**1、关于3DES在平台中的分类:**

**1.1）平台中 原生3DES密钥的分类**

**1.1.1）会话协商的3DES密钥**

**保存位置:**

**客户端保存在内存中;**

**服务端保存在内容中;**

**生成时间：**

**会话时**

**生命周期：**

**会话**

**用途：**

**1.1.1.1）加密普通网络请求**

**1.1.1.2）加密密码键盘**

**组成：**

**客户端随机数：客户端的通用唯一识别码（UUID）**

**服务器随机数：服务端的通用唯一识别码（UUID）**

**服务器SessionID：**

**1.1.2）客户端使用UUID生成3DES密钥**

**保存位置：**

**本地加密的文件中；（加密使用的密钥是服务器下发的UUIDKEY混淆后的3DES密钥）**

**生成时间：**

**程序启动**

**生命周期：**

**从程序安装到卸载**

**用途：**

**1.1.2.1）本地存储加解密**

**1.1.3）客户端license密钥生成器**

**用途：**

**1.1.3.1）生成随机数，混淆html的文件名**

**1.1.3.2）生成随机数，混淆xml文件内容**

**1.1.3.3）生成3DES密钥，html文件解密**

**1.2）平台中 前置服务器 3DES密钥的分类**

**1.2.1）会话协商的3DES密钥**

**用途：**

**1.2.1.1）加密普通网络请求（等同于1.1.1.1）**

**1.2.1.2）加密密码键盘（等同于1.1.1.2）**

**1.3）平台中 打包工具 3DES密钥的分类**

**1.3.1）打包工具license密钥生成器**

**用途：**

**1.3.1.1）生成随机数，混淆html的文件名（等同于1.1.3.1）**

**1.3.1.2）生成随机数，混淆xml文件内容（等同于1.1.3.2）**

**1.3.1.3）生成3DES密钥，html文件加密（等同于1.1.3.3）**

**疑问：3DES在平台中的应用是否已经列举完整？**

**2、关于3DES在平台中的生成方式**

**2.1）针对 1.1.1 会话协商的3DES密钥**

**生成方式：**

**截取客户端随机数的3-11间的字节**

**截取服务器随机数的3-11间的字节**

**截取sessionId的删除最后1位的后8位的字节**

**作为3DES的密钥**

**2.2）本地存储加解密3DES密钥生成方式**

**生成方式：**

**a)服务器下发**UUIDKEY**的 字符串A(UUIDKEY在服务器段所有的用户使用同一个，并且这个UUIDKEY是固定不变的。目的是将b中的描述 )**

**b)获取 客户端UUID 字符串B(该字符串每次都变化,所以在程序第一次启动的时候，会使用 服务器的UUIDKEY 进行加密保存到本地)**

**c)对字符串A 分别截取0-7，8-15，16-23 作为 客户端UUID（字符串B）加解密的 3DES密钥，以下代码处理的具体操作是什么？**

- (NSString \*)getPrivateKey:(NSString \*)privateKey

{

int config[] = {77, 22, 88, 44, 55, 66};

NSMutableString \*buffer = [NSMutableString string];

int start = 0;

int end = 0;

for (int i = 0; i < 3; i++) {

start = config[i \* 2] % privateKey.length;

end = config[i \* 2 + 1] % privateKey.length;

if (start > end) {

start ^= end;

end ^= start;

start ^= end;

}

[buffer appendString:[privateKey substringWithRange:NSMakeRange(start, end - start)]];

}

while (buffer.length < 24) {

[buffer appendString:@“0”];

}

return buffer;

}

**d)将 客户端UUID（字符串B）通过3DES密钥加密保存到系统钥匙链中。（目的：为了保证下次程序启动的时候，使用同一个客户端UUID。）**

**e)对 客户端UUID（字符串B） 进行混淆处理得到 字符串C，**

**以下代码处理的具体操作是什么？**

- (NSString \*)getKey

{

if (key == nil) {

int config[] = {77, 22, 88, 44, 55, 66};

NSString \*bindKey = [self getBindKey];

if (bindKey != nil) {

NSMutableString \*buffer = [NSMutableString string];

[buffer setString:@“”];

int start = 0;

int end = 0;

for (int i = 0; i < 3; i++) {

start = config[i \* 2] % bindKey.length;

end = config[i \* 2 + 1] % bindKey.length;

if (start > end) {

start ^= end;

end ^= start;

start ^= end;

}

[buffer appendString:[bindKey substringWithRange:NSMakeRange(start, end - start)]];

}

while (buffer.length < 24) {

[buffer appendString:@“0”];

}

key = [NSString stringWithFormat:@“%@“, buffer];

}

}

return key;

