公开密钥密码体制 RSA 算法的实现和应用

吴永森　张芹

(华东工学院)

摘要　本文介绍公开密钥密码体制 RSA 算法的实现。该算法得到较快的加密、解密速率。还描述了所采用的优化算法，并介绍了 RSA 算法在 LAN-PABX 局域网络的数字签名机制和密钥管理机制中的实际应用。❶

关键词　密码技术　密码通信算法　数字签名　密钥管理

THE IMPLEMENTATION AND APPLICATION OF RSA ALGORITHM ON PUBLIC-KEY CRYPTOGRAPHIC SYSTEM

【Abstract】　RSA algorithm used for public-key cryptographic system yields a higher speed of encryption and decryption. RSA algorithm incorporated with some optimizing devices is introduced for digital signatures mechanism and key management mechanism in LAN-PABX local network.

【**Key words**】　cryptographic technique / cryptographic communication / algorithm / digital signature / key management

1976 年，Diffie 和 Hellman 发表了著名论文《密码学的新方向》，奠定了公开密钥密码体制的基础。公开密钥密码体制加密操作使用公开密钥，解密操作使用用户的私用密钥，从根本上克服了传统密钥密码体制存在的主要缺点。1978 年由 Rivest、Shamir 和 Adleman 三人提出了著名的 RSA 算法这是公开密钥密码体制中最优秀的加密算法，被认为是密码学发展史上第二个里程碑。RSA 算法的理论基础是数论中的一条重要论断：求两个大素数之积是容易的，而将一个具有大素数因子的合数进行分解却是困难的。RSA 算法计算复杂，实现的难度大。软件实现主要问题是加密、解密操作要计算位数达十进制百位以上的幂剩余函数，执行的时间长，难以满足实际使用要求。我们在实现中采取了一系列优化算法，使加密、解密速率大大加快，达到了实用要求。

1　RSA 算法的实现

1.1 RSA 算法的基本描述

1.1.1 密钥的求取

①选取两个随机大素数 p 和 q；

②求 $n=p\times q$，欧拉函数 $\varphi(n)=(p-1)\times(q-1)$；

③选取与 $\varphi(m)$互素的正整数 d，即满足 $gcd(d,\ \varphi(n))=1$；

④求 e，使满足 $e\times d=1\pmod{\varphi(n)}$

公布 e，n；而将 p、q、d 保密。e 为公开加密密钥；d 为秘密解密密钥。

1.1.2　加密与解密

将待加密的明文分块，块长为[0，$n-1$]。对每块明文 M 加密：密文 $C=M^e\pmod{n}$

对每块密文 C 解密，明文 $M=C^d\pmod{n}$。

❶收稿日期：1992-07-13

1.2 素数测试

1.2.1 素数测试算法

为求取密钥，首先需获得大素数，其位数为十进制数10～100位。迄今为止尚无一种理想的有效算法。一般采用概率判断算法。概率判断算法是这样一种算法：对被测数 n，当 n 为素数，它输出"n 为素数"；当 n 为合数，它输出"n 为素数"这一错误回答的概率不大于某值 ε。在文献[1]中 Rivest 等人推荐了 Solovay-Strassen 概率算法。这种算法每次测试判断出错的概率不大于 1／2。并建议测试判断 100 次。Atkin 和 Larson 在文献[2]中证明了 SPPT(Strong Pseudoprime Test)算法较之于 Solovay-Strassen 算法执行时间更快更有效。SPPT 算法每次测试判断出错的概率不大于 1／4。我们采用了 SPPT 概率算法进行大素数测试。设定测试次数为 12 次。这样，误判概率仅为 $(1/4)^{12}$。从实用角度衡量，素数测试的正确性已有足够的保证:

SPPT 算法如下:

①对于 n，$n=m\cdot 2^{n}+1$，m 为奇数，α 为正整数。在 $1<a<n-1$ 范围内，取一个随机数 α;

②计算 $a^{m}(\mathrm{mod}\ n)\Longrightarrow b$，如果 $b=\pm 1$，判定 n 为素数;

