金融行业密钥详解

金融行业因为对数据比较敏感，所以对数据的加密也相应的比较重视。在其中有关密钥及加密方面的文章很少，并且散发在各个银行及公司的手中，在网上没有专门对这部分进行介绍的。本文对金融行业的密钥进行较深入的介绍，包括象到底什么是主密钥（MasterKey）、传输密钥（MacKey），为什么我们需要这些东西等。

本文采取追源溯本的方式，力求让对这感兴趣的人达到知其然，同时也知其所以然，而不是模模糊糊的知道几个概念和名词。因为本文主要是针对对金融行业密钥不是很熟悉的人，所以如果你对密钥很熟悉就不必仔细看了。

好了，咱们言规正传。我们知道，金融行业有很多数据要在网络上传递，包括从前置到主机，从自助终端到前置等，这些数据在网络上传来传去，我们很容易就会想到安全性的问题，如果这些数据被人窃取或拦截下来，那我们怎么敢在银行存钱了。这个问题在计算机出现时就被前人考虑到了，所以出现了很多各种各样的加解密技术。

抛开这些不管，假设当初由我们自己来设计怎样解决数据被窃取的情况。假设我们有一段数据，是 ATM 取款的报文，包括一个人的磁卡号、密码、取款金额，现在需要将这些数据从一台 ATM 机器传到前置机处理，这些数据是比较机密的，如果被人窃取了，就可以用该卡号和密码把帐户中的钱取走。

首先，我们可以想到用专用的银行内部网络，外面的人无法获得网络的访问权。这个仔细想想显然不可行的，因为一是不能保证外人一定没办法进入银行内部网络，二是银行内部人员作案是没法防止的。

接着，我们很容易想到，既然保证数据不被窃取的可能性很小，那我们何不变换一下思路，数据避免不了被窃取，那我如果将数据处理下，让你即使窃取到数据，也是一些无用的乱码，这样不就解决问题了吗。这个想法比较接近现在的做法了，当前置机接收到了数据，它肯定是对数据进行反处理，即与 ATM 端完全步骤相反的数据处理，即可得到明文的数据。我们再进一步想想，如果因为某种原因，报文中的取款金额被改变了，这样就会导致 ATM 出的钱和前置扣帐记录的钱不一致的情况，看来我们必须加上一个验证机制，当前置机收到 ATM 发送的一个报文时，能够确认报文中的数据在网络传输过程中没有被更改过。

怎样实现？最简单的，象计算机串口通讯一样，对通讯数据每一位进行异或，得到 0 或 1，把 0 或 1 放在在通讯数据后面，算是加上一个奇偶校验位，收到数据同样对数据每位进行异或，得到 0 或 1，再判断下收到数据最后一位与算出来的是否一致。这种方式太简单了，对于上面提到的 ATM 到前置机的报文来说，没什么用处，不过我们可以将对数据每一位异或的算法改成一个比较复杂点的。

因为 DES 算法已经出来了很多年了，并且在金融行业也有广泛的应用，我们何不用 DES 算法进行处理，来解决上面的问题呢。我们应该了解 DES 算法（此处指单 DES）的，就是用一个 64bit 的 Key 对 64bit 的数据进行处理，得到加密后的 64bit 数据。那我们用一个 Key 对上面的报文进行 DES 算法，得到加密后的 64bit 数据，放到报文的最后，跟报文一起送到前置机，前置机收到报文后，同样用 Key 对数据（不包括最后的 64bit 加密数据）进行 DES 加密，得出 64bit 的数据，用该数据与 ATM 发送过来的报文最后的 64bit 数据比较，如果两个数据相同，说明报文没有中途被更改过。

再进一步，因为 DES 只能够对 64bit 的数据进行加密，一个报文可不止 64bit，哪我们怎么处理呢？只对报文开头的 64bit 加密？这个是显然不够的。

我们可以这样，先对报文的开始 64bit 加密，接着对报文第二个 64bit 加密，依次类推，不过这有问题，因为每个 64bit 都会得到同样长度的加密后的数据，我不能把这些数据都放到报文的后面，那报文的长度不变成两倍长了。换个思路，我先对报文第一个 64bit 加密，得到 64bit 的加密后数据 data1，接着再拿加密后的 data1 与报文第二个 64bit 数据进行按位异或，得到同样长 64bit 的数据 data2，我再用 Key 对 data2 加密，得到加密后的数据 data3，再拿 data3 与报文第三个 64bit 数据进行按位异或，同样的处理依次类推。直到最后会得到一个 64bit 的数据，将这个数据放到报文的最后发到前置机，这样报文的长度只增加了 64bit 而已。这个算法就叫做 MAC 算法。

