否正确，以及文件是否存在。

```
private boolean write_bin(){
    if(papdu.cla != (byte)0x00)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 == (byte)0x17){ //写入持卡人基本文件
        if(personfile == null) //文件不存在
            ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
        else if(papdu.p2 + papdu.lc>personfile.get_size())//错误的长度
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        else
            personfile.write_bineary(papdu.p2, papdu.lc, papdu.pdata);
    }
    else if(papdu.p1 == (byte)0x16){//写入基本应用文件
        if(cardfile == null)//文件不存在
            ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
        else if(papdu.p2 + papdu.lc>cardfile.get_size())//错误的长度
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
       else
            cardfile.write_bineary(papdu.p2, papdu.lc, papdu.pdata);
```

```
    }
    else
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    return true;
}
```

代码-6 写入二进制文件

2.4.2.3 读二进制文件 read binary

读取二进制文件主要是读取发卡人和持卡人信息。KeyFile 中已经实现 read binary，即是把数据从数组中读出来，命令报文如下：

CLA	INS	P1	P2	Le
00	B0	文件标识符,持卡人基本文件为 0x17,应用基本文件为 0x16	欲写文件的偏移量	数据域长度

此处只需要判断指令是否正确以及文件是否存在，代码如下：

```
private boolean read_bin(){
    if(papdu.cla != (byte)0x00)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 == (byte)0x17){//读取持卡人基本文件
       if(personfile == null) //文件不存在
            ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
       else if(papdu.p2 + papdu.le>personfile.get_size())//错误的长度
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
       else
            personfile.read_binary(papdu.p2, papdu.le, papdu.pdata);
    }
    else if(papdu.p1 == (byte)0x16) {//读取基本应用文件
       if(cardfile == null)//文件不存在
            ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
        else if(papdu.p2 + papdu.le>cardfile.get_size())//错误的长度
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        else
       cardfile.read_binary(papdu.p2, papdu.le, papdu.pdata);
    }
    else
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    return true;
}
```

代码-7 读取二进制文件

2.5 电子钱包

2.5.1 电子钱包的圈存

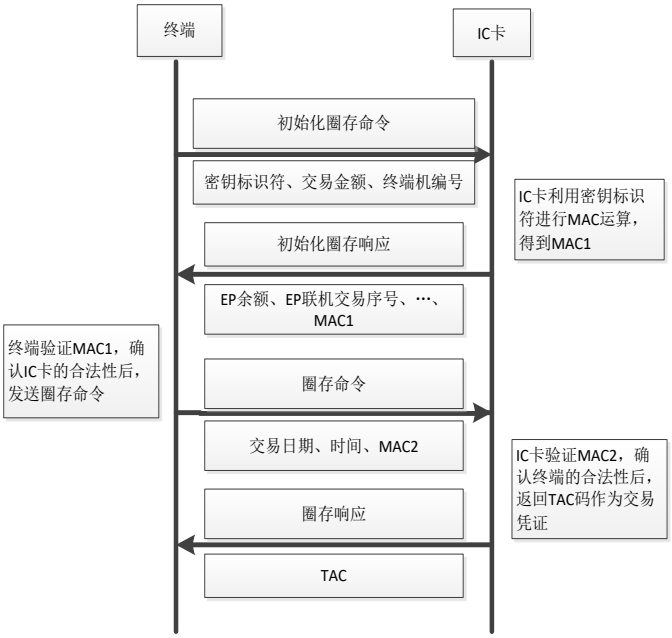


图-3 电子钱包流程图

电子钱包文件为 EPFile，需要有 PenChipher 和 Randgenerator 的支持，PenChipher 主要是实现计算过程密钥和 MAC 与 TAC。Randgenerator 为随机数生成。

此处我们需要完善这两个文件。

2.5.1.1 计算过程密钥

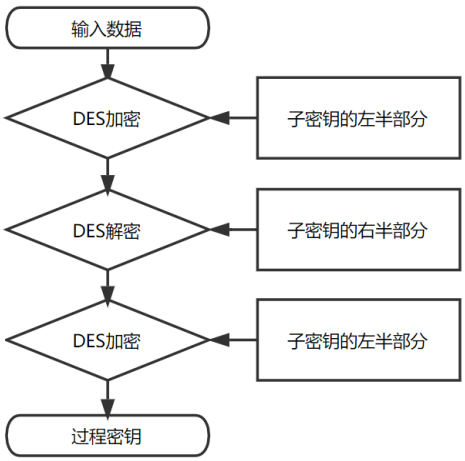


图-4 过程密钥的生成过程

过程密钥通过 3 次 DES 算法生成，我们需要调用已有的 DES 加密算法完成。

DES 加密可以使用 Cipher 中的函数进行，步骤如下：

- 使用 DESKey 接口中的 setKey()函数设置密钥值。参数为密钥和偏移
- 使用 Cipher 类中的 init()函数来设置加密对象实例，参数为初始密钥和模式，模式分加解密模式（Cipher.MODE_ENCRYPT 或 Cipher.MODE_DECRYPT）
- 使用 Cipher 类中的 doFinal()函数来完成运算，参数为加密数据、加密数据偏移量、数据长度、加密结果、