③置 $\beta=\alpha$;

④如果 $\beta=1$，判定 n 为合数;

⑤$\beta-1\Longrightarrow\beta$，$b^{2}(\mathrm{mod}\ n)\Longrightarrow b$;

⑥如果 $b=1$，判定 n 为合数;

如果 $b=-1$，判定 n 为素数;

⑦转至④。

1.2.2 小素数表

SPPT 算法测试素数的准确率高，但需进行大数的高次幂剩余运算以及多次平方剩余运算。因此时间开销大、效率低。由于大的合数通常含有小的素因子，故可在 SPPT 测试前，先用若干小素数进行筛选预处理，去掉相当一部份合数。筛选操作仅用到除法运算，时间开销小。这可显著地提高素数测试的效率。

由素数特性：除 2 以外的素数均为奇数；凡素数 p 必不能被小于 $\sqrt{p}$ 的素数所整除。故令 n 从 3 开始依次增 2 逐一搜索，判断 n 是否能被小于或等于 $\sqrt{n}$ 的奇素数整除。凡不能整除者为素数。以此构成一个由 500 个小素数组成的素数表。

1.3 强素数生成

幂剩余函数具有特殊的周期性：当依次求 M 的 e 次幂剩余时，在重复 h 次后，将会重新得到最初的明文 M。h 称之为周期。Simmous GJ 正是利用了幂剩余函数的周期性，于 1977 年破译了 MIT 公开密钥密码体制（即 RSA 体制）。为此，Rivest 等人在 1978 年正式发表 RSA 公开密钥密码体制的论文中，建议对素数 p、q 的选评满足三个条件：①p 和 q 在长度上相差几位；②$(p-1)$和$(q-1)$都应含有大的素数因子；③$\gcd(p-1, q-1)$应比较小。这样选择 p、q 实际上是使幂剩余函数的周期 h 足够大，使得 RSA 体制在实际上仍是不可破译的。

Gordon J 1984 年在文献[3]中提出了强素数概念。强素数可以抵抗破译者利用幂剩余函数的周期性进行的攻击。强素数 P 应满足如下四个条件:

①P 是一个位数足够长的随机选取素数

②$P-1$　含有一个大的素数因子 r

③$P+1$　含有一个大的素数因子 s

④$r-1$　含有一个大的素数因子 t

对强素数的这种定义，在 1986 年写入了 ISO-DP-9307。

如果 $P-1$ 有一个大的素数因子 r，那么 $P=jr+1$，j 为整数。由于 P、r 均为奇数(大的素数)，故 j 一定为偶数。这样，对于强素数 P，可以用下列三个等式表示:

①$P=2jr+1$，　或 $P=1\pmod{2r}$

②$P=2ks+1$，　或 $P=2s-1\pmod{2s}$

③$r=2lt+1$，　或 $r=1\pmod{2t}$

式中，j、k、l 为整数；r、s、t 为素数。

我们在实现 RSA 算法中 p、q 都使用强

素数。按以上分析，如下步骤来获得强素数：

(1) 选择两个指定长度的奇数 a、b。

(2) 利用前述的素数测试算法，在 a 附近产生随机素数 s；在 b 附近产生随机素数 t。

(3) 由 t 产生素数 r：

① $r:=1+2t$，若 r 为素数，即为所求 r；

② 若 r 不为素数，则 $r:=r+2t$。反复执行，直到 r 为素数，即为所求 r。

(4) 由 r、s 生成 P

① $P_0:=(S^{r-1}-r^{s-1}) \pmod{rs}$；

② 若 P_0 为偶数，$P_0:=P_0+rs$；

③ $P:=P_0+2rs$，若 P 为素数，即所求 P；

④ 若 P 不为素数，则 $P:=P+2rs$。反复执行，直到 P 为素数，即为所求 P。

1.4 密钥求取

1.4.1 加密密钥 e 和解密密钥 d 的求取次序

Rivest 等人在文献[1]中提出的 RSA 算法是在 d 与 $\varphi(N)$ 互素的条件下先选取 d，然后在 $e\times d=1(\mathrm{mod}\ \varphi(n))$ 条件下求得 e。这样所求得的加密密钥 e 的位数可能相当大，使加密操作时间很长。为了提高加密速率，我们不先选取 d，而是在 e 与 $\varphi(n)$ 互素的条件下先选取较小的 e，然后在 $e\times d=1\ (\mathrm{mod}\ \varphi(n))$ 条件下，求得 d。按此步骤，所求得的 d 也是满足文献[1]中要求的 d 与 $\varphi(n)$ 互素的条件的。

