基于智能卡的 RSA 数字签名实现关键问题解析

袁晓宇, 张其善
(北京航空航天大学, 电子信息工程学院, 北京 100083)

摘 要: 数字签名是一种应用于网络安全的重要安全机制, 智能卡或 Token 是用来实现数字签名验证的安全硬件载体, 如何在硬件载体上实现数字签名是一个较为关键的问题. 本文根据接触式智能卡系列标准及 PKCS (Public Key Cryptographic Standard) 系列相关标准, 成功实现了智能卡操作系统上的 RSA(一种非对称公钥密码算法) 算法下的数字签名、身份认证、信息加解密、密钥分配, 并着重解析了应用中智能卡 RSA 算法实现方面的关键问题, 提出了若干解决方案.

关键词: 智能卡; PKI; RSA; 数字签名
中图分类号: TP391 文献标识码: A 文章编号: 0372-2112 (2004) 11-1897-04

The Key Question Analysis of RSA Digital Signature Algorithm Based on Smart Card

YUAN Xiao-yu, ZHANG Qi-shan
(*School of Electronic Engineering, Beihang University, Beijing 100083, China*)

Abstract: Digital signature is an important security mechanism which is applied on the network security environment. The Smart card or Token is an optimal hardware to realize digital signature. How to implement digital signature on the Smart card or Token is a key issue. Based on contact smart card standards and series standards of PKCS(Public Key Cryptographic Standard), this paper implemented the RSA(a kind of public key asymmetric algorithm) digital signature、identification、message encryption、keys distribution, especially emphasized on analyzing some key questions about how to realize RSA on the Smart card practically, and present several method scheme.

Key words: smart card; PKI; RSA; digital signature

1 引言

随着网络应用的飞速发展, 以数字签名为基础的 PKI 公钥基础设施, 有效的保障了信息的安全, 并为数字化社会各种需要身份认证的实现与潜在业务提供支持. 当今, 国内外关于数字签名技术一直是研究热点, 不断变化更新的硬件支持推动了数字签名产品的不断发展, 本文采用 Infineon 公司的 SLE66CX320P 智能卡芯片[1], 成功开发了基于 RSA 算法的数字签名卡. 在进行签名认证实现时, 针对一些关键问题进行了深入分析, 并就解决实际问题提出了若干个解决方案, 成功的解决了实际应用中的难题, 达到产品更安全可靠, 性能更优化, 兼容性更好.

2 数字签名卡组成及规范

数字签名产品用于互联网上信息加密和身份认证, 它的构成大体分为两部分: 硬件——智能卡或 Token; 软件——智能卡中的操作系统及把智能卡或 Token 与互联网浏览器集成的并具有相关数字签名处理功能、实现在线应用的程序. 本文采用 Infineon 公司的 SLE66CX320P 智能卡芯片, 其携带高级加密协处理器的、支持 RSA 非对称密码算法[2], 密钥长度最大为 2048bits, 首先在芯片上实现智能卡操作系统, 进而实现数字签名功能. 智能卡的操作系统符合 ISO7816 1/2/3/4 规范及《中国金融集成电路(IC) 卡规范》, 数字签名的规范为 RSA 公司的 PKCS 系列相关标准, 支持 X.509v3 证书存储.

3 智能卡中 RSA 算法实现的关键问题

RSA 算法是基于大整数因子分解问题之上, 是非对称密码体制中的代表算法. 基于硬件令牌形式的 RSA 算法, 密钥保存更为安全可靠, 一旦公钥参数 (e, N) 和私钥参数 $(d, p, q, dp, dq, qinv)$ 产生后, 便分别安全的存放在智能卡的公钥文件和私钥文件中, 公钥文件设置有相应的读写权限, 私钥文件没有读取权限, 是不可以出卡的. 密钥对可以是一对或多对, 可设定为不同的用途. 基于智能卡的 RSA 数字签名实现, 需要注意的关键之处很多, 不然就会给安全上造成漏洞, 性能上造成缺陷, 以下就一些关键问题进行解析.

