面向对象之 PED

前言：

大家都知道，一个 POS 设备最核心有两块：PED（PIN Entry Device）和智能卡读卡器。围绕这两个安全模块的 IC 及认证费用也是最高的，比如 EMV，PBOC，PCI，及各个区域一系列的安全认证。
一个 POS 设备从研发到生产，几乎一半费用都在认证上，三分之一的投入是模具开模。而一个 POS 设备为什么研发需要近一年的时间，其主要原因也是模具的安全要求，和认证周期。

今天要说的是密钥管理相关的 PED 设备，这也是 POS 里面最核心的一个设备，它处理着 POS 的密钥管理，加解密运算，安全监测等业务。也就是说，如果一个 POS 设备没有 PED，那么是无法符合安全要求的，也过不了系列认证。
这要求 POS 使用了安全的硬件环境，如安全 CPU，结构上安全触法检测开关，密钥的存储，注入，使用都在安全保护的环境下。如果一个 POS 设备达不到这样的要去，那么就是伪安全的。比如著名的 2012 年的新国都 POS 事件，就让这家公司损失惨重。

PED 成为了很多电子厂商想进入 POS 领域的软门槛，自然 PED 相关技术成为了 POS 厂商的核心技术，而 PED 的需求也随着各个地区银行或 ISP 的安全要求不断增加，以造成每个 POS 厂商 PED 软硬件实现的复杂度。

如今，大部分 POS 厂商使用了安全 CPU，即 CPU 自带安全 RAM 和 ROM，将根密钥数据和公钥存放其中，另外进行总线的保护及各路 sensor 进行安全监测。因为 CPU 设计的复杂度，以造成 CPU 的昂贵，这是为什么 POS 的 CPU 一直落后很多电子设备的主要原因。为了处理高速运算，上 Andriod 和 WM 系统，很多 POS 厂商使用了协处理器，即将安全运算部分使用安全 CPU，图形及多媒体的处理使用通用 CPU。这些都是 POS PED 的硬件需求。在 PCI 的规范中，对硬件和结构还有非常多的要去。而这里主要谈的是软件。

什么是 PED 软件系统：

PED 软件是 POS 软件系统最核心的部分，他贯穿了软件系统最下层的 CPU 启动程序到软件系统最上层的业务应用程序。他的整个作用从密钥的产生，注入，存储，使用，销毁，到安全的密钥输入界面的控制，和银行前置数据通信的加解密运算，涉及到 POS 业务的整个生命周期。甚至于 IC 卡和 RF 卡的脱机联机交易，都离不开 PED。

可以说 PED 就是整个系统安全的基础，在整个 PED 系统中，有非常多的密钥体系，如对称，非对称，或 DUKPT，同时也涉及到多套运算方法，如 TDES，RSA，ECB，MAC，PINBLOCK。因为密钥的签名，还使用了 HASH 算法。当然，因为各地需求的不一致，很多地方还有自己的密钥管理方法和算法，也要求 POS 厂商提供在软件系统中。

传统 PED 的文件结构：

一个巨大的 PED 文件，需要一开始就根据密钥体系的要去初始化分配密钥区域，定义每个区域密钥长度，密钥的功能，密钥派生和从属关系。在安全 Sensor 被触法或密钥 crc/hash 监测不过的时候，该文件被格式化。

正如 PED 软件工作过程所要求的，这是一种面向过程的方法建立的 PED 密钥体系。其好处在于，简单，使用方便。基本是一个萝卜一个坑。但是坏处在于一旦有多的萝卜，坑就不够，或者萝卜的大小不一样，就要重新针对这个萝卜去加坑。

这就表示，一旦 PED 的对象多了，类别多了，不同的收单机构的需求就很难兼容。可行的做法是分支版本或者不断向上兼容，不断升级软件版本。这给软件系统的维护带来非常大的不便。

例如，传统需求 Master Key 需要 30 组，但是突然有一天某地收单行需要 50 组，这就需要改软件结构，又突然有一天某地区要求 100 组，这里就又需求增加到 100 了。于是升级版本带来了一系列的开发，测试，发布，维护的工作。从此面向过程，最终面向死亡。

什么是面向 PED:

既然面向过程的方法遭遇到如此多的困难，那么面向对象的方法是否可以有效解决 PED 的技术难题呢？

面向对象的核心思想是类和对象。其起源于我们怎么思考这个世界，这个世界是由什么组成的。

话题太大了，那么就算 PED 软件系统。

如果把 PED 软件系统当作对象，那么 PED 软件系统是由什么组成的，有人说函数，有人说密钥，有人说密钥体系，有人说加解密算法，有人说数据...