$\because e\times d=1 \quad (\mathrm{mod} \quad \varphi(n))$

即 $e\times d+k\times \varphi(n)=1$

$\therefore$ 存在整数 e，k

使 $\gcd(d,\ \varphi(n))$

1.4.2 扩展的欧几里德算法

使用欧几里德算法可求出给定的两个整数的最大公因数。若最大公因数为 1，则此两数互素。文献[4]对欧几里德算法进行了扩展。扩展的欧几里德算法不仅可以求取最大公因数，判断两数是否互素，还可以用于计算模的乘逆。这在求取解密密钥 d，以及在快速解密操作中求取常数 A 时，都将会用到。

在扩展的欧几里德算法中：

输入量 u、v 为非负整数

输出量为 u_1、u_2、u_3

$uu_1+vu_2=u_3=\gcd(u,v)$

引入辅助矢量 (v_1,v_2,v_3),(t_1,t_2,t_3)。

算法如下：

① 初始化。置 $(u_1,u_2,u_3)\leftarrow(1,0,u)$

$v_1,v_2,v_3\leftarrow(0,1,v)$

② 判断 $v_3=0$，若 $v_3=0$，结束

若 $v_3\neq0$，执行③

③ 置 $q\leftarrow\lfloor u_3/v_3\rfloor$

$(t_1,t_2,t_3)\leftarrow(u_1,u_2,u_3)-v_1,v_2,v_3\cdot q$

$(u_1,u_2,u_3)\leftarrow(v_1,v_2,v_3)$

$(v_1,v_2,v_3)\leftarrow(t_1,t_2,t_3)$

转②继续

1.5 高次幂剩余算法

加密操作 $C=M^e(\mathrm{mod}\ N)$ 就是求高次幂剩余。在解密操作和生成素数中也需求高次幂剩余。为了保证 RSA 算法的安全强度，模数 N 至少应取十进制 100 位以上。加密数据 M 分块与 N 位数相同，幂的次数也是较高的。因此，若按通常的先求乘幂后取余的算法，则计算的复杂性为指数型。这样大的计算量在一般微机上耗费的时间是很长的。我们采用了 Rivest 等人在文献[1]中提出的优化算法。这种优化算法的基本思想是在每次平方或相乘运算后，立即进行取余运算，以减少计算量。计算 $C=M^e(\mathrm{mod}\ N)$ 具体算法如下：

(1) 设 $e_ke_{k-1}\ldots e_1e_0$ 的二进制表示；

(2) 置变量 $C=1$；

(3) 对于 $i=k$，$k-1$，…，0，重复执行：

① $C:=C^2(\mathrm{mod}\ N)$；

② 若 $e_i=1$，则 $C:=C\times M\ \ (\mathrm{mod}\ N)$；

(4) 所得 C 值即为所求。

1.6 快速解密算法

由于解密密钥 d 甚大，若解密操作的乘幂取模运算也使用加密操作所用的算法实现，解密时间将会很长。为了缩短解密时间，Quisquater 等人在文献[5]中提出了一种快速解密算法。这种算法比经典的算法约快 4～8 倍。

若 C 为待解密的密文，$p<q$

设 $C_1 = C \pmod p$; $C_2 = C \pmod q$;

$d_1 = d(\bmod (p-1))$;

$d_2 = d(\bmod (q-1))$;

$m_1 = m(\bmod p) = c_1^{d1}(\bmod (p))$;

$m_2 = m(\bmod q) = c_2^{d2}(\bmod (q))$;