3.1 密钥长度的选取

在 RSA 算法上至今仍没有发现严重的安全漏洞, 但随着计算机的计算能力的提高及大数分解技术的发展, 增加密钥长度能大大增加破译的难度. 以下对密钥长度进行分析和选择.

3.1.1 公钥参数 N 长度的选择 就目前的密码分析技术看来, 若要保障 20 年以内的安全, 可以考虑密钥模数 N 长度为 1024bits; 若要求更高的安全性, 可以选择芯片所允许的 2048bits. 若 N 的长度选取的太小, 信息被破解的快, 密钥的生

收稿日期: 2004-01-15; 修回日期: 2004-06-06

命周期缩短. 若选择芯片协处理器所支持的 N 最大长度2048bits 时, 会限制智能卡的存储空间, 增加实际签名或解密的计算时间, 虽然随着科技的发展, 大容量、高性能的芯片不断的开发出来, 目前对存储空间和处理能力的任何增加都会使智能卡的成本增加. 综合考虑 N 的长度取为 1024bits, 另外密钥的模长还可以根据以后的要求而灵活的改动, 并易于系统的更新换代. 总之, 也就是说安全等级高的, 则公钥参数 N 的长度选取大些, 安全等级低的, 则选取相对小些的长度.

3.1.2 公钥参数 ***e*** 长度的选择 公钥 e 的长度由发卡方决定, 它的长度不能超过其对应 N 的模数长度的 1/4. 当 N 的长度为 1024bits 时, e 的长度不能超过 256bits, 另外, e 的长度较短时, 进行加解密的速度会加快, 降低存储公开密钥的空间. 参考所选择芯片的资料, 选取 e 的长度为 32bits[2], 满足 $16 \leq e$, $bitlength \leq 256$.

表 1 公钥参数的长度 (长度单位: bits)

公钥参数	e	N
模长度	32	1024

3.1.3 私钥参数 ***d*** 长度的选择 私钥 d 的长度应大于 N 的长度的 1/4, 实际应用中, d 的长度可以取的小一些, 以降低签名或解密的时间. 1990 年 Wiener 提出了一种针对 d 长度较小的攻击法, 他证明了若 d 的长度小于 N 长度的 1/4 时, 利用连分数算法, 可以在多项式时间内求出正确的 d. 由安全性及实际应用考虑, 一般仍是尽量降低 e 的长度, 而不降低 d 的长度, 本文选取 d 的模长度为 1024bits.

3.1.4 其他私钥参数长度的选择 其他私钥参数长度的选择受 N 的长度影响, 存在一定的数学关系[2], 其选取关系式如下:

$$P.bytelength = N.bitlength/8/2 + 1$$
$$Q.bytelength = N.bitlength/8/2 - 1$$
$$Dp.bytelength = N.bitlength/8/2 + 1$$
$$Dq.bytelength = N.bitlength/8/2 - 1$$
$$Qinv.bytelength = N.bitlength/8/2 + 1$$

其中 $N.bitlength$ 为公钥 N 的长度, $P.bytelength$, $Q.bytelenth$ 分别为生成 N 的两个大素数 p、q 的字节长度, $Dp.bytelength$, $Dq.bytelength$, $Qinv.bytelength$ 分别为公钥对中用于参与中国剩余定理算法而涉及的参数 dp、dq、$qinv$ 的字节长度, 如表 2 所示.

表 2 私钥参数的长度 (长度单位: bits)

参数	D	P	Q	Dp	Dq	$Qinv$
模长度	1024	520	504	520	504	520

3.2 密钥参数初始化

带有加密协处理器的芯片提供有 chipcard Cryto API, 其内嵌了相应源代码, 以满足用户的功能开发要求和安全要求, 使得编程接口简便易行. 核心程序是以 A51 汇编实现的, 以获得较好的代码效率. 算法的接口是以 C51 形式打包的, 易于开发者使用. 在接口中, 普遍采用的一种重要的变量类型 *pCLONG*, 定义如下:

```
typedef struct{
    unsigned int bitlength;
    unsigned char field[1];
} CLONG, * pCLONG ;
```

