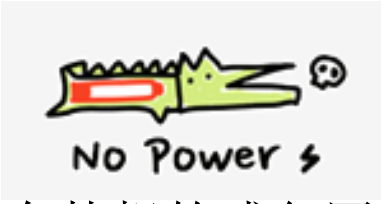
电子钱包的圈存交易



有梦想的咸鱼四

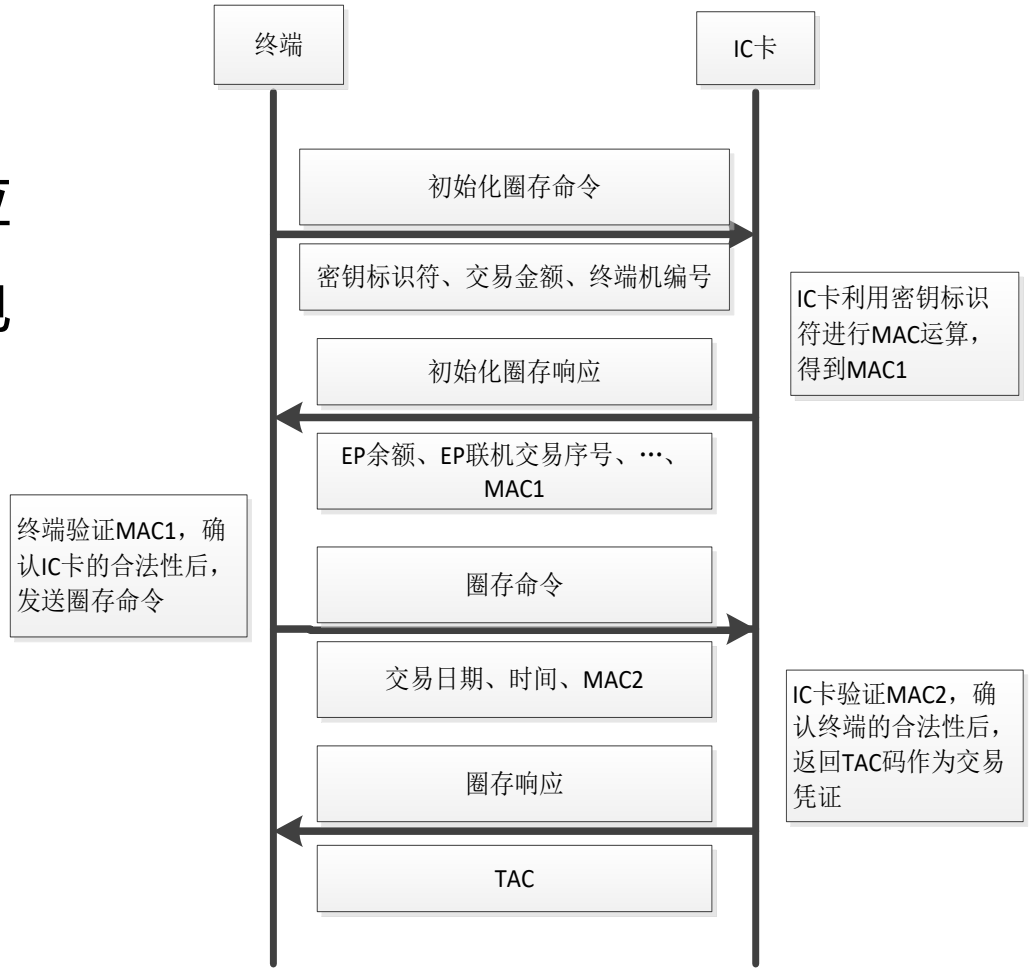
4 / 7 / 2017

目录

- 圈存
- 安全管理

圈存

- 概念：
  - 持卡人将其在银行相应帐户上的资金划转到电子钱包中。
- 工作流程如右图：



圈存

- 终端发出初始化圈存命令

代码	值	
CLA	80	
INS	50	
P1	00	
P2	02	
Lc	0B	
Data	密钥标识符	08
	交易金额	00 00 10 00（40.96元）
	终端机编号	00 11 22 33 44 55
Le	10	