结果偏移量。

```
public PenCipher(){
    desEngine = Cipher.getInstance(Cipher.ALG_DES_CBC_NOPAD, false);//获得加密实例
    deskey = KeyBuilder.buildKey(KeyBuilder.TYPE_DES, KeyBuilder.LENGTH_DES, false);//生成DES密钥实例
}
public final void des(byte[] key, short kOff, byte[] data, short dOff, short dLen, byte[] res, short rOff,
byte mode){
    ((DESKey)deskey).setKey(key, kOff);   //设置DES密钥
    desEngine.init(deskey, mode);//初始化密钥及加密模式
    desEngine.doFinal(data, dOff, dLen, res, rOff);//加密
}
```

代码-8 DES 算法

我们根据图-4 算法流程调用三次 DES 算法实现过程密钥的成生。

```
public final void gen_SESPK(byte[] key, byte[]data, short dOff, short dLen, byte[] res, short rOff){
    byte[] tmp1 = JCSystem.makeTransientByteArray(dLen, JCSystem.CLEAR_ON_DESELECT);
    byte[] tmp2 = JCSystem.makeTransientByteArray(dLen, JCSystem.CLEAR_ON_DESELECT);
    des(key,(short)0,data,dOff,dLen,tmp1,rOff,Cipher.MODE_ENCRYPT);
    des(key,(short)8,tmp1,dOff,dLen,tmp2,rOff,Cipher.MODE_DECRYPT);
    des(key,(short)0,tmp2,dOff,dLen,res,rOff,Cipher.MODE_ENCRYPT);
}
```

代码-9 过程密钥生成

COS 中有测试过程密钥的指令，报文如下：

CLA	INS	P1	P2	Lc	Data		Le
00	60	00	00	0x18(24)	密钥,16 字节	输入的数据,8 字节	08

其中输入的数据再圈存时为：伪随机数||电子钱包联机交易序号||8000；此时密钥为圈存密钥。

在消费时输入的数据为：伪随机数||电子钱包脱机交易序号||终端交易序号的最右两个字节；此时密钥为消费密钥。

```
 private boolean get_sespk(){
PenCipher pencipher = new PenCipher();
byte[] key = JCSystem.makeTransientByteArray((short)16, JCSystem.CLEAR_ON_DESELECT);
byte[] data = JCSystem.makeTransientByteArray((short)8, JCSystem.CLEAR_ON_DESELECT);
Util.arrayCopyNonAtomic(papdu.pdata, (short)0, key, (short)0, (short)16);
Util.arrayCopyNonAtomic(papdu.pdata, (short)16, data, (short)0, (short)8);
pencipher.gen_SESPK(key, data, (short)0, (short)8, papdu.pdata, (short)0);
return true;
}
```

代码-10　获取过程密钥

2.5.1.2 计算 MAC 与 TAC

MAC 和 TAC 需使用过程密钥来计算，计算方法相同只是输入数据不同，步骤如下：

- 将初始值设定为 8 个字节长的 16 进制数 0x 0000000000000000
- 在输入的数据尾部填入 0x80，检查数据的字节数是否为 8 的倍数。如果不足，则在尾部添加 0x00，直至满足 8 的倍数为止。
- 将这些数据分割为 8 字节长的数据块组。
- 之后算法流程如下图-5

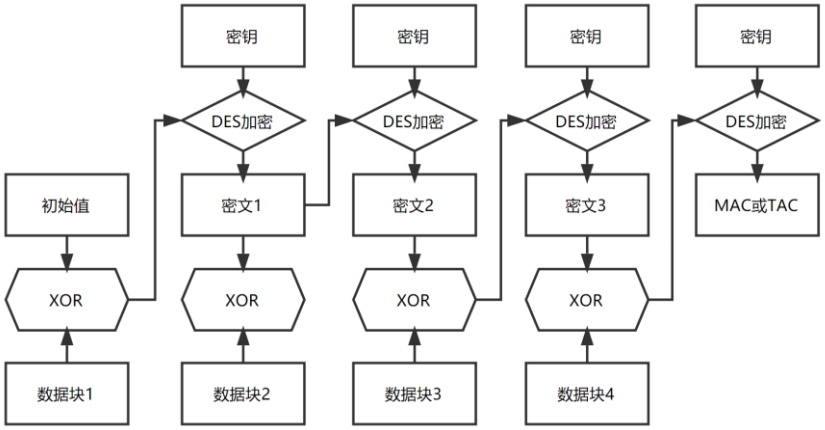


图-5 MAC 或 TAC 计算过程

计算 MAC 代码如下

```
public final void xorblock8(byte[] data1, byte[] data2, short d2off){ //异或模块
     for(short i = 0; i < 8; i++)
          data1[i]^=data2[i+d2off];
}
public final void gmac4(byte[] key, byte[] data, short len, byte[]  res){
     byte[] tmp1 = JCSystem.makeTransientByteArray((short)8, JCSystem.CLEAR_ON_DESELECT);
     byte[] tmp2 = JCSystem.makeTransientByteArray((short)8, JCSystem.CLEAR_ON_DESELECT);
     Util.arrayFillNonAtomic(tmp1, (short)0, (short)8, (byte)0x00);//初值
     data[len++] = (byte)0x80;
     if(len%8!=0){//填充至8的倍数
          Util.arrayFillNonAtomic(data, len, (short)(8-len%8), (byte)0x00);
          len+=(8-len%8);
     }
     for(short off = 0; off < len; off+=8){//异或、des加密
          xorblock8(tmp1, data, off);
          des(key,(short)0,tmp1, (short)0, (short)8, tmp2, (short)0, Cipher.MODE_ENCRYPT);
          Util.arrayCopyNonAtomic(tmp2, (short)0, tmp1, (short)0, (short)8);
     }
     Util.arrayCopyNonAtomic(tmp2, (short)0, res, (short)0, (short)4);//MAC、TAC长度为4byte
}
```