是的，作为一个软件系统设计人员来看，PED 是由数据和功能组成的，而数据又分为密钥，随机数，被加密数据，参数等等。

这就分出了很多的类别。

站在抽象的角度，如果给类下定义，什么是数据，那么就要看这些数据类的特征，这个特征就包括了属性和方法，

每个数据都有属性比如明文，密文，随机产生，密钥长度，格式，而每个数据可以用作加密，解密，填充，签名，计数这些方法。

数据是一个抽象的概念，它仅仅是一个概念而不是实体。但是具有某种属性和方法的数据则是实体。每一个数据都是实体的一个对象。
如果对 PED 里的数据进行有效类别的划分，描述所具有相同属性和方法的对象，那么就可以建立类。

那么在定义整个 PED 系统结构的时候，就可以认定 PED 里有一系列的不同特征的数据，为这些不同数据赋予自己的方法。
所以面向 PED 就可以编写自己需要的数据类型，以便更好地解决问题。

如何使用面向对象：
例如每个 ACQ 对密钥管理系统的需求不一样，如一个 POS 设备需要满足不同 ACQ 的要求，这就必须建立不同的 PED 对象模型。
即每一个 ACQ 对应一个 PED 对象。针对不同的 ACQ，在使用前初始化不同的 PED 对象模型：如 ACQ 要求的密钥层次，密钥长度，密钥区域，派生关系，密钥的功能。除密钥之外，其他的数据也一样。

```
private class key
{
private String name; //密钥名称
private int length; //密钥长度
private char zone; //密钥区域
private boolean derived; //派生层次
public char usage;//可以用来做的加解密功能
//……等等
}

public void setName(String a){ //这个方法是用设置密钥名称的
name= a;
}

public String getName(){ //这个方法是得到密钥的姓名
return name;}
```

依次类推，可以给密钥增加加解密的功能函数等等。

那么针对不同 ACQ，应用程序可以方便建立多级密钥体系，不同大小的密钥区域，不同长度的密钥，以及每种密钥都可以建立自己特有的运算法则。

POS 只会为 PED 的属性分配内存。因为每个 PED 的都不一样！就像你往公司带的午饭和我往公司带的午饭不一样是一个道理！但方法就不同了。早晨带的饭中 午就凉了，你需要用微波炉来加热。微波炉可不用你带，公司就有（只占公司的一块空间），它放在了午餐桌上。你想想，微波炉属于谁的？它属于所有员工的！因 为每个员工都可以用它。而不必每个员工都带一份。由此可见，每个员工（对象）都有一份午饭（属性），但所有的员工（对象）只一个微波炉（方法）。所有的员 工（对象）都可以通过这个微波炉（方法）来改变自己午餐（属性）的冷热状态。殊途同归！在 POS 中也就是这样，方法只有一份，供所有的 PED 使用！而属性是 每个 PED 一份，因为每个 PED 的都不一样。

我眼中的密钥管理：

对我而言，PED 最核心的是密钥管理的功能。而密钥是一种特殊的数据，这好比 PED 是一个酒店，酒店里有很多房间，每个房间都有自己的门锁，想要进入房间使用房间，就需要钥匙。因为每个房间大小不一样，功能不一样，住的人也不一样，所以在使用的时候就必须保证房间不能被滥用，这就要求房间使用合理，保证房间每一个钥匙都是安全的，不会出现同一个房间在同时被不同客户使用，也不会出现钥匙都复制丢失。所以 PED 密钥管理和酒店物业管理是一个道理。

我在 2008 年的时候就曾经为前公司设计并编码了整套面向对象的密钥体系，其实比我今天诉说的要完整很多，但是可惜并未被接受。

事实证明每几个月就会有不同部门的人因为 PED 的需求而重新讨论整理一遍，当然，如果一件事情的维护成本要远小于开发成本，那么不断修改和升级也是可以接受的。

之所以随手写写这个文章，是因为 PED 并没有想象中那么复杂和神秘，也不存在所谓的 PED 专家，大多数专家是因为领导自己不懂技术而造就的。

当然，由于我已经好多年不做 PED 开发了，所以文章的一些观点并不一定正确，如有同行质疑，那么一定是我错了。