秦九韶算法思想在RSA密码算法中的应用研究

董付国[1],厉玉蓉[1,2]
DONG Fu-guo[1],LI Yu-rong[1,2]

1.山东工商学院 信息与电子工程学院,山东 烟台 264005
2.山东大学 计算机科学与技术学院,济南 250100
1.School of Information and Electronics Engineering,Shandong Institute of Business and Technology,Yantai,Shandong 264005,China
2.School of Computer Science and Technology,Shandong University,Jinan 250100,China

DONG Fu-guo,LI Yu-rong.Study on QinJiuShao algorithm and its application in RSA.Computer Engineering and Applications,2008,44(28):65-66.

Abstract: This paper studies QinJiuShao algorithm for fast computation of high-rank polynomial,analyses fast algorithm of modular exponentiation in RSA,and gives its implementation.Analysis and experiment results show that computation quantity of this algorithm does not get too larger with exponentiation getting larger very fast,and may reduce when exponentiation well-selected.Computation of modular exponentiation is very low because of big plain-text group and cipher-text group,so this algorithm is of great significance to design fast algorithm of RSA and select public-key and private-key of RSA.
Key words: QinJiuShao algorithm;RSA;fast algorithm;modular exponentiation

摘 要:介绍了用于快速计算高次多项式值的“秦九韶算法”,并用类似思路分析了RSA算法中方幂模快速实现算法,最后给出了该算法的具体实现。算法分析和实验结果证明,该算法的计算量不会随着指数的快速增大而增大,通过精心选择指数,还可以减少运算量。RSA算法中明文分组和密文分组都较大,方幂模运算消耗大量的运算时间。因此,简化方幂模计算减少计算次数对设计RSA快速算法和选择密钥具有重要的指导意义。
关键词:秦九韶算法;RSA;快速算法;方幂模
DOI:10.3778/j.issn.1002-8331.2008.28.023 **文章编号:**1002-8331(2008)28-0065-02 **文献标识码:**A **中图分类号:**TP309

1 引言

随着计算机技术的飞速发展,信息安全问题越来越严重,几乎所有领域中都用到加密/解密的技术。密码学算法主要有对称密钥密码体制和非对称密钥密码体制两种。前者适合大量明文加密,并且具有较高的处理效率,例如用硬件实现的DES算法比RSA快1 000倍,软件实现的DES比RSA快100倍,但对称密钥密码体制中加密密钥和解密密钥相同或可互相快速推导,比非对称密钥密码体制算法需要更多的密钥,且无法保证密钥的分发和传输等环节的安全性,其安全性主要取决于密钥的保密性;后者虽速度相对较低,但加密密钥和解密密钥不相同甚至无法在有限的时间内进行互相推导,不仅可以更安全地传输数据,并且在数字签名算法中已经得到广泛应用。结合两种体制算法设计加密/解密系统,使用对称密码算法进行大量数据加密/解密,而使用非对称密码算法进行对称密码算法密钥的传输,是目前大多数系统的设计思路。目前对称密钥体制算法中Rijndael算法比较成熟并得到广泛应用,而非对称密钥体制中RSA仍是主流算法。RSA算法要求密钥长度不低于1 024 bit,且分组长度较大,加密和解密运算过程涉及到大数的方幂模运算,严重影响了处理速度[1-8]。本文主要介绍秦九韶算法原理及其在RSA快速算法中的应用,实验结果证明,在不影响RSA算法安全性的前提下,大幅度加快了处理速度。

2 RSA算法原理

RSA算法属于非对称密钥密码体制,基于成熟的数论理论,使用人持有私钥,而对应的公钥可以公开,其安全性取决于大数进行素数分解的数学难题。RSA算法过程如下:

步骤1 选择长度一样、距离较大且数值足够大的两个强素数p、q;

步骤2 计算公开模数$n=p\times q$;

步骤3 计算$\varphi(n)=(p-1)\times(q-1)$;