圈存

- IC卡对初始化圈存命令进行处理

1. IC卡根据密钥标识符查找圈存密钥
2. IC卡生成随机数，利用圈存密钥产生过程密钥。
3. IC卡利用所生成的过程密钥产生MAC1。，
4. IC卡将返回相应的数据

说明	值
EP余额	00 00 00 00
EP联机交易序列号	00 00
密钥版本号DPK	01
算法标识DPK	00
伪随机数（IC卡）	B1 EE 18 0C
MAC1	F2 0B 5E 52

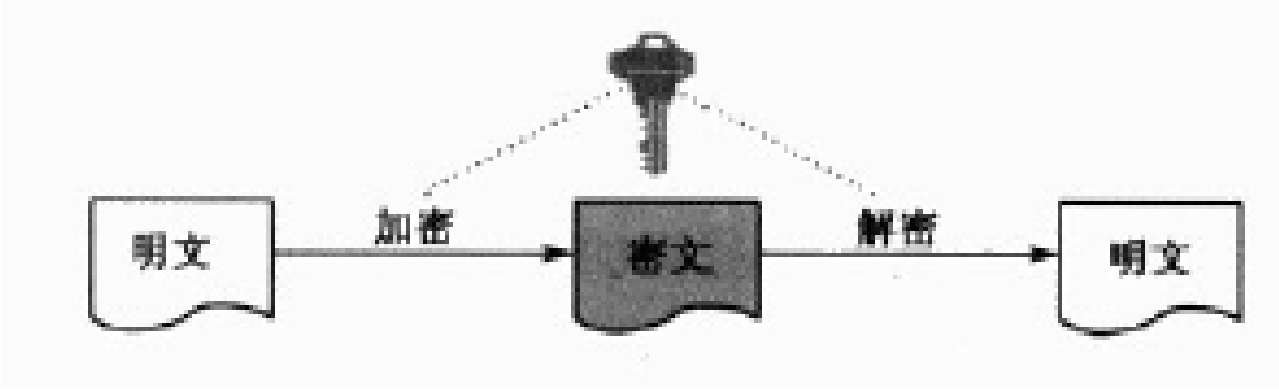
圈存

- IC卡收到圈存命令后，验证MAC，并生成TAC码，返回给终端。

说明	长度（字节）
TAC码	4

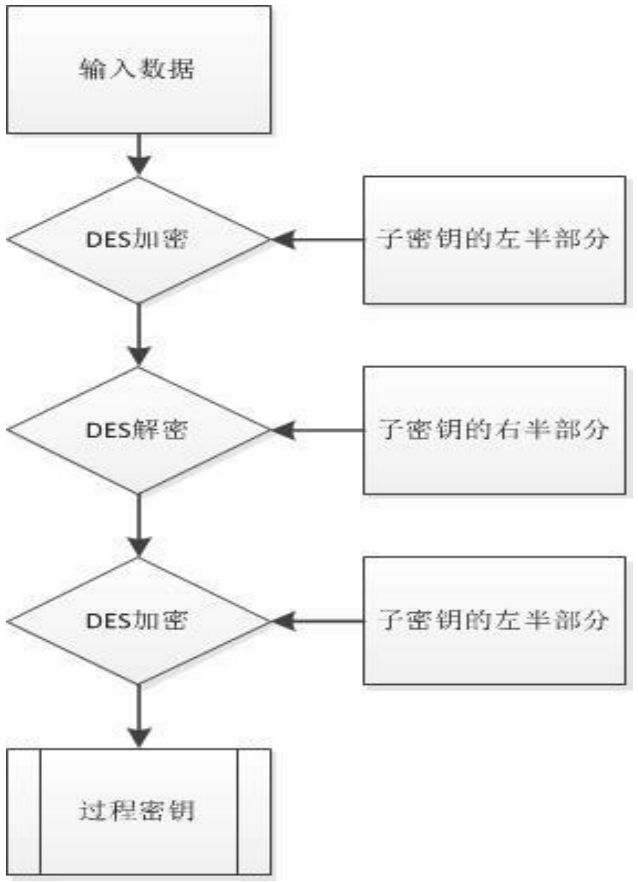
安全管理

- 数据的加解密运算
  - 以最为常见的DES加密算法为例。DES算法是一种对称加密算法，其加密和解密所使用的密钥一致。



安全管理

- 过程密钥的生成
  - 过程密钥是在交易过程中用可变数据产生的8字节长密钥
  - 子密钥总长是16字节，输入数据是8字节，得到的过程密钥是8字节



安全管理