代码-11 计算 MAC 或 TAC

COS 中还附带有计算 MAC 的函数，报文为

CLA	INS	P1	P2	Lc	Data		Le
00	62	00	00	0x09 ~ 8+n	密钥,8 字节	输入的数据, 1~n 个字节	04

代码如下

```
private boolean get_mac(){
    PenCipher pencipher = new PenCipher();
    byte[] key = JCSystem.makeTransientByteArray((short)8, JCSystem.CLEAR_ON_DESELECT);
    byte[] data = JCSystem.makeTransientByteArray((short)32, JCSystem.CLEAR_ON_DESELECT);
    Util.arrayCopyNonAtomic(papdu.pdata, (short)0, key, (short)0, (short)8);
    Util.arrayCopyNonAtomic(papdu.pdata, (short)8, data, (short)0, (short)(papdu.lc -8));
    pencipher.gmac4(key, data, (short)(papdu.lc -8), papdu.pdata);
    return true;
}
```

代码-12 计算 MAC

2.5.1.3 圈存的初始化

圈存初始化通过发送初始 APDU 命令，具体报文如下

CLA	INS	P1	P2	Lc	Data			Le
80	50	00	02	0B	密钥索引号,1 字节	交易金额,4 字节	终端机编号,6 字节	10

收到圈存初始化命令后，先判断指令是否正确，跟据密钥标识符找出相应的圈存密钥，如果找不到，抛出异常，找到密钥后，IC 卡生成随机数，利用所查找到的密钥产生过程密钥。然后根据过程密钥生成 MAC1，之后将相应的数据返回，返回报文如下：

说明	EP 余额	EP 联机交易序列号	密钥版本号 DPK	算法标识 DPK	伪随机数(IC 卡)	MAC1
长度/字节	4	2	1	1	4	4

此处返回的信息中重要的是 MAC1。终端验证 MAC1，验证成功后终端向 IC 卡发送圈存命令。

```
private boolean init_load() {
    short num,rc;
    if(papdu.cla != (byte)0x80)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 != (byte)0x00 && papdu.p2 != (byte)0x02)//参数错误
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    if(papdu.lc != (short)0x0B)//长度错误
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    if(epfile == null)//文件不存在
        ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
    num = keyfile.findkey(papdu.pdata[0]);//寻找密钥的记录号
    if(num == 0x00)//找不到相应密钥
        ISOException.throwIt(ISO7816.SW_RECORD_NOT_FOUND);
    rc = epfile.init4load(num, papdu.pdata);
    if(rc == 2)//圈存超过最大值
        ISOException.throwIt((condef.SW_LOAD_FULL));
    papdu.le = (short)0x10;
    return true;
}
```

代码-13 圈存初始化

2.5.1.4 圈存命令

圈存命令如下：

CLA	INS	P1	P2	Lc	Data			Le
80	52	00	02	0B	交易日期(主机) 4 个字节 0000 年 00 月 00 日	交易时间(主机) 3 个字节 00 时 00 分 00 秒	MAC2 4 个字节	04

IC 卡收到圈存命令后，利用过程密钥生成 MAC2。与圈存命令传送的 MAC2 进行比较，如果相同，则 MAC2 有效。IC 卡将电子钱包联机交易序号加 1，并且把交易金额加在电子钱包的余额上。IC 卡生成 TAC 码并将 TAC 码返回给终端。

```
private boolean load() {
    short rc;
    if(papdu.cla != (byte)0x80)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 != (byte)0x00 && papdu.p2 != (byte)0x00) //参数错误
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    if(epfile == null) //文件不存在
        ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
    if(papdu.lc != (short)0x0B)//长度错误
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    rc = epfile.load(papdu.pdata);
    if(rc == 1)//MAC校验错误
        ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
    else if(rc == 2)//圈存超额
        ISOException.throwIt(condef.SW_LOAD_FULL);
    else if(rc == 3)//密钥未找到
        ISOException.throwIt(ISO7816.SW_RECORD_NOT_FOUND);
    papdu.le = (short)4;
    return true;
}
```

