第33卷 第20期 计 算 机 工 程 2007年10月
Vol.33 No.20 Computer Engineering October 2007
·安全技术· 文章编号：1000—3428(2007)20—0148—02 文献标识码：A 中图分类号：TP309

RSA密钥对高效生成算法

姚国祥，林良超
(暨南大学信息科学技术学院，广州 510632)

摘 要：RSA是公钥密码体系中十分重要的加解密算法，RSA的效率瓶颈主要在大素数的寻找和指数模幂运算上。RSA密钥对的生成过程直接地涉及以上两大瓶颈计算问题。该文分析了RSA密钥对生成过程中涉及到的各种算法，并且通过修改随机数的生成方法来达到进一步改进预筛选算法的目的。

关键词：RSA算法；大素数寻找；指数模幂运算；密钥对

Efficient Method of RSA Key-Pair Generation

YAO Guo-xiang, LIN Liang-chao
(College of Information Science and Technology, Jinan University, Guangzhou 510632)

【Abstract】RSA is a very important public-key encryption algorithm. The bottleneck problems of RSA efficiency are big prime number finding and exponential modular computation. RSA key-pair generation depends on the two problems mentioned just now. This paper carefully analyse the main algorithms used by the key-pair generation process, and improves the preselection algorithm by modifying the method of random number generation.

【Key words】RSA algorithm; big prime number finding; exponential modular computation; key-pair

RSA是公钥密码体制中很常用的一个算法[1]，目前在计算机等多个领域都得到广泛的使用，很多通用的安全协议都是基于RSA算法实现的。使用RSA完成各种服务首先要生成长度足够的密钥，RSA密钥生成需要一定的数学理论基础，并且实现起来方法有很多种，寻找高效的实现方法很值得研究[2]。

RSA密钥的生成算法主要是大素数找寻问题，其中又涉及到大整数的内存表示、随机数生成、素性测定、有限群元素模逆、模运算等子问题。

本文着重阐述算法的寻找及优化的原理，并在OpenSSL程序包的基础上改写随机数的生成实现，取得了较高效率。

1 RSA算法原理

假设明文分组为M，密文分组为C。RSA算法主要参数如下：

p，q为2个素数(保密)；令$n=pq$(公开，计算得到)；由欧拉定理及p，q是素数，可得$\phi(n)=(p-1)(q-1)$；选取e，使得$\gcd(\phi(n),e)=1$，$1<e<\phi(n)$(公开，选择的)；由$de\equiv 1 \bmod \phi(n)$，计算d(保密)；这样，(e, n)作为公钥，(d, n)作为私钥。

加密算法：$C=M^e \bmod n$

解密算法：$C^d \bmod n=(M^e \bmod n)^d \bmod n=M^{cd} \bmod n=M$

RSA算法首先要将信息数字化，并取长度小于$\log_2 n$位数字作为输入信息块，是属于分组密码。

从上面形式化来看，密钥的产生很简单，但实际上要追求高效率地实现它还是相当有难度的。

2 大素数的生成

密钥对的生成过程大部分时间都花费在两个大素数的寻找上面，因此有必要优化大素数生成算法以提高密钥对的生成速度。目前素数寻找主要有两个思路：一是在寻找过程中确定性地测试所得到的数是素数，这种方法在计算上几乎是不可行的；二是随机生成一个数，然后用概率素性测试算法对得到的数进行测试，普遍使用的是这种方法。

2.1 素数的分布

小于N的素数的个数近似地等于$N/\log N$[3]。如果x是随机选取的小于N的一个整数，则x是素数的概率约为$(N/\log N)/N$，即是$1/\log N$。如果x是一个n位(二进制位，以下同)的整数，那么$\log x=n\times\log 2=0.69n$。所以，$n$位以内的整数素数的比率大概为$1/0.69n$。表1给出了素数的分布。