其中变量 *bitlength* 为密钥的模长度, *field* 为密钥值. 在产生公钥对时, 需要分别对公钥 (e, N), 私钥 $(d, p, q, dp, dq, qinv)$ 进行定义, 定义为 *pCLONG* 结构类型, 并初始化. 首先对 e 值大小初始化, 由于选择较低的 e 值, 可以加速加解密运算, 但是 e 太小时, 存在着低指数攻击的危险, 容易泄漏明文信息, 一般使用上 e 值多为 2、3 或 $2^{16}+1$. 这里 e 值初始化为 $2^{16}+1$, 即 *pCLONG* $e = \{0x00, 0x20, 0x00\ 0x01, 0x00, x01\}$, 共 32bits. 其中前两个字节 0x0020 为 e 的模长度, 后面的四个字节 0x00010001 为 e 的初始化值 $2^{16}+1$ 的十六进制. 如果 e 的初始值为 0, 则 Crypto API 函数将以随机数作为初始化值, 用于公钥对的产生, 这样易于产生不安全的因素; 如果 e 的初始值为任意不为零的值, 则以指定的值进行公钥对的产生. 其他的参数也都以这种形式定义, 前两个字节代表模长度, 后面的字节代表实际值. 如果对 dp、dq、$qinv$ 参数指定 *NULL*, 则 Crypto API 函数将不产生与中国剩余定理相关的参数, 即不使用中国剩余定理算法. 公钥对产生后, 把依次产生的公私钥参数存储到相应的公私钥文件当中.

3.3 充分利用协处理器

协处理器也叫高级加密引擎(ACE—Advanced Crypto Engine), 它本身就带一定的 RAM, 在算法上支持大数乘法, 支持中国剩余定理算法, 可以运行所有的模指数运算. 大多 ACE 所提供的功能有: 公钥对的产生, 加密解密操作、DES、RSA、散列值算法、DSA 算法. 在安全上, ACE 提供的 CPRYPTO API 含有相应的功能, 对攻击具有防范措施, 例如欠压保护电路, 时钟频率保护电路以及对 RSA 攻击的保护.

3.4 快速实现智能卡对读卡设备的应答

由于 RSA 算法中大数乘方取模困难, 其速度往往较慢. 而当智能卡在进行一些指令操作时, 要求其操作或交易都是在极为短暂的时间内完成的, RSA 算法有时不能满足智能卡与读卡设备之间的某些指令应答时间及功能的实现. 针对数字签名智能卡来说, 就是快速实现公钥对的产生, 快速实现加解密, 从而提高整个系统的性能, 保证一个完备功能产品的实现. 下面提出了几种可行方法, 并对此进行了分析和介绍.

3.4.1 采用 **CRT** CRT 即中国剩余定理, 又称孙子定理, 和 RSA 算法结合在智能卡中, 能够快速实现加解密计算. 在产生公钥对的过程中, 对类型为 *pCLONG* 的私钥参数 $(d, p, q, dp, dq, qinv)$ 分别赋零值, 形如 $p = \{0x04, 0x00, 0x00, 0x00, \cdots\cdots 0x000\}$, 前两个字节为模长度, 后面的字节为数据. 执行产生公钥对操作

$$rsakeygen(N, e, d, p, q, dp, dq, qinv)$$

便会自动给 p、q、dp、dq、$qinv$ 赋值. 如果初始化 p、q、dp、dq、$qinv$ 为 *NULL*, 这样将不会涉及 CRT 算法. 参数 dp、dq、$qinv$ 的具体含义为:

$$dp = d \bmod (p-1)$$
$$dq = d \bmod (q-1)$$
$$qinv = q^{-1} \bmod p$$

采用中国剩余定理的优越之处, 可以从表 3 中看出[3], 当采用 CRT 算法时, 无论是在 5MHz 还是在 15MHz 的频率下, 操作时

间都相应的大大减少了, 大约为原来的 1/4.