好了，到目前为止我们已经知道了什么是 MAC 算法，为什么需要它，接着我们再看看经常被提起的另外一个名词。在上面说到 MAC 算法的时候，我们会注意到其中进行 DES 加密算法时提到了一个 Key，这个用来参与 MAC 计算的 Key 就常被称为 MacKey，也有叫工作密钥、过程密钥的。

我们继续来处理 ATM 和前置机间网络数据传输的问题。前面提到的 MAC 算法对传送的报文进行了处理，保证了在网络传输过程中数据不会被有意或无意的篡改，但是，我们再进一步想想，如果仍然是上面提到的一个取款报文，如果想作案的话，我不改报文的内容，我只是截取报文的内容，因为内容里面有卡号和密码，都是明文的形式，很容易就看出来哪些内容是卡号、哪些内容是密码。有了卡号和密码，我就好办了，找个读卡器就能够很快的制出一张磁卡，然后拿这个磁卡可以随便取钱了，根本不需要修改报文，这样你就算前置机对报文的 MAC 校验通过了，也只是保证了报文没改动过，对于防止作案没有实质上的帮助。

那我们很容易想到，我再加上一道加密，这次我把整个存款的报文都用 DES 加密，将明文全部转换成密文，然后送到前置机，这下好了吧。即使你把报文截取了也没用，你拿着这些密文也没有用，你也没有 DES 的密钥来解密它，只有前置机才知道密钥。这是个好主意，确实防止了卡号和密码等被人获知的危险。这也是现在普遍采取的做法，不过我们需要对这个做法进行一些改进。

首先，我们要知道用 DES 对数据加解密是耗时间的，尤其是使用硬加密（下一步讲什么是硬加密）的情况，速度是比较慢的。我们来想想，整个存款报文有必要每个数据都 DES 加密吗，象报文中的什么流水号、ATM 号等信息，对它们加密没什么意义，进一步讲，取款金额加密也没意义，假设你取 500 块，但是你将报文改成了 100 块，导致主机只把你帐户扣 100 块钱，你白赚了 400 块。这个听起来挺划算的，实际上是不可行的，因为这样造成了帐务上的短款，银行当然会查账的，根据 ATM 记录的硬件出钞张数和主机扣款金额，肯定会把你查出来的，那这种掩耳盗铃的做法，下场显而易见，想必没人这么傻。

我们来考虑一个报文中到底什么信息是需要加密的，目前一般的做法是只对帐号和密码（也有只对密码加密的）进行加密，其他的内容不加密的，明文就明文，没什么大不了的。对帐号和密码加密有个术语，我们可能都听说过，叫 PinBlock，即 PIN 块，就是对帐号和密码进行 DES 加密处理后的一个密文数据块。即然使用了 DES 算法来加密帐号和密码，则必然有个 Key 来加密，那么我们就把这个 Key 称为 PinKey，就是专门来加密用户帐户和密码的 Key。

至于怎样进行加密形成最后的密文 PinBlock，有很多标准的，象 IBM3624、ANSI、ISO、DIEBOLD 等标准，其实它们大同小异，就是在对报文中的密码进行一个预处理，再用 PinKey 来 DES 加密，主要的差别就是怎样预处理而已，比如有的是密码后面补 F，补够 16 位，就是类似这样的预处理。

到这里我们应该理解 PinKey 和 PinBlock 了。通过 PinKey 和 MacKey 对报文进行了两重处理，基本

上报文就是安全的了。如果我们对 DES 算法比较了解，就会知道，如果想对加密后的密文解密，必须要知道 Key 才行，所以说 Key 一定要保密。怎样来保密 Key 呢？我们前面提到的无论是算 MAC 还是算 PIN 块，都是直接拿明文的 Key 来计算的，那么这个 Key 很容易被窃取的，比如有人在机器上装了个黑客程序，只要检测到你在用 Key 加密数据，就把明文的 Key 获取了。这个听起来好像挺玄乎的，不过是有这个可能性的，尤其是网上银行这些东东最容易中招了。