表1 素数分布

	256位	512位	1 024位	2 048位
n位整数内平均每多少个整数含有一个素数	176	355	710	1 420

2.2 加速素数寻找的预筛选算法

常用的概率素性测试算法有Fermat，Solovay-Strassen和Rabin-Miller[4]，本文中用T来表示。则一个简单的素数寻找算法可以如下设计：

算法1

(1)随机产生一个n位奇数q；

(2)测试q，如果$T(q)$ = false，转向步骤(1)；

(3)输出q并结束。

由表1可知，要产生一个512位的素数平均要对178个奇数(355/2 = 178，因为不用测试偶数)调用素性测试函数，要产生一个1 024位的素数则要710/2 = 355次。

对上述算法一个改进就是引入预筛选算法，q在进行T

基金项目：广东省和广州市科技攻关基金资助项目(2006B15401002，2006Z3-D0151)

作者简介：姚国祥(1959－)，男，教授，主研方向：计算机网络，信息安全，大型数据库系统；林良超，硕士研究生

收稿日期：2007-05-15 **E-mail：**langchue@yahoo.com.cn

测试前得先通过预筛选算法，只有那些通过筛选的 q 才有资格进入 T 测试。筛选算法可由试除法改进得来，主要是第一次取随机，以后递增 2 的方法。试除法主要思想就是构造一个小素数表，用表中元素对待测数 q 筛选。

算法 2

(1)令 p_i 为第 i 小的奇素数($p_1=3, p_2=5, p_3=7, \cdots$)。

(2)令 $S(k)$ 是 p_i 的集合，即 $S(k)=\{p_i \mid p_i \le k, k \in N\}$，其中，$k$ 是正整数。

(3)对给定的整数 q，依次用 $S(k)$ 中的元素去除 q。

(4)如果 q 不能被 $S(k)$ 中任一元素整除，则 q 通过筛选；否则，q 不能通过筛选。

试除法对于阻碍那些奇合数进入T测试是很有效的，对于 512 位整数经过筛选后进入T测试比率如表 2 所示[5]。

表 2　512 位整数经过筛选后进入 T 测试比率

	$S(29)$	$S(256)$	$S(512)$	$S(2560)$	$S(5120)$
比率/(%)	30.9	20.0	17.8	14.03	13.1

结合表 1 来看，未引入筛选算法时每找到一个 512 位素数平均需要进行 178 素性测试，如果经过 S(29)筛选，则只需测试 178 * 30.9% = 55 次；如果用 S(512)筛选则测试次数更加少，只需 178 * 17.8% = 32 次。这缩减了调用素数测试程序的次数，因此也提高了产生素数的效率。

下面细化筛选算法的具体实现及进一步讨论一些技巧性的优化。令 $q^{(i)}$ 是算法 1 中第 i 次产生的随机数，这样的随机序列记为 $q^{(0)}, q^{(1)}, \cdots, q^{(i)}, \cdots$，在计算机中每次产生伪随机数都是耗费较多计算时间的，因此，可只在第一次产生随机的奇数 $q^{(0)}$，对于 $i \ge 1$，可由 $q^{(i)}=q^{(i-1)}+2$ 计算得来。

另一方面，在算法 2 的步骤(3)中，对 $\forall p_j \in S(k)$ 都需要判断 $q \bmod p_j$ 是否为 0，每次都需要取模运算。事实上，按照 $q^{(i)}=q^{(i-1)}+2$ 取 q 序列，在这种取法下算法 2 步骤(3)的取模运算还可以优化。

令 $w_j^i = q^{(i)} \bmod p_j$，则 $w_j^{i+1} = q^{i+1} \bmod p_j = (q^{(i)}+2) \bmod p_j$ $= (w_j^{(i)}+2) \bmod p_j$。

引入数组 $[w_1^{(i)}, w_2^{(i)}, \cdots, w_j^{(i)}, \cdots]$，则下一次运算可由上一次迭代而来。

算法 3

(1)for each $j \in \{r \mid p_r \in S(k), r \in \mathbb{N}\}$，$\mathbb{N}$是自然数集，计算 $w_j^{i+1} = w_j^i + 2$。

(2)如果 $w_j^{i+1} \ge p_j$，则 $w_j^{i+1} = w_j^{i+1} - p_j$。

因此，算法 2 的步骤(4)可以这样来实现：对于 $q^{(i)}$ 只需扫描一遍数组 $[w_1^{(i)}, w_2^{(i)}, \cdots, w_j^{(i)}, \cdots]$，若数组中没有元素为 0，则 $q^{(i)}$ 通过筛选；只要有一个元素为 0，$q^{(i)}$ 都不能通过筛选。最终的算法如下：