- MAC或TAC的生成
  - MAC 是报文鉴别码，是终端和卡片之间的身份认证
  - TAC 是交易验证码，是作为交易的一个有效凭证
  - MAC和TAC需**使用过程密钥来计算**，计算方法相同，但是输入的数据不同，具体过程如下：

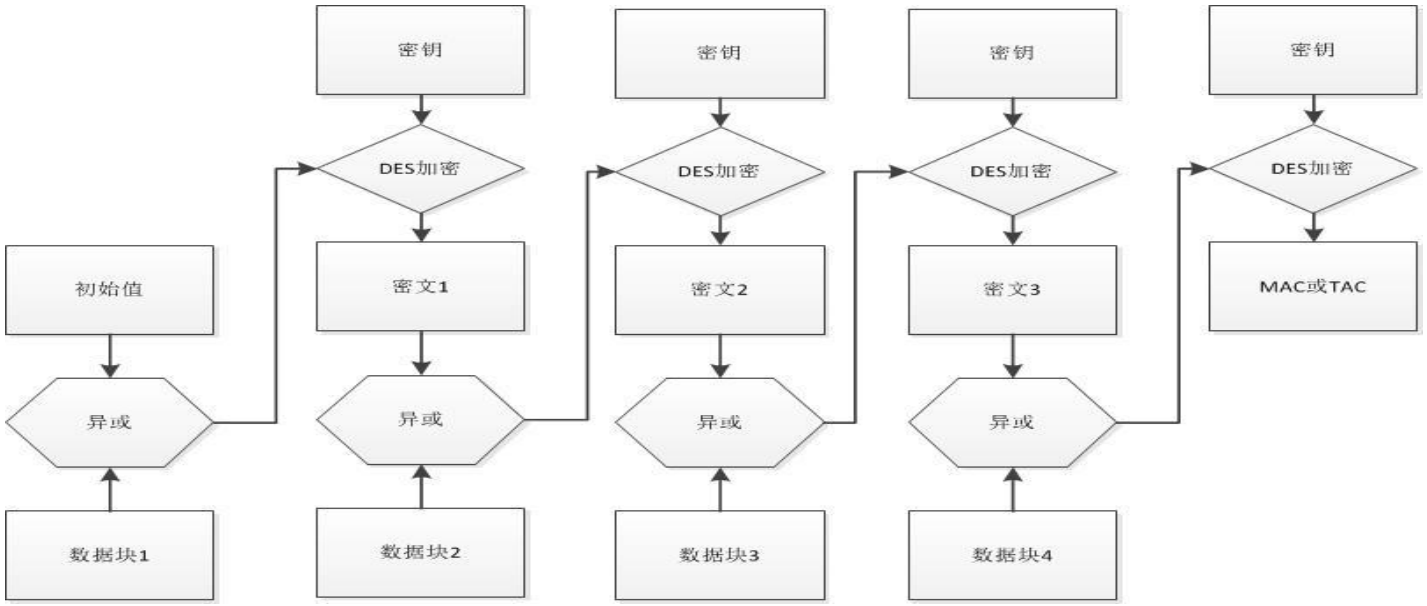
1. 将**初始值**设定为8个字节长的16进制数（0x 00 00 00 00 00 00 00 00）。

2. 数据**填充**：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。

安全管理

- MAC或TAC的生成

3. 将这些数据分割为8字节长的数据块组。
4. 由下述方式生成MAC或TAC



安全管理

- 数据的加解密运算
  - 首先使用KeyBuilder中的buildKey()方法**生成DES密钥实例**

```
Key deskey = KeyBuilder.buildKey(KeyBuilder.TYPE_DES, KeyBuilder.LENGTH_DES, false);
```