代码-14 圈存

2.5.2 电子钱包的消费

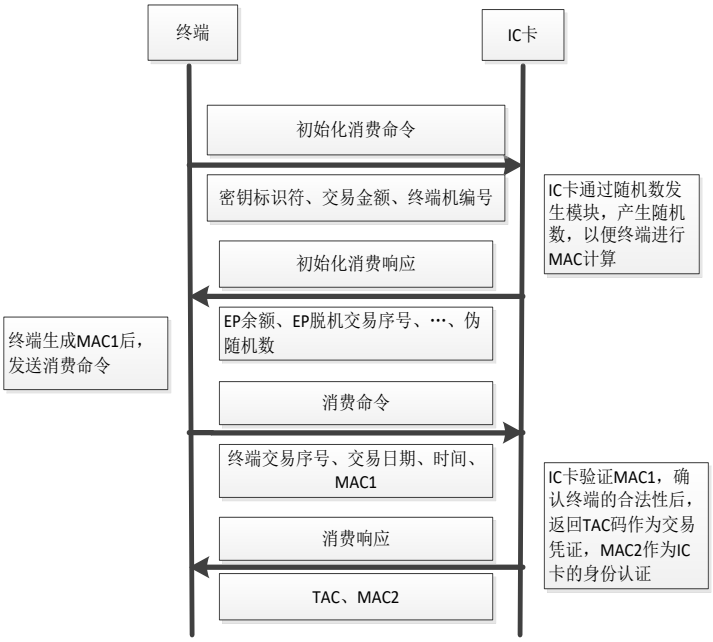


图-4 消费流程

2.5.2.1 消费初始化

消费时，终端先给卡片发送消费初始化命令，命令报文如下

CLA	INS	P1	P2	Lc	Data			Le
80	50	01	02	0B	密钥索引号，1 字节	交易金额，4 字节	终端机编号，6 字节	0F

首先判断命令是否正确，然后根据密钥标识符，在密钥文件中查找该密钥标识符对应的消费密钥，找到密钥后生成随机数，然后检查余额是否足够，余额不足则抛出异常，余额充足将相应的数据返回，返回报文如下：

说明	EP 余额	EP 脱机交易序列号	透支限额	密钥版本号 DPK	算法标识 DPK	伪随机数(IC 卡)
长度/字节	4	2	3	1	1	4

终端将命令响应数据传送给主机，主机利用消费主密钥产生消费子密钥，并生成 MAC1，之后发送消费命令。

```
private boolean init_purchase(){
    short num,rc;
    if(papdu.cla != (byte)0x80)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 != (byte)0x01 && papdu.p2 != (byte)0x02)//参数错误
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    if(papdu.lc != (short)0x0B)//长度错误
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    if(epfile == null)//文件不存在
        ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
    num = keyfile.findkey(papdu.pdata[0]);//寻找密钥的记录号
    if(num == 0x00)//找不到相应密钥
        ISOException.throwIt(ISO7816.SW_RECORD_NOT_FOUND);
    rc = epfile.init4purchase(num, papdu.pdata);
    if(rc == 2)//余额不足
        ISOException.throwIt((condef.SW_BALANCE_NOT_ENOUGH));
    papdu.le = (short)15;
    return true;
}
```

代码-15 消费初始化

2.5.2.2 消费

主机利用消费主密钥产生消费子密钥，并生成 MAC1，并发送消费报文如下：

CLA	INS	P1	P2	Lc	Data				Le
80	54	01	00	0F	终端交易序号 4 个字节	交易日期(主机) 0000 年 00 月 00 日 4 个字节	交易时间(主机) 00 时 00 分 00 秒 3 个字节	MAC1 4 个字节	08

收到消费命令后，利用所查找到的密钥产生过程密钥，然后通过过程密钥生成 MAC1 与消费命令传送的 MAC1 进行比较，如果相同，将电子钱包脱机交易序号加 1，并且把电子钱包的余额减去交易金额。之后利用过程密钥生成 MAC2 和 TAC，并将 MAC2 和 TAC 返回。

说明	交易验证码 TAC	MAC2
长度/字节	4	4

至此消费完成。

```
private boolean purchase(){
    short rc;
    if(papdu.cla != (byte)0x80)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 != (byte)0x01 && papdu.p2 != (byte)0x00)//参数错误
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    if(epfile == null)//文件不存在
        ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
    if(papdu.lc != (short)0x0F)//长度错误
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    rc = epfile.purchase(papdu.pdata);
    if(rc == 1)//MAC校验错误
        ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
    else if(rc == 2)//余额不足
        ISOException.throwIt((condef.SW_BALANCE_NOT_ENOUGH));
    else if(rc == 3)//密钥未找到
        ISOException.throwIt(ISO7816.SW_RECORD_NOT_FOUND);
    papdu.le = (short)8;
    return true;
}
```

代码-16 消费

2.5.3 电子钱包查询余额

余额查询直接发送查询命令，命令报文如下

CLA	INS	P1	P2	Lc	Data	Le
80	5C	00	02	不存在	不存在	04

接收到命令后判断命令是否正确，然后调用原有函数进行返回。

```
private boolean get_balance(){
  short res;
  byte[] balance = JCSystem.makeTransientByteArray((short)4, JCSystem.CLEAR_ON_DESELECT);
    if(papdu.cla != (byte)0x80)//CLA错误
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    if(papdu.p1 != (byte)0x01 && papdu.p2 != (byte)0x02)//参数错误
        ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    if(epfile == null)//文件不存在
        ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);
    res = epfile.get_balance(balance);
    if(res ! = (short)0)//返回电子钱包余额
        Util.arrayCopyNonAtomic(balance, (short)0, papdu.pdata, (short)0, (short)4);
    papdu.le = (short)0x04;
    return true;
}
```

代码-17 查询余额

第三章 电子钱包应用的功能测试

3.1 测试获取缓冲区与缓冲区内容返回

```
#//选择电子钱包文件
/select 1535200801
```

```
Card Manager AID   :  A0000000003000000
Card Manager state :  OP_READY

    Application:  SELECTABLE (--------) 153520080101
    Load File  :      LOADED (--------) A0000000035350   (Security Domain)
     Module    :                       A000000003535041
    Load File  :      LOADED (--------) 1535200801
     Module    :                       153520080101
cm>  /select 1535200801
 => 00 A4 04 00 05 15 35 20 08 01 00                  ......5 ...
 (221064 nsec)
 <= 90 00                                             ..
Status: No Error
```

