文章编号:1001 - 2486(2002)05 - 0053 - 04

分块模幂算法*

倪谷炎
(国防科技大学理学院,湖南 长沙 410073)

摘 要:RSA 是目前最成熟的一种公钥密码体系,RSA 加解密算法的速度依赖于模幂算法,因而快速模幂算法一直是人们感兴趣的课题。该文提出了一种基于分块的模幂算法,对算法复杂性进行了讨论,并与其他模幂算法作了比较,从理论上论证了它具有更快的模幂速度。

关键词:信息安全;密码学;RSA;模幂算法
中图分类号:O156.2 文献标识码:A

A Powering Algorithm Based on Disparting Blocks

NI Gu-yan
(College of Science, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: RSA cryptosystem is the best one of the public key cryptosystems, and its enciphering or deciphering speed depends on a powering algorithm. This paper proposes a powering algorithm based on disparting blocks. By analyzing its complexity and comparing the complexity with other powering algorithms, it is proved that this is the fastest powering algorithm in Z/nZ.

Key words: information security; cryptography; RSA; powering algorithm

随着网络的迅猛发展,信息时代的到来,电子商务的介入,使我们对电子数据交换、电子邮件、电子资金转帐等数据传输的安全保密就更加关心。公钥密码技术是密码技术的核心,公钥密码体制是实现数字签名、身份论证和密钥分配的理想方法,而 RSA 算法是目前用得最多的公钥密码算法,可是其加密速度比较缓慢,从而影响它使用的广泛性,为此一些有关基于 RSA 加密的快速模幂算法的文章不断涌现,这是一个很有实际意义的研究课题。

1 关于 RSA 公钥密码系统

RSA 公钥密码系统是由 Rivest、Shamir 和 Adleman 三人根据 Diffie 和 Hellman 的公钥密码设想提出的一种数据加密体制,基础是数论的欧拉定理,它的安全性基于大整数分解的困难性。

1.1 RSA 加密算法

RSA 加密算法的过程是

1) 取两个素数 p 和 q(保密);

2) 计算 $n = pq$ (公开),$\varphi(n) = (p-1)(q-1)$(保密);

3) 随机选取整数 e(加密密钥),满足 $\gcd(e, \varphi(n)) = 1$(e 公开);

4) 计算整数 s(解密密钥),满足 $se \equiv 1 \pmod{\varphi(n)}$($s$ 保密)。

利用 RSA 加密第一步需将明文数字化,并取长度小于 $\log_2 n$ 位的数字 m 作明文块。

加密算法: 加密后得到密文 $c \equiv m^e \pmod{n}$

解密算法: 解密后得到明文 $m \equiv c^s \pmod{n}$

* 收稿日期:2002 - 04 - 02
作者简介:倪谷炎(1966—),男,讲师,博士生。

1.2 RSA安全性

若整数 $n(=pq)$ 被分解,则 p 和 q 成为已知,$\varphi(n)=(p-1)(q-1)$ 也就可以算出,进一步,根据同余式

$$se\equiv 1(\bmod\ \varphi(n))$$

算出解密密钥 s,于是RSA被击破。

但还不能证明对RSA攻击的难度就相当于对整数 n 分解的难度。为了安全起见,对素数 p 和 q 还有要求:

a) 素数 p 和 q 长度相差不大;

b) $p-1$ 和 $q-1$ 有大素数因子;

c) $\gcd(p-1,q-1)$ 很小。

2 模幂算法

RSA加密和解密都要进行 $\bmod\ n$ 的幂运算[1],即求 $m^e(\bmod\ n)$ 运算。由于RSA公钥密码体制中选取的整数 n 一般是十进制的200位以上,加解密运算的速度当然缓慢,因而成为RSA加解密的瓶颈技术,于是有关RSA快速加密算法文章不断出现。下面介绍几个用于加密的模幂算法。

2.1 传统模幂算法

求 $m^e(\bmod\ N)$,

1) 把 e 表示为二进制,即 $e=\sum_{i=0}^{n-1}e_i2^i$, $e_i\in\{0,1\}$

2) 对明文 m 进行加密运算过程如下

$$c\equiv((\cdots((1\cdot m^{e_{n-1}})_N^2\cdot m^{e_{n-2}})_N^2\cdots m^{e_1})_N^2\cdot m^{e_0})_N \tag{1}$$

$(\cdot)_N^2$ 表示先做平方运算,再做模 N 运算。该算法共做了 $n+h(e)-2$ 次模 N 乘法运算,其中 $h(e)$ 表示 $e_0,e_1,\cdots,e_{n-1}$ 中不为零的个数。