这样看来，我们还要对 PinKey 和 MacKey 本身进行加密，不要让人知道了。怎样实现，同样是 DES 算法大显身手的地方。我再找个 Key 对 PinKey 和 MacKey 进行一次加密，这样你就看不到 PinKey 和 MacKey 的明文了，好，解决问题了。这时用来对 PinKey 和 MacKey 进行加密的 Key 就被我们称为 MasterKey，即主密钥，用来加密其他密钥的密钥。不过，需要等一下，那 MasterKey 怎么办，它是明文啊。再找个 Key 来加密 MasterKey，那最终无论处理多少道，最后的那个 Key 肯定是明文，这样看来，安全的问题还没有解决啊。

既然此路不通，那我们需要换个思维角度了，仔细想想怎样处理明文的 MasterKey。黑客程序只能窃取我软件上的东西，如果我把 MasterKey 放到硬件里面怎么样，黑客是没能力跑到我硬件里面把 MasterKey 取出来的，当然，不排除道高一尺、魔高一丈的情况，但至少 99.9%的黑客都没这能力的。那这样不就解决了我们遇到的问题了吗，只要把 MasterKey 放到硬件里面（一般是键盘的加密模块里面）就好了。

好，到这里，我们已经不怕有人把报文中的关键信息获取到了，总算是安全了。

在最近，老是有人提到"硬加密"，这个有什么用呢？我上面不是已经解决了加密的问题了吗，还要这个概念干什么？看来我还是有些地方没考虑到。我一直想的是将明文的密码加密成密文，其中有个环节需要考虑下，明文的密码是怎样形成的，不就是我按键盘上面的数字形成的吗。以前我的软件处理是这样的，键盘每按一下，我就把那个数字在程序里面先存起来，等到 4 位或 6 位密码按完后，再把它们合在一起，再送给 PinKey 加密。那如果黑客程序直接把我的按键信息获取，那他根本不用破解报文中用 PinKey 加密后的密码，直接简单的就把我输入的密码得到了，我前面费尽心思对密码进行加密处理变得一点意义都没有了。

怎么办？如果我把获取按键的程序固化进入加密硬件（一般在键盘中），按键的数字根本不通过上层的软件，直接一步进入硬件里面处理，等到按键按完了后，硬件直接把经过一道处理的按键信息给我上层软件，此时已经是密文了，就相当于把前面计算 PinBlock 的处理移到硬件里面去了，那黑客就没法获取我的按键了。这种处理现在就被称为硬加密，伴随着 EMV 和 3DES 算法，变得越来越流行了，好像自助终端不支持硬加密就不行，连 EMV 也强制要求了。

最近还有个名词经常被提到，就是 3DES。为什么要提出 3DES 的概念呢？我在一篇文章中提到了 3DES 的具体算法，其实推出 3DES 是因为原来的单 DES 算法随着计算机硬件的速度提升，存在被破解的可能性，所以将算法进行了改进，改为 3DES 算法。但是对于我们理解金融行业的密钥及加密机制来说，用什么算法都一样。不同算法的差别只是怎样对数据进行移位变换等具体处理而已。
对于 ATM 交易安全性的考虑问题，系统通过 pin 加密，MAC 效验来保证系统交易数据的合法性及完整性,PIN BLOCK 产生，PIN 加密，MAC 效验都可在 ATM 的加密键盘进行。

以下简单解释概念：

1．工作密钥(WK)PIN Key：持卡人密码的加密传输(TPK,ZPK,PVK)

2．MAC Key：用于交易报文的鉴别，保证数据完整性(TAK, ZAK)

3．COM Key: 用于交易报文的通讯加密/解密(TEK,ZEK)

4．密钥交换密钥(KEK)Zone Master Key：节点间交换工作密钥时加密保护(ZMK)

5．Terminal Master Key：用于主机与金融终端交换工作密钥(TMK)

6．本地主密钥(LMK)Local Master Key：用于加密存储其它密钥

系统密钥的管理是保证整个系统交易安全的关键，三级密钥管理体系：

LMK(本地主密钥)　　最高层密钥，用于加密 TMK,ZMK

TMK(终端主密钥)，ZMK(区域主密钥)　　交换密钥，用于加密 PIN KEY MAC KEY,COM KEY

PIN KEY,MAC KEY,COM KEY　　PIN KEY 用于加密密码

工作密钥　　MAC KEY 用于效验报文

COM KEY 用于通讯加密