表 3　采用 CRT 和没有采用 CRT 时的对比列表

操作	模数长度 N	指数 e	执行时间(5MHz)	执行时间(15MHz)
Modular Exponentiation RSA Decrypt/RSA Sign Verify	1024bit	16bit	20ms	7ms
Modular Exponentiation RSA Decrypt/RSA Sign. Verify	1024bit	1024bit	820ms	273ms
Modular Exponentiation using CRT RSA Decrypt/RSA Sign. Generate	1024bit	1024bit	250ms	83ms

3.4.2　充分利用芯片中携带的 **PLL** 模块　为了提高芯片的运算速度, 当芯片以外部频率工作时, PLL(倍频)模块能使内部频率以数倍于原来的频率工作, 这样能在适当增大功耗的前提下大大减少算法的运算时间, 倍频前后的时间对比效果参见表 4.

表 4　倍频前后操作时间对比(N: **1024**bits; e: **32**bits)

操作	3.58MHz	10MHz
Generate Rsa key pair	15~50s	5~20s
Using private key (encrypt or decrypt)	435ms	156ms
Using public key(encrypt or decrypt)	55ms	20ms

在使用芯片的 PLL 模块时, 可以使芯片系统设定为最大值的倍频值, 以充分发挥芯片的性能. 重点是在进行某一具体的操作前, 打开倍频设置, 一旦操作在此倍频下进行完毕, 程序中应立即关掉倍频设置. 否则将会严重的扰乱在 $T=0$ 通信协议下, 以原来的频率进行串口操作的一些中断程序的运行, 或影响正常的读写 EEPROM 的操作, 或影响真随机数的产生, 代码实现如下:

```
{.... SlePllSwitch(0x28);//转换为最大倍频
Rsakeygen(N, e, d, p, q, dp, dq, qinv);//产生公钥对
SlePllSwitch(0x03);// 频率还原
.....}
```

3.4.3　返回特定的状态码　如果算法中已运用了中国剩余定理, 又充分的利用了 PLL 模块, 智能卡仍然不能正常操作, 或状态不理想, 那么可以参考以下两种方案:

方案 1: 根据 ISO/IEC7816-3 标准, 返回状态字节{0x60, XX}. 其中, 第一个状态码 0x60 通知读卡设备进行状态延迟, 第二个状态码字节 XX 通知读卡设备等待延时时间. 也就是说这一返回状态码使读卡设备保持卡原有的工作电压, 在 XX 时间内不发送信息, 也不接收信息.

方案 2: 当读卡设备不具有对返回状态码{0x60, XX}的识别设定时, 可以采取一种较为巧妙的方法. 当卡接收到一条指令进行相关的操作需要有较长时间, 而同时读卡设备又在等待卡的迅速应答, 卡可以在读写设备所要求的应答时间内返回一状态代码{0x90, 0x00}, 使读卡设备认为已得到正确应答, 从而使卡保持原有的工作电压, 正常的工作状态, 继续处理上一指令中未完成的工作. 至于上一次指令的成功与否, 可以另外设状态位, 以供下一次指令操作进行判断. 这一方法, 需要设计者设定两次操作的时间间隔, 以保证上一次操作有充分的时间完成任务.

3.4.4　设定密钥对　根据需要在智能卡芯片中事先生成或注入密钥对, 这也是一种可行的、巧妙的实现方法. 在应用当中, 提取的是已生成好的密钥对, 解决了由于产生公钥对的时间过长, 而使得卡对读卡器不能正常应答的难题. 当然由于事先生成的密钥对是存放在智能卡的芯片内的, 设定有权限, 安全性是完全可以保障的. 在微处理器芯片的性能和内存容量允许的情况下, 事先生成的密钥对可以是许多对, 当一个密钥对保密性已不再满足的情况下, 可以再次的选择其他的密钥对, 也可以选择新的密钥对用作其他的用途.

4　数据加解密及会话密钥的分配

选定密钥长度后, 进行加解密时还会遇到这样的情况: 一是在进行加密时, 如果待加密的文件长度大于密钥长度, 则要把明文分成块, 块的大小可变, 但不超过密钥的长度, 加密后把明文块转化为与密钥长度相同的密文块. 二是如果明文的长度小于密钥的长度, 例如明文长度小于密钥长度 128 字节时, 根据对 Openssl 的测试(Openssl 软件包提供了强大的功能与丰富的函数, 用于实现 SSL/TLS 协议和其相关的 PKI 标准)采取以下垫整规则对明文垫整:

00　01　FF　FF　.....FF　00+　明文数据

数据总长度保持 128bytes. 这样接收端能够准确地判断收到的信息, 使得在同一系统中应用不同的签名卡具有更好的兼容性.