  - 使用Cipher类中的getInstance()函数来**获得加密实例**

```
Cipher desEngine = Cipher.getInstance(Cipher.ALG_DES_ECB_NOPAD, false);
```

安全管理

- 数据的加解密运算（cdes函数）
  - 使用DESKey接口中的setKey()函数**设置密钥值**

```
((DESKey)deskey).setKey(akey, kOff);
```

  - 使用Cipher类中的init()函数来**设置加密对象实例**

```
desEngine.init(deskey, mode);
```

  - 使用Cipher类中的doFinal()函数来完成运算

```
desEngine.doFinal(data, dOff, dLen, r, rOff);
```

安全管理

- 随机数的产生
  - 采用getInstance()函数来获得随机数类的实例

```
RandomData rd = RandomData.getInstance(RandomData.ALG_SECURE_RANDOM);
```

  - 调用generateData()函数生成随机数

```
rd.generateData(v, (short)0, (short)size);
```

任务

① 完成PenCiper.java中的两个函数，分别是计算过程密钥和计算MAC值；

② 通过测试，如读取计算后的密钥或MAC值，检验是否正确；

③ 读懂EPFile.java和Purse.java中初始化圈存和圈存两部分内容，在完成安全管理部分后测试圈存部分。

Task one

```
/*
 * 功能：生成过程密钥
 * 参数：key 密钥； data 所要加密的数据； dOff 所加密的数据偏移量； dLen 所加密的数据长度； r 加密后的数据；
 * rOff 加密后的数据存储偏移量
 * 返回：无
 */
public final void gen_SESPK(byte[] key, byte[]data, short dOff, short dLen, byte[] r, short rOff)
```

调用cdes函数进行过程密钥的生成：

参数：

- key 密钥;
- kOff 密钥的偏移量;
- data 所要进行加解密的数据;
- dOff 数据偏移量；
- dLen 数据的长度;
- r 加解密后的数据缓冲区；
- rOff 结果数据偏移量；
- mode 加密或解密运算模式