A 为常数: $A \times p \equiv 1(\bmod q)$

$0 < A < q-1$

由于解密者知道 p、q、d，所以上述参数都是可知的。解密操作是由中国剩余定时得到的:

明文 $m = [((m_2+q-m_1) \cdot A) \bmod q] \cdot p + m_1$

1.7 大数的基本运算

大数的基本运算是实现 RSA 算法的基础。其中，大数加法、大数减法和大数乘法都比较容易实现。这里只介绍大数除法运算。

大数除法(取余)运算，若按位处理，操作简单，但速率慢。合理方法应按字(16 位)处理，这样可以利用机器的除法指令。如何准确上商是实现大数除法的关键问题。我们用先求商的近似值，后进行修正得到商的准确值的优化算法。获得了较快的操作速率。

设　被除数 $u = u_0 u_1 \ldots u_n$

除数　$v = v_0 v_1 \ldots v_n$

商数 q 为 b 进制（现取为 $b = 2^{16}$）

设 q 的近似值为 $\hat{q}$,

令 $\hat{q} = \min(\lfloor (u_0 b + u_1) / v_1 \rfloor, b-1)$

当 $v_1 \geqslant \lfloor b/2 \rfloor$ 时，可以证明 $\hat{q}-2 \leqslant \hat{q} \leqslant q$ [4]。

由此可得到实现大数除法的算法如下:

(1) 标准化

置 $d \leftarrow \lfloor b/(v_1+1) \rfloor$

将被除数 u，除数 v 同时左移 d 位，得

$u = u_0 u_1 \ldots u_{m+n}$, $v = v_1 v_2 \ldots v_n$

使之满足: $v_1 \geqslant \lfloor b/2 \rfloor$

(2) 初始化

置 $j \leftarrow 0$, $u = u_j u_{j+1} \ldots u_{j+n}$

(3) 上商(近似值)，计算余数。

$\hat{q} = \min(\lfloor (u_j b + u_{j+1}) / v_1 \rfloor, b-1)$

$u_j u_{j+1} \ldots u_{j+n} \leftarrow u_j u_{j+1} \ldots u_{j+n} - v_1 v_2 \ldots v_n \times \hat{q}$

(4) 测试余数，修正近似商。

如果 $u_j u_{j+1} \ldots u_{j+n} > 0$，重复执行:

$\hat{q} \leftarrow \hat{q}+1$,

$u_j u_{j+1} \ldots u_{j+n} \leftarrow u_j u_{j+1} \ldots u_{j+n} - v_1 v_2 \ldots v_n$

如果 $u_j u_{j+1} \ldots u_{j+n} \leqslant 0$，顺序执行。

(5) 测试余数，上商（准确值)。

如果 $u_j u_{j+1} \ldots u_{j+n} < 0$

则 $q_j \leftarrow \hat{q}-1$

$u_j u_{j+1} \ldots u_{j+n} \leftarrow u_j u_{j+1} \ldots u_{j+n} + v_1 v_2 \ldots v_n$

如果 $u_j u_{j+1} \ldots u_{j+n} = 0$

则 $q_j \leftarrow \hat{q}$

(6) 测试循环次数

$j = j+1$，如果 $j \leqslant m$，转(3)执行

否则，顺序执行

(7) 结果

“余数” $u_{m+1} u_{m+2} \ldots u_{m+n}$ 右移 d 位，得到真正的余数 $u_{m+1} u_{m+2} \ldots u_{m+n}$

商数为: $q_0 q_1 \ldots q_m$

1.8 实现结果

使用 8086／8088 汇编语言编程。在 1.8MHz 的 IBM PC／XT 上，模数 N 选为 21.5 字，即二进制 344 位(十进制 104 位)。加密数据按 N 分块，也为十进制 104 位。

当加密密钥 e 取为 16 位时，加密时间仅为18 秒／块；解密时间约为 1 分钟／块。可见，加密速率已基本达到了实用要求。若选用主频较高的 386 档次的微机，使用软件方式实现 RSA 算法完全满足实用要求。