此时发送选择电子钱包命令，此处成功返回为 90 00

3.2 文件系统建立

	CLA	INS	P1	P2	Lc	DATA					
密钥文件	80	E0	00	00	07	39	0037	FF	FF	FF	FF
基本应用文件	80	E0	00	16	07	38	001e	FF	FF	FF	FF
持卡人文件	80	E0	00	17	07	2F	FFFF	FF	FF	FF	FF
电子钱包文件	80	E0	00	18	07	3F	FFFF	FF	FF	FF	FF

具体命令如下

```
#//建立密钥文件
/send 80e00000073ffffffffffff
#//建立发卡方基本信息文件
/send 80e000160738001effffffff
#//建立持卡人基本信息文件
/send 80e0001707390037ffffffff
#//建立电子钱包文件
/send 80e00018072Fffffffffffff
```

```
cm>  /send 80e00000073ffffffffffff
 => 80 E0 00 00 07 3F FF FF FF FF FF FF               .....?......
 (322122 nsec)
 <= 90 00                                             ..
Status: No Error
cm>  /send 80e000160738001effffffff
 => 80 E0 00 16 07 38 00 1E FF FF FF FF               .....8......
 (315805 nsec)
 <= 90 00                                             ..
Status: No Error
cm>  /send 80e0001707390037ffffffff
 => 80 E0 00 17 07 39 00 37 FF FF FF FF               .....9.7....
 (519105 nsec)
 <= 90 00                                             ..
Status: No Error
cm>  /send 80e00018072Fffffffffffff
 => 80 E0 00 18 07 2F FF FF FF FF FF FF               ...../......
 (803724 nsec)
 <= 90 00                                             ..
Status: No Error
```

建立密钥文件

发卡方信息文件

持卡人信息文件

电子钱包文件

此时我们分别建立，密钥文件、发卡方信息、持卡人信息、电子钱包文件，此时全部范围 90 00 说明文件创建成功。

3.3 测试写入密钥文件

	CLA	INS	P1	P2	Lc	DATA					
						密钥类型	使用权	更改权	密钥版号	算法标识	16 字节密钥
消费密钥	80	D4	01	06	15	3e	F0	F0	01	00	09F4ACB09131420B8FE1B4CC007AC52B
圈存密钥	80	D4	01	06	15	3f	F0	F0	01	00	EB9BC6DCDF74FF4E4B43F2E34A6727B6
TAC 密钥	80	D4	01	06	15	34	F0	F0	90	00	CEB726EDC01B793BC37DC09E2F768534

具体命令如下：

```
#//写入密钥文件
#//增加消费密钥　密钥为 09F4ACB09131420B8FE1B4CC007AC52B
/send 80d40007153ef0f0010009F4ACB09131420B8FE1B4CC007AC52B
#//增加圈存密钥　密钥为 EB9BC6DCDF74FF4E4B43F2E34A6727B6
/send 80d40008153ff0f00100EB9BC6DCDF74FF4E4B43F2E34A6727B6
#//增加 TAC 密钥　密钥为 CEB726EDC01B793BC37DC09E2F768534
/send 80d400061534f0f09000CEB726EDC01B793BC37DC09E2F768534
```

```
cm>  /send 80d40007153ef0f0010009F4ACB09131420B8FE1B4CC007AC52B
 => 80 D4 00 07 15 3E F0 F0 01 00 09 F4 AC B0 91 31    .....>.........1
    42 0B 8F E1 B4 CC 00 7A C5 2B                      B......z.+
 (321726 nsec)                    增加消费密钥
 <= 90 00                                             ..
Status: No Error
cm>  /send 80d40008153ff0f00100EB9BC6DCDF74FF4E4B43F2E34A6727B6
 => 80 D4 00 08 15 3F F0 F0 01 00 EB 9B C6 DC DF 74    .....?.........t
    FF 4E 4B 43 F2 E3 4A 67 27 B6   增加圈存密钥       .NKC..Jg'.
 (330412 nsec)
 <= 90 00                                             ..
Status: No Error
cm>  /send 80d400061534f0f09000CEB726EDC01B793BC37DC09E2F768534
 => 80 D4 00 06 15 34 F0 F0 90 00 CE B7 26 ED C0 1B    .....4......&...
    79 3B C3 7D C0 9E 2F 76 85 34                      y;.}../v.4
 (286593 nsec)          增加TAC密钥
 <= 90 00                                             ..
Status: No Error
```

此时增加密钥之后返回为 90 00 说明密钥文件正常。

3.4 测试读写发卡人持卡人信息文件

3.4.1 写入发卡方信息

```
#//写入发卡方基本信息　（626400223333000103010001200108170000000120010101200112315566）
/send 00D616001E626400223333000103010001200108170000000120010101200112315566
#//读取发卡方基本信息
/send 00B016001E
```

```
cm>  /send 00D616001E62640022333300010301000120010817000000012001010120011231556 6
 => 00 D6 16 00 1E 62 64 00 22 33 33 00 01 03 01 00    .....bd."33.....
    01 20 01 08 17 00 00 00 01 20 01 01 01 20 01 12    . ....... ... ..
    31 55 66                                           1Uf
 (354097 nsec)                    写入发卡人信息
 <= 90 00                                              ..
Status: No Error
cm>  /send 00B016001E
 => 00 B0 16 00 1E                读取发卡人信息        .....
 (274356 nsec)
 <= 62 64 00 22 33 33 00 01 03 01 00 01 20 01 08 17    bd."33...... ...
    00 00 00 01 20 01 01 01 20 01 12 31 55 66 90 00    .... ... ..1Uf..
Status: No Error
```