2.2 SMM模幂算法[2]

运用(1)式进行加密迭代,假设每一次迭代后的中间结果为 A,当 $A,m\leqslant\frac{N-1}{2}$ 时,下一步迭代计算方法不变;当 $A,m>\frac{N-1}{2}$ 时,令 $i=N-A$,$j=N-m$,应用下面公式

$$i^2=A^2(\bmod\ N)$$

$$ij=Am(\bmod\ N)$$

$$-ij\equiv A(N-m)\equiv m(N-A)(\bmod\ N)$$

进行计算。

2.3 ISMM模幂算法[2]

1) 把 e 表示为冗余二进制,即 $e=\sum_{i=0}^{n-1}e_i2^i$, $e_i\in\{\bar{1},0,1\}$,当幂剩余运算的指数 e 中有三个以上连“1”时,将 e 表示冗余二进制

$$e=\sum_{i=1}^{n-1}p_i2^i,\ p_i\in\{\bar{1},0,1\}$$

式中的‘$\bar{1}$’表示‘-1’,例如‘1111’可以表示为‘$100\bar{1}$’。

2) 对每一个密文 m 计算 $m(\bmod\ N)$ 的逆 m^{-1}。

3) 从 e 的冗余二进制的高位起,对(1)式用SMM算法进行计算。遇到‘1’和‘0’的位,计算方法不变;遇到‘$\bar{1}$’的位就用 m^{-1} 去乘上步的中间结果。

该算法共用了1次求 $m(\bmod\ N)$ 的逆 m^{-1} 运算和 $n+a_1+2a_2-2$ 次模 N 乘法运算。其中 a_1 表示 e 的二进制中单独(非连续)1的个数,a_2 表示 e 的二进制中有二个以上连续1的个数。

3 分块模幂算法

分块模幂思想 :首先把加密幂钥 e 表示为二进制,并按连续的‘1’和连续的‘0’进行分块 ;其次计算 $a^e \pmod N$ 时把 e 的二进制中连续的 1 作为一个整体来进行计算。

例如 $e = <1100100111011>_2 = \underline{11}\ \ \underline{00}\ \ \underline{1}\ \ \underline{00}\ \ \underline{111}\ \ \underline{0}\ \ \underline{11}$,则

$$a^e \pmod N = (((a^{\underline{11}})^8 \cdot a^{\underline{1}})^{32} \cdot a^{\underline{111}})^8 \cdot a^{\underline{11}} \qquad (\bmod N)$$

在计算前,预先算出 $a^{\underline{1}}$,$a^{\underline{11}}$和 $a^{\underline{111}} \pmod N$。

为对新算法作一般性的描述,把 e 的二进制形式采用如下新的表示形式,记

$$e = \{(u_1, v_1), (u_2, v_2), \cdots, (u_L, v_L)\}$$

其中,$u_1, u_2, \cdots, u_L$ 和 $v_1, v_2, \cdots, v_L$ 分别为 :从 e 的二进制高位往下数,

u_1 表示第一次出现连续‘1’的个数 ;

v_1 表示第一次出现连续‘0’的个数 ;

u_2 表示第二次出现连续‘1’的个数 ;

v_2 表示第二次出现连续‘0’的个数 ;

依此类推,因为 e 是奇数,规定 $v_L = 0$。例如 $= e = <1100100111011>_2 = \{(2,2),(1,2),(3,1),(2,0)\}$。

于是对一般的 e, 假设 $a_1, a_2, \cdots, a_L$,分别表示相应连续 1 的位置的幂 $a^{11\cdots1}$, 那么

$$a^e = (\cdots((a_1)^{2^{v_1+u_2}} \cdot a_2)^{2^{v_2+u_3}} \cdots)^{2^{v_{L-1}+u_L}} \cdot a_L$$

基于这种思想, 结合 SMM 模幂算法[2], 下面给出分块模幂算法。

分块模幂算法 :

Input :$l = \max\{u_1, u_2, \cdots, u_L\}$, m, N

Output :$c \equiv m^e \pmod N$

Step 1(预处理)计算 $a_1, a_2, \cdots, a_l$,使 $a_i < N/2$,且 $a_i \equiv \pm m^{2^i - 1} \pmod N$。如果 $a_{u_L} \equiv -m^{2^{u_L}-1} \pmod N$,记 sign = 1。

$a_1 = m$;

for($i = 2$;$i <= l$;$i++$) $a_i \equiv a_{i-1}^2 \cdot m \pmod N$;

if($a_L > N/2$) sign = 1 ;

else sign = 0 ;

for($i = 1$;$i <= l$;$i++$) if($a_i > N/2$) $a_i = N - a_i$

Step 2(模幂计算)

$c = a_{u_1}$;

for($i = 2$;$i <= L$;$i++$) {

if($c > N/2$) $c = N - c$;

$c \equiv c^{2^{u_i+v_{i-1}}} \cdot a_{u_i} \pmod N$;

} ;

if(sign = = 1) $c = N - c$;