会话密钥是对称密钥, 通信中利用对称密钥进行加解密的速度要比 RSA 算法快, 人们更倾向于选择 DES 算法, 而利用 RSA 算法实现会话密钥的分发, 则很好的解决了通信中既要保障安全性又要保障时间性的要求. 例如 64bits 的 DES 对称密钥按照上述方法垫整, 再用对方的公钥加密打包发送, 只有拥有对应私钥的对方才能打开得到 DES 密钥, 实现密钥分发. DES 密钥垫整形式如下:

00　01　FF　FF　.....FF　00(120bytes) + 会话密钥(8bytes)

5　数字签名

5.1　Hash 函数选择

Hash 函数, 也就是杂凑函数, 是把任意长度的输入消息串变换成固定长度的输出串的一种函数, 主要用于完整性校验, 杂凑函数的选择应该至少满足以下几个条件:

(1)输入长度是任意的;

(2)输出长度是固定的, 根据目前的计算技术应至少取 128bits, 以抵抗生日攻击;

(3)对每一个给定的输入, 计算输出是很容易的;

(4)给定杂凑函数的描述, 找到两个不同的输入消息杂凑到同一值是计算上不可行的.

一个安全的杂凑函数在数字签名实现中起着重要作用, SHA 是美国 NIST 和 NSA 设计的一种标准算法, 用于数字签名, 而 SHA-1 是在 90 年代不断完善的基于 SHA 之上的一种算法, 产生 160bits 的杂凑值, 具有较高的安全性, 关于散列值算法之间的比较可参考有关文献[4].

5.2　数字签名实现

由于公开密钥和私有密钥之间存在数学关系, 使用其中

一个密钥加密的数据仅且只能用另一个密钥解开. 发送者用自己的私有密钥加密数据传给接收者, 接收者用发送者的公钥解开数据后, 就可唯一的确定消息来自于谁, 保证了发送者对所发信息的不可抵赖性, 也就完完全全地模拟了现实生活中的签名. 数字签名实现中需要注意的有如下几方面.

5.2.1 数字签名中 HASH 值垫整 在对 20bytes 的 hash 值进行加密即签名的时候要遵循一定的规则[6] 垫整. 例如密钥长为 1024bits 时, 规则为: 固定数据 00 01 + *FF*... *FF*(90bytes) + 00 30 21 30 09 06 05 2*B* 0*E* 03 02 1*A* 05 00 04 14(16bytes 的规则值) + HASH 值(20bytes). 这和前述 4 中的加密垫整规则有所不同.

5.2.2 SHA-1 在卡内实现的数字签名 对外部发送来的信息, 利用卡内提供的 SHA-1 算法计算 Hash 值, 然后采用卡内的私钥对 Hash 值进行加密, 再把明文信息和加过密的 Hash 值打包一起送出卡外; 或者把加过密的 Hash 值送出卡外, 与明文信息打包在一起, 发送给收信方, 从而实现数字签名. 这样的好处是可以采用智能卡自身携带的 SHA-1 算法, 计算 hash 值方便易行. 不足之处, 鉴于读卡器与卡的通信速率是一定的(一般为 9600bps), 对大数据量的文件传送, 通信耗时较大.

5.2.3 SHA-1 在卡外实现的数字签名 在微机上利用 SHA-1 算法计算 Hash 值, 把得到 20 字节的散列值送到智能卡中, 然后采用智能卡中的私钥进行加密, 再把密文送出卡外, 在卡外与原有的明文一起打包发送给收信方, 以这样的方式实现的数字签名极大减少了读卡设备与智能卡之间的通信量, 能够以较快的速度实现数字签名.

6 数字签名的认证

接收方收到的数字签名组成是 Plain text + Encrypt(Hash), 通过把 Plain text 在卡内或在卡外进行 SHA-1 计算得到的散列值 Hash1, 与解密 Encrypt(Hash) 得到的散列值 Hash 相比较, 从而把认证过程分为两方法.