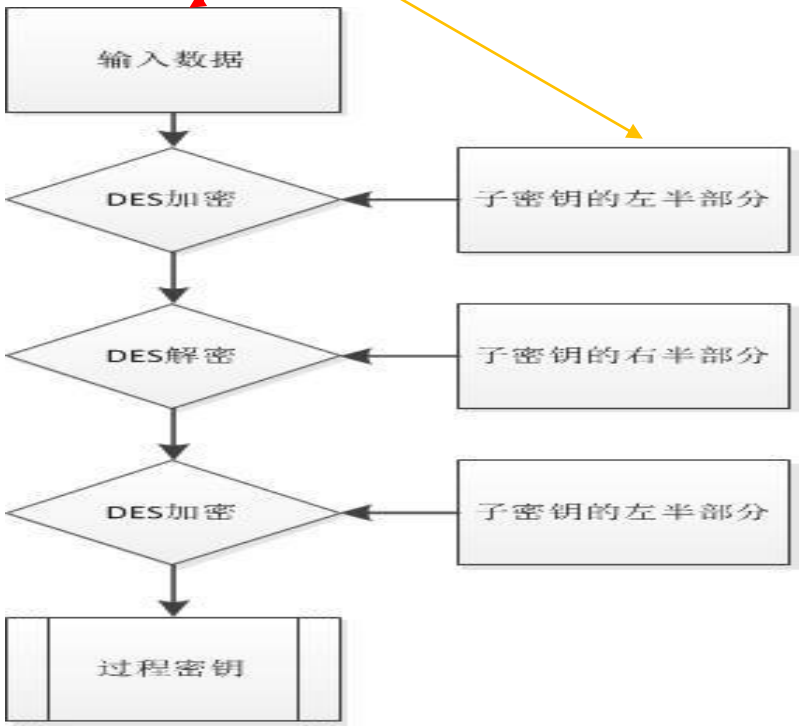
Mode参数选择：

Cipher.*MODE_ENCRYPT* *加密*

Cipher.*MODE_DECRYPT* *解密*

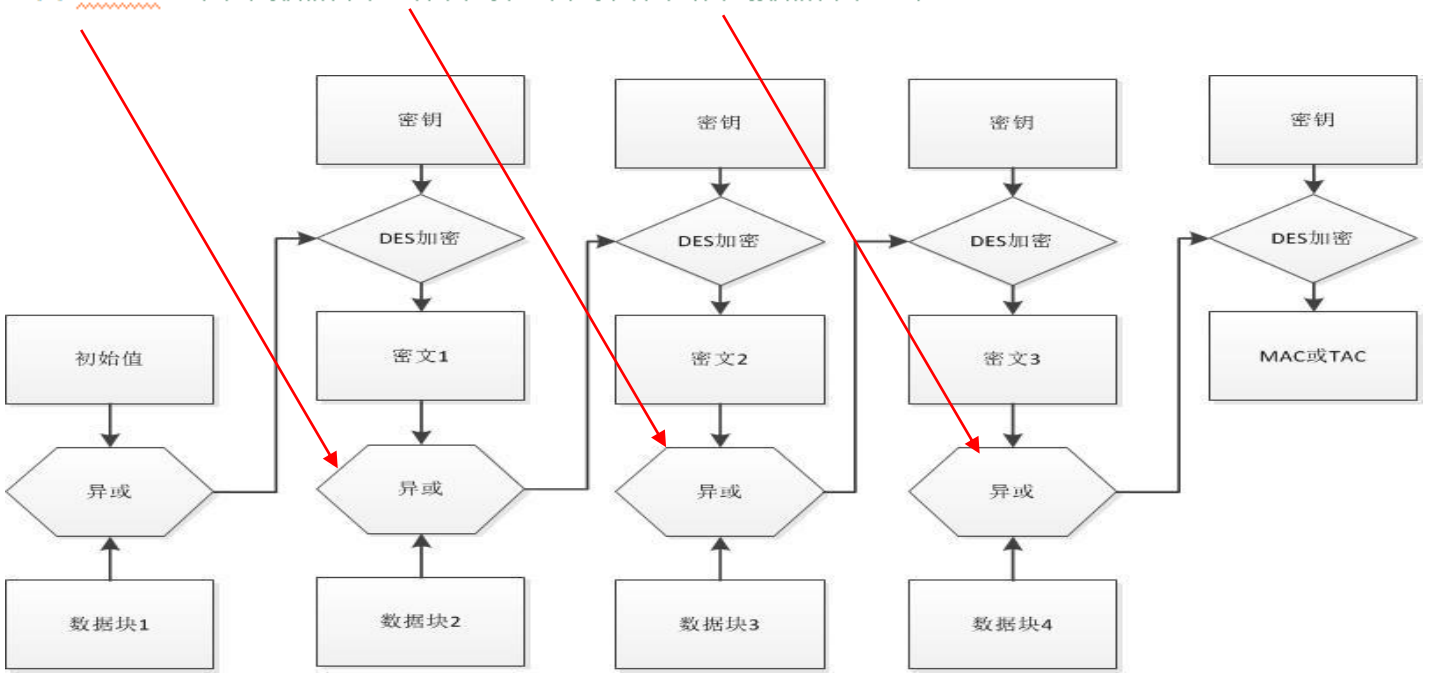
Task one

```
public final void gen_SESPK(byte[] key, byte[]data, short dOff, short dLen, byte[] r, short rOff)
```



Task Two

```
/*
 * 功能：8个字节的异或操作
 * 参数：d1 进行异或操作的数据1 d2:进行异或操作的数据2 d2_off:数据2的偏移量
 * 返回：无
 */
public final void xorblock8(byte[] d1, byte[] d2, short d2_off){
    //todo: 两个数据块进行异或，异或结果存入数据块d1中
```