算法 4

(1) 随机选择 n 位奇整数 q，令 $i=0$，$[w_1, w_2, \cdots] = [q \bmod p_1, q \bmod p_2, \cdots]$

(2)for each $j \in \{r \mid p_r \in S(k), r \in \mathbb{N}\}$，如果存在 j 使得 $w_j = 0$，则转向步骤(4)。

(3)如果 $T(q)$ = True，输出 q 并且结束；否则，转向步骤(4)。

(4)for each $j \in \{r \mid p_r \in S(k), r \in \mathbb{N}\}$，$w_j = w_j + 2$，如果 $w_j \ge p_j$，则 $w_j = w_j - p_j$。

(5)$q = q + 2$，$i = i + 1$，转向步骤(2)。

2.3　T 函数算法的确定

下面讨论 T 函数(素性测试函数)的选取，本文使用 Rabin-Miller 概率测试法，这也是美国国家标准和技术研究所(NIST)在数字签名标准(DSS)建议中推荐使用的算法。

T(n)算法描述：设已知 $n-1=2^k q$，其中，$k > 0$，q 为奇数。

算法 5 (素性测试)

(1)在{2，…，n-1}中随机产生一个整数 a，计算 $m = a^q \bmod n$。

(2)if $m = 1$，then 返回“通过测试”。

(3)for $j = 0$ to $k-1$ do

　　$m = m^2 \bmod n$

　　if　$m = n$-1，then 返回“通过测试”。

(4)返回“非素数”。

注：通过测试并不能说该数就是素数，还应该增加一个参数 t，表示算法 5 执行的次数，一般取为 10，则 n 是素数的可信度达到 0.999 999。

3　公钥和私钥的生成

找到了大素数后，下一步是寻找整数e，并对e进行模逆运算得到d，这样密钥对(e, n) (d, n)就产生了。常用而高效求解模逆的方法就是扩展欧几里德算法[6]。

算法 6(公钥和私钥生成算法)

(1)计算 $m = (p-1)(q-1)$。

(2)随机选取 $1<e<m$。

(3) $[A_1, A_2, A_3] \leftarrow [1, 0, m]$ ；$[B_1, B_2, B_3] \leftarrow [0, 1, e]$

(4)if $B_3 = 0$，then转向步骤(2)；

(5)if $B_3 = 1$，then

　　if　$B_2 \ge 0$, then 返回$d = B_2$

　　else 返回$d = B_2 + m$

(6) $Q = floor(A_3 / B_3)$。

(7) $[T_1, T_2, T_3] \leftarrow [A_1 - QB_1, A_2 - QB_2, A_3 - QB_3]$。

(8) $[A_1, A_2, A_3] \leftarrow [B_1, B_2, B_3]$。

(9) $[B_1, B_2, B_3] \leftarrow [T_1, T_2, T_3]$，转向步骤(4)。

以上算法是在找到了两个大素数之后紧接着执行的，步骤(4)利用扩展欧拉算法“副产品”检测e和m是否互素，当B_3等于 0 时，e和m有非 1 的公因子，转向步骤(2)重新选择e；当B_3等于 1 时，B_2的值是e模m的逆，从而进一步求得私钥d；当B_3不等于 0 或者 1，执行步骤(6)~步骤(9)，再转向步骤(4)进行判断。

4　总结

本文详细地研究了 RSA 密钥高效生成过程算法，大素数的生成、素性测试，及模逆运算的扩展欧几里德法，其中以大素数的寻找作为算法优化的重点。

RSA 算法在公钥密码体制中占有十分重要的地位，其数学原理非常简单，但在系统实现却比较复杂，通过研究密钥对具体产生过程对掌握整套算法的实现起到了关键的作用。

参考文献

1 Diffie W. The First Ten Years of Public-Key Cryptography[J]. Proceedings of the IEEE, 1988, 76(5).

2 田莹等. RSA 算法的一种快速软件实现[J]. 计算机应用与软件, 2004, 21(3).

(下转第 152 页)