此处我们写入发卡人信息返回 9000 说明写入正常，之后我们测试读取发卡人信息，我们看到读写出来的信息（62 64 00 22 33 33 00 01 03 01 00 01 20 01 08 17 00 00 00 01 20 01 01 01 20 01 12 31 55 66）与写入的信息一致。说明读写发卡人信息正常。

3.4.2 写入持卡人信息

```
#//写入持卡人基本信息 持卡人信息
（000053414D504C452E434152442E41444631000000001101029812180010110102981218001000000000000000000000000000000000000005）
/send
00D617003700053414D504C452E434152442E41444631000000001101029812180010110102981218001000000000000000000000000000000000000005
#//读取持卡人基本信息
/send 00B0170037
```

```
cm>  /send 00D617003700053414D504C452E434152442E41444631000000001101029812180011
 => 00 D6 17 00 37 00 00 53 41 4D 50 4C 45 2E 43 41    ....7..SAMPLE.CA
    52 44 2E 41 44 46 31 00 00 00 00 11 01 02 98 12    RD.ADF1.........
    18 00 10 11 01 02 98 12 18 00 10 00 00 00 00 00    ................
    00 00 00 00 00 00 00 00 00 00 00 05                ............
 (341465 nsec)
 <= 90 00                         写入持卡人信息        ..
Status: No Error
cm>  /send 00B0170037
 => 00 B0 17 00 37                读取持卡人信息        ....7
 (276329 nsec)
 <= 00 00 53 41 4D 50 4C 45 2E 43 41 52 44 2E 41 44    ..SAMPLE.CARD.AD
    46 31 00 00 00 00 11 01 02 98 12 18 00 10 11 01    F1..............
    02 98 12 18 00 10 00 00 00 00 00 00 00 00 00 00    ................
    00 00 00 00 00 00 05 90 00                         .........
Status: No Error
```

此处写入和读取信息之后都返回 9000 说明命令持卡人文件读写正常，同时我们可以看到读取出来的信息（00 00 53 41 4D 50 4C 45 2E 43 41 52 44 2E 41 44 46 31 00 00 00 00 11 01 02 98 12 18 00 10 11 01 02 98 12 18 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 05）与写入的信息一致。

3.5 测试圈存

3.5.1 初始化圈存

圈存初始报文如下

CLA	INS	P1	P2	Lc	密钥索引	交易金额	终端机编号	Le
80	50	00	02	0B	08	00001000	001122334455	10

具体指令如下：

```
/send 805000020B0800001000001122334455 10
```

```
cm>  /send 805000020B0800001000001122334455 10
 => 80 50 00 02 0B 08 00 00 10 00 00 11 22 33 44 55     .P..........."3DU
    10                                                  .
 (2473 usec)
 <= 00 00 00 00 00 00 01 00 27 55 AE 2D F1 97 CB 4B     ........'U.-...K
    90 00                                               ..
Status: No Error
```

此处返回的信息如下：

EP 余额	EP 联机交易序列号	密钥版本号 DPK	算法标识 DPK	伪随机数(IC 卡)	MAC1
00 00 00 00	00 00	01	00	27 55 AE 2D	F1 97 CB 4B

首先验证 MAC1，我们先需要生成过程密钥，

CLA	INS	P1	P2	Lc	Data 密钥||伪随机数|| EP 联机交易序列号||8000		Le
00	60	00	00	0x18	EB9BC6DCDF74FF4E4B43F2E34A6727B6	2755AE2D00008000	08

命令如下：

```
/send 0060000018EB9BC6DCDF74FF4E4B43F2E34A6727B62755AE2D0000800008
```

```
cm>  /send 0060000018EB9BC6DCDF74FF4E4B43F2E34A6727B62755AE2D0000800008
 => 00 60 00 00 18 EB 9B C6 DC DF 74 FF 4E 4B 43 F2     .`........t.NKC.
    E3 4A 67 27 B6 27 55 AE 2D 00 00 80 00 08           .Jg'.'U.-.....
 (1452 usec)
 <= A8 AD 62 59 7D 9A 92 E8 90 00                       ..bY}.....
Status: No Error
```

返回的过程密钥为 A8AD62597D9A92E8，使用 des.exe 验证

两结果相同说明此时计算过程密钥代码工作正常,**之后不再验证过程密钥计算函数**。通过过程密钥计算 MAC1

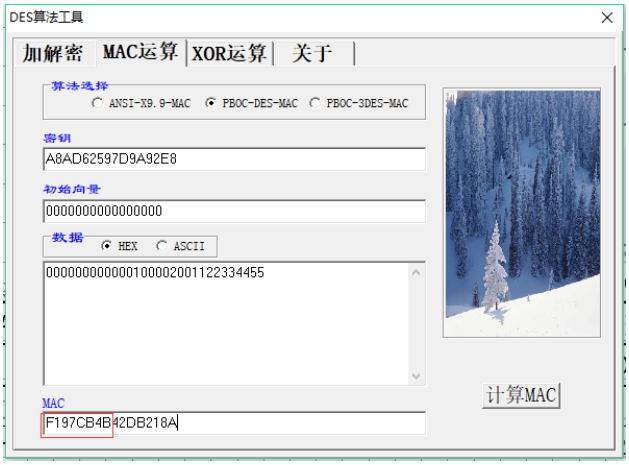
CLA	INS	P1	P2	Lc	Data 过程密钥|| EP 余额（交易前）||交易金额||交易类型||终端机编号		Le
00	62	00	00	17	A8AD62597D9A92E8	0000000000010000200112233445	04