步骤4 选择大于1且小于$\varphi(n)$并满足条件$\gcd(\varphi(n),e)=1$的整数e;

步骤5 计算d,使得$d\times e=1(\bmod \varphi(n))$;

步骤6 销毁p、q。

此时,得到公开密钥对$\{e,n\}$和私钥对$\{d,n\}$。若x为需要加密的明文,则相应的密文y通过下面的加密变换得到:

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60673153,No.60773053);山东省自然科学基金(the Natural Science Foundation of Shandong Province of China under Grant No.Y2005G09,No.Y2007A28)。
作者简介:董付国(1977-),男,讲师,研究方向为虚拟现实、信息安全;厉玉蓉(1975-),女,博士后,副教授,研究方向为数学分析、计算机图形学。
收稿日期:2007-11-19 **修回日期:**2008-02-01

$y=E_{\{e,n\}}(x)=x^e \bmod n$

若已知密文 y 和相应的解密密钥对$\{d,n\}$,则相应的明文 x 通过下面的解密变换得到:

$x=E_{\{d,n\}}(y)=y^d \bmod n$

非法接收者在仅知道公开密钥对$\{e,n\}$的情况下,若想得到解密密钥对$\{d,n\}$,除了穷举攻击,必须对数 n 进行素因子分解从而得到 p、q 进而得到 $\varphi(n)$,才可能得到 d,而大整数的素因子分解是个数学难题,从而保证了系统的安全性[1-8]。

3 秦九韶算法原理

中国宋代秦九韶算法最初为了快速计算高次多项式的值,如要计算多项式 $f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ 的值,传统算法是对每项进行计算然后多项累加,而用秦九韶算法[9-11]将多项式改造为 $f(x)=(((\cdots(a_n)x+a_{n-1})x+\cdots+a_1)x+a_0$ 后大幅度减少了乘法的次数,从而加快了计算速度。一般多项式值计算的算法描述为:

$$\begin{cases}S_n=a_n\\S_k=x\times S_{k+1}+a_k,k=n-1,n-2,\cdots,1,0\\P_n(x)=S_0\end{cases}$$

借用该算法思想可以简化方幂模计算,如计算 m^e,将指数 e 转换为二进制数得到$(e_ne_{n-1}\cdots e_2e_1e_0)_B$,其中 $e_k\in\{0,1\},k=n,n-1,\cdots,2,1,0$,则计算过程为:

$$m^e=m^{e_n\times2^n+e_{n-1}\times2^{n-1}+\cdots+e_2\times2^2+e_1\times2^1+e_0\times2^0}=$$

$$m^{e_n\times2^n}\times m^{e_{n-1}\times2^{n-1}}\times\cdots\times m^{e_2\times2^2}\times m^{e_1\times2^1}\times m^{e_0\times2^0}=$$

$$((\cdots((m^{e_n})^2\times m^{e_{n-1}})^2\times\cdots\times m^{e_2})^2\times m^{e_1})^2\times m^{e_0}$$

算法描述为:

$$\begin{cases}S_n=m^{e_n}\\S_k=(S_{k+1})^2\times m^{e_k},k=n-1,n-2,\cdots,1,0\\P_n(m)=S_0\end{cases}$$

如计算 m^{255},255 对应的二进制数为 11111111,计算过程如下:

$$m^{255}=m^{128+64+32+16+8+4+2+1}=m^{128}\times m^{64}\times m^{32}\times m^{16}\times m^8\times m^4\times m^2\times m=$$

$$(((((((m)^2\times m)^2\times m)^2\times m)^2\times m)^2\times m)^2\times m)^2\times m$$