2 RSA 算法的应用

在高保密性能的 LAN-PABX 局域网络中，RSA 算法主要用于实现数字签名机制和密钥管理机制。有关数字签名机制和密钥管理机制的设计及安全性分析，将在文献[6]中论述。本文仅介绍 RSA 算法在实现这两种机制中的实际应用。

2.1 RSA 算法在数字签名中的应用

数字签名机制由数字签名程序模块实现，其实施过程由签名连接建立、签名传送和签名

连接释放三个阶段组成。RSA 算法应用于签名传送阶段。当签名连接建立后，收发双方进入数字签名通信环境，开始签名传送阶段:

签名发送方 A:

(1) 将待签名的信息 m，按签名协议格式进行封装，构成待签明文 M;

(2) 使用 A 的私钥 D_A，对 M 进行 RSA 解密变换 $D_A(M)=S_A$，获得 A 的签名文本 S_A;

(3) 使用 B 的公钥 E_B 对 S_A 进行 RSA 加密变换 $E_B(D_A(M))=C$，获得签名密文 C;

(4) 将 C 发送给接收方 B。

签名接收方 B:

(1) 收到 A 发来的签名密文 C 后，向 A 回答;

(2) 使用 B 的私钥 D_B 对 C 进行 RSA 解密换 $D_B(C)=D_A(M)=S_A$，获得 A 的签名文本 S_A;

(3) 使用 A 的公钥 E_B 对 S_A 进行 RSA 加密变换 $E_A(S_A)=E(D_A(M)) = M$，获得明文 M;

(4) 将 M 按签名协议格式进行拆卸，获得信息 m。

2.2 RSA 算法在密钥管理中的应用

传统的密钥密码体制（如数据加密标准 DEC 法算等)，加密和解密使用同一个秘密密钥。系统的密钥总数为用户总数的 n(n−1)／2 倍。随着用户的增加，密钥总数急剧增大，给密钥管理带来很大的困难。RSA 算法加密密钥是公开的，系统的密钥总数仅为用户总数的 2n 倍。这就大大地减小了密钥管理的难度。

采用主密钥和会话密钥两级组成的分层密钥结构。主密钥由系统管理员按 RSA 算法要求生成，由人工经保密途径分配。会话密钥（8 个字节）由发送方随机生成，使用 RSA 算法加密后，由密钥管理程序模块自动分配到接收方。会话密钥按 DES 算法（或快速加密算法）加密数据。主密钥更改周期由系统管理员确定；会话密钥“一次一密”，用后销毁。

系统可提供两类密钥管理服务：无鉴别功能的密钥管理和具有鉴别功能的密钥管理，供用户选用。具有鉴别功能的密钥管理，包括了以数字签名形式发送有关“证书”信息。

密钥管理机制由密钥管理程序模块实现。该安全模块调用了加密机制的各加密算法模块及数字签名模块，并包括有关的收发双方联络及通信软件。

3 结束语

以软件方式实现 RSA 算法的主要问题是操作时间太长，不能满足实用要求。我们在实现 RSA 算法中采用了一系列优化算法，使加密和解密速率大大加快，达到了实用要求。在“高保密性能的 LAN−PABX 局域网络”中得到实际应用。该项目已于 1990 年 11 月通过了江苏省科委的技术鉴定。

王梅和缪华星同志参加了部份工作。

参考文献

1 Rivest RL, Shamir A, Adleman LA. Method for Obtaining Digital Signatures and Public Key Cryptosystems. CACM. 1978, 21(2): 120− 126

2 Atkin AOL, Larson RG. On a Primality Test of Solovay and Strassen. SIAM J. COMPUT. 1982, 11(4): 789−791

3 Gordon J. Strong Primes are Easy to Find. Eurocrypt. 1984: 216− 223

4 Knuth DE, The Art of Computer Programming. Seminumerical Algorithms. 2 nd ed. 1981: 235− 239, 296− 307

5 Quisquater JJ, Couvreur C. Fast Decipherment Algorithm for RSA Public−Key Cryptosystem. Electronics Letters. 1982, 18(21): 905−907

(编辑 刁烈新)