命令如下

```
/send 0062000017A8AD62597D9A92E8000000000000010000200112233445504
```

```
cm>  /send 0062000017A8AD62597D9A92E8000000000000010000200112233445504
 => 00 62 00 00 17 A8 AD 62 59 7D 9A 92 E8 00 00 00     .b.....bY}......
    00 00 00 10 00 02 00 11 22 33 44 55 04              ........"3DU.
 (1338 usec)
 <= F1 97 CB 4B 90 00                                   ...K..
Status: No Error
```

此时返回的 MAC1 为 F197CB4B 与圈存初始化时 MAC1 一致，此时在使用 des.exe 验证



MAC1 一致说明计算 MAC 的函数正常。**之后不再验证 MAC 计算函数**

3.5.2 圈存

圈存需要生成 MAC2 报文如下：

CLA	INS	P1	P2	Lc	Data 过程密钥||交易金额||交易类型||终端机编号||交易日期||交易时间		Le
00	62	00	00	17	A8AD62597D9A92E8	000010000200112233445520111221214822	04

命令如下：

```
/send 006200001AA8AD62597D9A92E8000010000200112233445520111221214822 04
```

```
cm>  /send 006200001AA8AD62597D9A92E8000010000200112233445520111221214822 04
 => 00 62 00 00 1A A8 AD 62 59 7D 9A 92 E8 00 00 10     .b.....bY}......
    00 02 00 11 22 33 44 55 20 11 12 21 21 48 22 04     ....."3DU ..!!H".
 (1741 usec)
 <= C9 20 43 E5 90 00                                   . C...
Status: No Error
```

此时返回的 MAC2 为 C92043E5，通过此 MAC2 圈存命令报文如下：

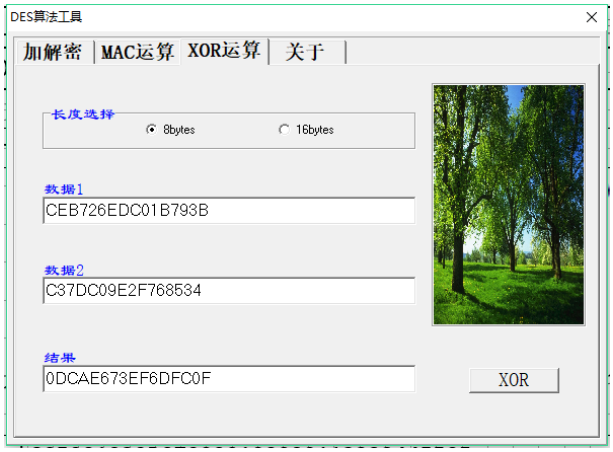
CLA	INS	P1	P2	Lc	Data 交易日期||交易时间||MAC2			Le
80	52	00	02	0B	20111221	214822	C92043E5	04

命令如下

```
/send 805200000B20111221214822C92043E504
```

```
cm>  /send 805200000B20111221214822C92043E504
 => 80 52 00 00 0B 20 11 12 21 21 48 22 C9 20 43 E5     .R... ..!!H". C.
    04                                                  .
 (3494 usec)
 <= 14 62 AD 13 90 00                                   .b....
Status: No Error
```

此时返回 TAC 为 1462AD13，之后需要我们验证 TAC，密钥此时为 TAC 密钥左 8 个字节与右 8 字节异或



此时的密钥为 0DCAE673EF6DFC0F，使用和计算 MAC 方式相同的方法计算 TAC

CLA	INS	P1	P2	Lc	Data 密钥||余额（交易后）||联机交易序号||交易金额||交易类型||终端机编号||交易日期||交易时间		Le
00	62	00	00	20	0DCAE673EF6DFC0F	00001000000000001000020011223344552011122121482204		04

具体报文如下

```
/send 00620000200DCAE673EF6DFC0F00001000000000001000020011223344552011122121482204
```

```
cm>  /send 00620000200DCAE673EF6DFC0F00001000000000001000020011223344552011122121482204
 => 00 62 00 00 20 0D CA E6 73 EF 6D FC 0F 00 00 10     .b.. ...s.m.....
    00 00 00 00 00 10 00 02 00 11 22 33 44 55 20 11     ..........."3DU .
    12 21 21 48 22 04                                   .!!H".
 (2193 usec)
 <= 14 62 AD 13 90 00                                   .b....
```

此时 TAC 验证一致。

3.6 测试消费

3.6.1 消费初始化

消费初始报文如下

CLA	INS	P1	P2	Lc	Data 密钥索引||交易金额||终端机编号			Le
80	50	01	02	0B	07	00001000	001122334455		0F