}

**f)对 字符串C 分别截取0-7，8-15，16-23 间字节，**

**作为最终文件加解密使用的3DES密钥**

**疑问：最终使用的3DES密钥 是UUID，那么，将设备ID再使用服务器的UUIDKEY 加密保存到系统钥匙链中的意义是什么？**

**答：为了应用卸载之后能够再次使用。**

**2.3）针对 1.1.3.1）生成随机数，混淆html的文件名**

**生成方式：**

**a）使用 SHA1 算法计算字符串 html 的 hash值=Hash1**

**使用 SHA1 算法计算字符串 24 的 hash值=Hash2**

**使用 SHA1 算法计算文件ptframework\_license内容 的 hash值 = Hash3**

**b）对这三个hash值进行混淆处理，合并为一个 字符串A，**

**将 字符串A 追加到 html 文件名后面 获取 字符串B**

**c）对 字符串B 进行 MD5 计算，获取混淆后的html文件名**

**疑问：理解是否正确？**

**2.4）针对 1.1.3.2）生成随机数，混淆xml文件内容**

**生成方式：**

**a）使用 SHA1 算法计算字符串 xml 的 hash值=Hash1**

**使用 SHA1 算法计算字符串 24 的 hash值=Hash2**

**使用 SHA1 算法计算文件ptframework\_license内容 的 hash值 = Hash3**

**b）对这三个hash值进行混淆处理，合并为一个 字符串A，**

**将 字符串A 追加到 xml 文件内容后面 获取 字符串B**

**c）对 字符串B 进行 MD5 计算 获取 hash值，保存到 xml 节点，准备验签使用**

**疑问：理解是否正确？**

**2.5）针对 1.1.3.3）生成3DES密钥，html文件解密**

**生成方式：**

**a）使用 SHA1 算法计算字符串 html 的 hash值=Hash1**

**使用 SHA1 算法计算字符串 24 的 hash值=Hash2**

**使用 SHA1 算法计算文件ptframework\_license内容 的 hash值 = Hash3**

**b）对这三个hash值进行混淆处理，合并为一个 字符串A，**

**这个处理的具体操作是什么？**

- (NSData \*)getKeyWithName:(NSString \*)keyName keyLength:(int)l {

NSData \*data0 = [PTHash getHashByString:PTHASH\_TYPE\_SHA1 str:keyName];

NSData \*data1 = [PTHash getHashByString:PTHASH\_TYPE\_SHA1 str:[NSString stringWithFormat:@"%d",l]];

NSData \*data2 = [NSData dataWithData:licenceData];

int a = (int)[data0 length];

int b = (int)[data1 length];

int c = (int)[data2 length];

const Byte \*name = [data0 bytes];

const Byte \*mask = [data1 bytes];

const Byte \*data = [data2 bytes];

Byte \*ret = NULL;

size\_t bufferPtrSize = l;

ret = malloc(bufferPtrSize \* sizeof(Byte));

memset((void \*)ret, 0x00, bufferPtrSize);

Byte m,n,d;

int i = 0;

while (i < l) {

m = mask[i % b];

n = name[i % a];

d = data[i % c];

ret[i] = (m ^ n ^ d);

i++;

if (i % a == 0) {

data0 = [PTHash getHashByBytes:PTHASH\_TYPE\_SHA1 data:[NSData dataWithBytes:name length:a]];

a = (int)[data0 length];

name = [data0 bytes];

}

if (i % b == 0) {

data1 = [PTHash getHashByBytes:PTHASH\_TYPE\_SHA1 data:[NSData dataWithBytes:mask length:b]];

b = (int)[data1 length];

mask = [data1 bytes];

}

if (i % c == 0) {

data2 = [PTHash getHashByBytes:PTHASH\_TYPE\_SHA1 data:[NSData dataWithBytes:data length:c]];

c = (int)[data2 length];

data = [data2 bytes];

}

}

return [NSData dataWithBytes:ret length:l];

}

**c) 对 字符串C 分别截取0-7，8-15，16-23 间字节，**

**作为html文件加解密使用的3DES密钥**

**疑问：理解是否正确？**

**3、关于中信银行国密改造的时间用了多少人月？作为这次国密改造的参考时间。**

**苏鳞甲回复：**

**1、关于本地存储使用的密钥混淆方法，可以根据不同客户进行重写混淆方法。**

**2、如果本地存储密钥被破解，采用强制更新客户端的方式来弥补。**

**3、国密改造工期：**

**中信银行当前国密改造的工期4人月。**

**影响工期因素：**

1. **国密算法的稳定性。**
2. **对应平台的开发人员对源码的熟悉程度；**

**预计工期：6人月；**