6.1 SHA-1 在卡内实现的签名认证

把数据 Plain text 送入智能卡内进行 SHA-1 计算, 这种方法如果 Plain text 过长, 则从主机到智能卡之间传输的数据量就较大, 可能会出现用一条指令不能把认证信息全部送入到智能卡中的现象. 但是如果 Plain text 的长度小于 128 字节, 好处是可以利用卡中的 SHA-1 函数方便的计算 plain text 的 Hash 值.

6.2 SHA-1 在卡外实现的签名认证

在卡外对 Plain text 进行 SHA-1 计算, 然后送入智能卡, 这里送入到卡内的数据 Hash1 + Encrypt(Hash) 长度只有 20bytes + 128bytes = 148bytes, 避免了过多的数据在主机和 IC 卡之间进行传送, 用一条指令就可以把认证信息送入到智能卡中, 减少了指令的操作时间, 同时也保障了认证的合理性.

7 签名认证实例

实例: 在计算机上采用 SHA-1 算法对信息进行单向散列值计算, 得到固定的 20 字节的数据, 然后送入 SLE66CX320P 智能卡芯片中, 按本文前述的 5.2.1 中的垫整规则垫整, 再用私钥进行数据加密即数字签名, 然后把加密数据送出卡外, 并结合原有的明文信息一起发送给收信方. 当收信方接收到数据, 用同样的 SHA-1 算法对信息明文进行散列值计算, 得到 Hash1, 再把 Hash1 + Encrypt(Hash) 送往智能卡中, 智能卡对 Encrypt(Hash) 用从对方证书中得到的公钥进行解密, 解密得到按 5.2.1 中垫整过的 128 字节的数据, 根据规则提取出 Hash 值. 把 Hash1 与 Hash 两个散列值进行比较, 就可以判断签名验证通过与否. 为保障证书的可靠, 对从 CA 下载证书的验证也采用同样的签名认证方法, 如果本级证书之上有多级证书, 验证过程要从根证书开始, 直至本级证书.

8 结束语

本文就基于智能卡的 RSA 数字签名的整个实现过程中存在的关键问题, 分别作了详细的分析, 并提出了若干方案, 成功解决了实际应用中的难题, 有效的提高了系统的性能, 另外基于 USBKey 形式的 RSA 数字签名卡, 直接与微机 USB 口连接, 也已成功开发应用. 随着数字签名卡技术的完善, 其应用的范围和形式也将越来越广阔, 数字签名技术的重要性逐步凸现, 可以说没有签名认证技术, 就没有现代社会的安全机制保障, 数字签名技术的普遍应用也一定大势所趋.

参考文献:

[1] Preliminary Confidential Data Book 12.00, Security & Chip Card Ics SLE66CxxxP[Z]. Infineon Technologies AG.

[2] Confidential Application Note 03.01, Application Programming Interface for Cryptographic Functions on the SLE66CxxxP[Z]. Infineon Technologies AG.

[3] 亿恒科技保密与智能卡芯片[Z]. 金卡工程, 2002, 1.

[4] 王育民, 刘建伟. 通信网的安全理论与技术(第一版)[M]. 西安: 西安电子科技大学出版社, 1999.4.

[5] 冯登国. 国内外密码学研究现状及发展趋势[J]. 通信学报, 2002, 23(5): 18-26.

[6] RSA Security Inc. PKCS # v2.1 RSA Cryptography Standard[S]. June 14. 2002.

[7] 赖溪松, 等. 计算机密码学及其应用[M]. 北京: 国防工业出版社, 2001.

作者简介:

袁晓宇 女, 1970 年出生于河南郑州, 现为北京航空航天大学电子信息工程学院博士研究生, 从事专业为通信与信息系统, 主要研究方向: 智能卡, 信息安全等.

张其善 男, 1936 年出生于浙江浦江, 北京航空航天大学教授, 博士生导师, 国家级有突出贡献的科技专家, 中国电子学会会士, 美国 IEEE 高级会员, 主要从事信息传输与处理, GPS 等方面的研究.