具体命令如下

```
/send 805001020B0700001000001122334455 0F
```

```
cm>  /send 805001020B0700001000001122334455 0F
 => 80 50 01 02 0B 07 00 00 10 00 00 11 22 33 44 55     .P.........."3DU
    0F                                                  .
 (480023 nsec)
 <= 00 00 10 00 00 00 00 00 00 01 00 C7 AD CA 50 90     ..............P.
    00                                                  .
Status: No Error
```

此时我们可以看到返回的信息如下

EP 余额	EP 脱机交易序列号	透支限额	密钥版本号 DPK	算法标识 DPK	伪随机数(IC 卡)
00001000	0000	000000	01	00	C7ADCA50

此时消费需要计算 MAC1，首先计算过程密钥，报文如下

CLA	INS	P1	P2	Lc	Data 密钥||伪随机数|| EP 脱机交易序列号||终端交易序号的最右两个字节		Le
00	60	00	00	0x18	09F4ACB09131420B8FE1B4CC007AC52B	C7ADCA5000000304		08

命令如下：

```
/send 006000001809F4ACB09131420B8FE1B4CC007AC52BC7ADCA500000030408
```

```
cm>  /send 006000001809F4ACB09131420B8FE1B4CC007AC52BC7ADCA500000030408
 => 00 60 00 00 18 09 F4 AC B0 91 31 42 0B 8F E1 B4     .`........1B....
    CC 00 7A C5 2B C7 AD CA 50 00 00 03 04 08           ..z.+...P.....
 (1535 usec)
 <= 5A D8 7B 7F CA 01 D1 C6 90 00                       Z.{.......
Status: No Error
```

此时返回的过程密钥为 5AD87B7FCA01D1C6，之后计算 MAC1

CLA	INS	P1	P2	Lc	Data 过程密钥||交易金额||交易类型标识||终端机编号||交易日期||交易时间		Le
00	62	00	00	1A	5AD87B7FCA01D1C6	00001000060011223344552011122121482204		04

```
cm>  /send 006200001A5AD87B7FCA01D1C60000100006001122334455201112212148220 4
 => 00 62 00 00 1A 5A D8 7B 7F CA 01 D1 C6 00 00 10     .b...Z.{........
    00 06 00 11 22 33 44 55 20 11 12 21 21 48 22 04     ....."3DU ..!!H".
 (1927 usec)
 <= 5B 44 D9 7E 90 00                                  [D.~..
Status: No Error
```

此时得到 MAC1 为 5B44D97E。

3.6.2 消费

CLA	INS	P1	P2	Lc	Data 终端交易序号||交易日期||交易时间||MAC1				Le
80	54	01	00	0F	01020304	20111221	214822	5B44D97E	08

具体命令如下：

```
/send 805401000F01020304201112212148225B44D97E08
```

```
cm>  /send 805401000F01020304201112212148225B44D97E08
 => 80 54 01 00 0F 01 02 03 04 20 11 12 21 21 48 22     .T....... ..!!H"
    5B 44 D9 7E 08                                      [D.~.
 (4355 usec)
 <= 11 83 BB A1 A2 41 AE 85 90 00                       .....A....
Status: No Error
```

此时我们得到 TAC 为 1183BBA1，MAC2 为 A241AE85，我们需要验证计算 MAC2

CLA	INS	P1	P2	Lc	Data 过程密钥||交易金额		Le
00	62	00	00	0C	5AD87B7FCA01D1C6	00001000	04

命令为

```
/send 006200000C5AD87B7FCA01D1C60000100004
```

```
cm>  /send 006200000C5AD87B7FCA01D1C60000100004
 => 00 62 00 00 0C 5A D8 7B 7F CA 01 D1 C6 00 00 10     .b...Z.{........
    00 04                                               ..
 (1261 usec)
 <= A2 41 AE 85 90 00                                   .A....
Status: No Error
```

此时 MAC2 验证成功，之后验证 TAC

CLA	INS	P1	P2	Lc	Data 密钥||交易金额||交易类型||终端机编号||交易终端序号||交易日期||交易时间		Le
00	62	00	00	1A	0DCAE673EF6DFC0F	00001000060011223344550102030420111221214822	04

命令为

```
/send 006200001E0DCAE673EF6DFC0F00001000060011223344550102030420111221214822 04
```

```
cm>  /send 006200001E0DCAE673EF6DFC0F00001000060011223344550102030420111221214822 04
 => 00 62 00 00 1E 0D CA E6 73 EF 6D FC 0F 00 00 10     .b......s.m.....
    00 06 00 11 22 33 44 55 01 02 03 04 20 11 12 21     ....."3DU.... ..!
    21 48 22 04                                         !H".
 (2080 usec)
 <= 11 83 BB A1 90 00                                   ......
Status: No Error
```

此时 TAC 一致说明验证成功。

3.6.3 超额消费

经过之前的圈存和消费此时卡内金额应该为 0，再次使用消费初始化命令

```
cm>  /send 805001020B07000010000011223344550F
 => 80 50 01 02 0B 07 00 00 10 00 00 11 22 33 44 55    .P..........."3DU
    0F                                                 .
 (3253 msec)
 <= 94 01                                              ..
Status: 0x9401
```

此时返回 9401 以为余额不足无法消费。

3.7 余额查询

余额查询命令如下

```
/send 805C000204
```

```
cm>  /send 805C000204
 => 80 5C 00 02 04                                     .\...
 (5119 msec)
 <= 00 00 00 00 90 00                                  ......
Status: No Error
```

此时经过之前的冲值与消费余额应该为 0 和查询结果一致。