算法证明 :1)先考虑从第 $i-1$ 步到第 i 步的加密过程。假设 c_i, c_{i-1}分别是加密到第 i 和 $i-1$ 步的阶段性密文,即

$$c_{i-1} = m^{\{(u_1,v_1),\cdots,(u_{i-1},0)\}} \pmod N, \quad c_i = m^{\{(u_1,v_1),\cdots,(u_{i-1},v_{i-1}),(u_i,0)\}} \pmod N$$

应用传统的模幂算法

$$c_i \equiv ((\cdots(((c_{i-1})^{2^{v_{i-1}}})^2 \cdot m)^2 \cdots)^2 \cdot m)^2 \cdot m \pmod N \quad (\text{共有 } u_i \text{ 个 } m)$$

$$\equiv (c_{i-1})^{2^{v_{i-1}+u_i}} \cdot m^{2^{u_i}-1} \pmod N$$

$$\equiv \pm (c_{i-1})^{2^{v_{i-1}+u_i}} \cdot a_{u_i} (\mathrm{mod}\ N)$$

其中“ ± ”是由于取 $a_i < N/2$ 导致的。

2)由于

$$(-k)^2 \equiv k^2 (\mathrm{mod}\ N)$$

$$c^2 \cdot (-a) \equiv N - c^2 a (\mathrm{mod}\ N)$$

所以,由第一个同余式知道在加密算法的前 $L-1$ 步因取 $a_{u_i} < N/2$ 和 $c_i < N/2$ 而导致“ ± ”的出现,但不会影响密文 c ;而到第 L 步,由第二个同余式知道“ if(sign = = 1) $c = N - c$ ”是不可缺少的。所以该算法是完全正确的。

4 算法的复杂性分析

由于在 RSA 公钥密码体制中,为确保加密的安全,通常都把 N 和 e 取得相当大,因此在算法复杂性分析上,主要是看它的模 N 乘法的次数。该算法的 Step1 用了 $2l-2$ 次的模 N 乘法;对 Step2 的 for()循环中每个 i 做 $u_i + v_{i-1} + 1$ 次模 N 乘法,因此 Step2 共用了

$$\sum_{i=2}^{L} (u_i + v_{i-1} + 1) = n - u_1 + L - 1$$

次模 N 乘法。总共做模 N 乘法运算 $n + L + 2l - u_1 - 3$ 次。

5 算法复杂性比较

该算法与传统的算法相比较少做了

$$h(e) - L - 2l + u_1 + 1 \approx h(e) - L, (\text{因为通常 } h(e) \gg 2l + u_1)$$

次模 N 乘法运算。由于每个信息单元都少做这么多次的模 N 乘法运算,那么在对一个文件加密时就能节约很多的时间。另外,由文献[2]知,取 $a_i < N/2$ 和 $c_i < N/2$ 又能节约不少时间。所以该算法同传统的算法相比所花的时间会更少。

例如,选取 e ,其二进制长度 $n = 200$, $h(e) = 100$, $L = 20$, $l = 7$, $u_1 = 3$,新算法模 N 乘法运算次数 $\mathrm{New} = 200 + 20 + 2 \times 7 - 3 - 3 = 228$;而用传统模幂算法的模 N 乘法次数 $\mathrm{Old} = 200 + 100 - 2 = 298$。$\mathrm{New}/\mathrm{Old} = 228/298 = 76.5\%$,即仅模 N 乘法一项就节约了近 1/4 的时间。

与文献[2]的算法相比较:1)文献[2]的算法在做预处理时先求出 $m (\mathrm{mod}\ N)$ 的逆元,由于 N 比较大,所以求出 $m (\mathrm{mod}\ N)$ 的逆将花去很多的时间;2)文献[2]模 N 乘法运算为 $n + a_1 + 2a_2 - 2 = n + L + a_2 - 2$ 次,比本文的算法多 $(a_2 - 2 - (2l - u_1 - 3)) = a_2 + u_1 + 1 - 2l$ 次。

因此由以上两条,本文算法每对一个信息单元加密将节省更多的时间。

参考文献:

[1] Cohen H. A Course in Computational Algebraic Number Theory[M]. 3rd edition, Beijing World Publishing Corporation, 1997.

[2] 陈运,龚耀寰. RSA 快速算法研究[J]. 通信保密,2000,(3).

[3] Chen Yun. An Improved SMM Algorithm[J]. Journal of Electronics, 1999, 16(1).

[4] 卢开澄. 计算机密码学(第二版)[M]. 北京 清华大学出版社,1998.

[5] Elkenbracht-Huizing R M. Factoring Integers with the Number Field Sieve[D]. 1997.

[6] Lenstra JR H W. Computational Methods in Number Theory (Part 1)[M]. 2ed edition,1984.