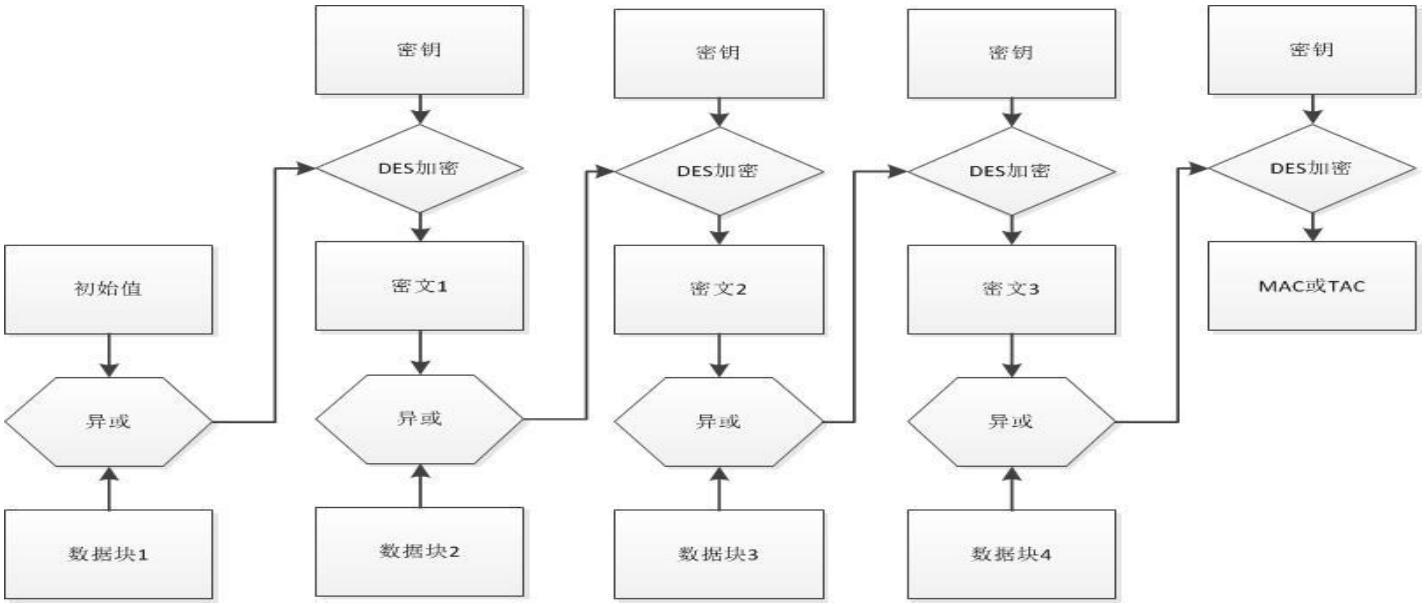
Task Three

```
/*
 * 功能：字节填充
 * 参数：data 所要填充的数据； len 数据的长度
 * 返回：填充后的字节长度
 */
public final short pbocpadding(byte[] data, short len){
    //todo: 填充字符串至8的倍数
```

先进行尾数填充0x80，然后再进行判断是否是8的倍数即可

Task Four

```
/*
 * 功能：MAC和TAC的生成
 * 参数：key 密钥; data 所要加密的数据; dl 所要加密的数据长度; mac 所计算得到的MAC和TAC码
 * 返回：无
 */
public final void gmac4(byte[] key, byte[] data, short dl, byte[] mac){
```



伪代码：

新建临时数组tmp并进行初始化
调用pbocpadding函数对data进行填补并得到长度
对数据段进行循环遍历，每8个一组调用xorblock8函数和cdes函数
对mac进行赋值，mac为tmp数组的前4位

脚本测试：

圈钱初始化命令发出后，如果运行正确，则会返回加密算法使用的随机数

#//圈存初始化
/send 805000020B08000010000011223344551O

```
cm>  /send 805000020B08000010000011223344551O
 => 80 50 00 02 0B 08 00 00 10 00 00 11 22 33 44 55
    10
 (3092 usec)
 <= 00 00 00 00 00 00 01 00 06 61 CA C8 30 91 0B 2C
    90 00
Status: No Error
```

通过使用这个随机数，来进行运算mac2从而修改圈钱指令，然后卡片再根据mac2来判断是否应该执行圈钱。这里提供了一个Des.exe来方便大家计算

本屁屁踢为**商小四原**创，**未**经**允**许**不得**转载
引用请**注明出**处

小四出品，必属精品