相比逐个相乘法的 254 次乘法,秦九韶算法只用了 14 次乘法,计算 m^{108},108 对应的二进制数为 1101100,计算过程如下:

$$m^{108}=m^{64+32+0+8+4+0+0}=m^{64}\times m^{32}\times m^0\times m^8\times m^4\times m^0\times m^0=$$

$$((((((m)^2\times m)^2\times 1)^2\times m)^2\times m)^2\times 1)^2\times 1=$$

$$((((((m)^2\times m)^2)^2\times m)^2\times m)^2)^2$$

相比逐个相乘法的 107 次乘法,秦九韶算法只用了 12 次,优化后只用了 9 次,计算 m^{128},128 对应的二进制数为 10000000,计算过程如下:

$$m^{128}=m^{128+0+0+0+0+0+0+0}=m^{128}\times m^0\times m^0\times m^0\times m^0\times m^0\times m^0\times m^0=$$

$$((((((m)^2\times 1)^2\times 1)^2\times 1)^2\times 1)^2\times 1)^2\times 1=$$

$$((((((m)^2)^2)^2)^2)^2)^2)^2$$

相比逐个相乘法的 127 次乘法,秦九韶算法只用了 14 次乘法,优化后只用了 7 次乘法。

可以看出,若指数 e 的二进制位数为 L,秦九韶算法最坏情况(e 对应二进制数为全 1)下需要计算 $2(L-1)$次乘法,如果 e 对应的二进制数中有大量零存在,还可以减少乘法计算次数,最好情况(e 对应二进制数中只有 1 位为 1,其余位均为 0)下只需要计算 $L-1$ 次乘法,将复杂度 $O(n)$降低到 $O(\log n)$,大幅度简化计算过程,加快了计算速度。

4 秦九韶算法在 RSA 算法中的应用及实现

在 RSA 算法中,$m^e \bmod n$ 计算中由于运算数值较大,不仅容易超出计算机的数值计算范围,并且消耗大量的计算机时间,是影响 RSA 算法效率的主要因素之一。借用秦九韶算法的思想,将指数 e 转换为二进制数再计算,则可大幅度减少乘法的计算次数,提高整个算法的处理速度。

用秦九韶算法快速计算方幂模的核心算法为:将指数 e 转换为二进制数,用 e_i 表示二进制数中第 i 位值,L 为二进制数的长度,则令 $c=1,i=L-1$,有

$$\begin{cases}S_n=m^{e_n}\\S_k=((S_{k+1})^2\times m^{e_k})\bmod n\\P_n(m)=S_0\end{cases}$$

其中 $k=n-1,n-2,\cdots,1,0$,算法描述如下:

Step1 $c=1,i=L-1$

Step2 while(i>=0)do step3 to step5

Step3 $c=(c\times c)\bmod n$

Step4 if $e_i=1$,then

$c=(c\times m)\bmod n$

Step5 $i=i-1$

算法结束时,c 即为所求。算法中,对中间结果提前进行取模运算,有效防止了中间数过大而超出数值计算范围,并借用秦九韶算法思想大幅度简化了模幂的计算过程。

5 实验分析与结论

实验中,随机选择大指数进行分析,实验数据如表 1 所示。

表 1 几种算法中乘法计算次数比较

十进制指数	对应二进制数	直接计算需要的乘法次数	利用秦九韶算法思想需要的乘法次数
32 323	1111111001000011	32 322	22
32 666	1111111110011010	32 665	24
65 537	10000000000000001	65 536	17
33 332 666	1111111001001110110111010	33 332 665	40
656 312 692	10011100011110100010010111010 0	656 312 691	43
656 277 504	10011100011110000000000000000	656 277 503	36

算法分析以及实验结果证明,利用秦九韶算法思想设计的算法中乘法次数只取决于指数对应的二进制数位数及二进制数中的 1 的个数,随着指数的快速增大,乘法次数增加较慢,通过精心选择结构合理的指数,甚至会大幅度减少。算法的这个特点对于设计 RSA 的快速算法有着重要意义,对 RSA 算法中密钥选择也具有实际的指导意义。

参考文献:

[1] 步山岳,张有东.计算机信息安全技术[M].北京:高等教育出版社,2006.

[2] Diffie W,Hellman M E.New directions in cryptography[J].IEEE Transactions on Information Theory,1976,IT-22(6):644-654.

(下转 78 页)