A RAPID GENERATION SCHEME OF LARGE PRIMES

LINA ZHANG
Department of Computing Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, Shaanxi, China

ABSTRACT

RSA is by far the most widely adopted standard in public key cryptography algorithm. Its security depends on the Integer Factorization Problem, that it is very easy to calculate the product of two large prime numbers, but the decomposition the product and get the prime factors are very difficult. Therefore, the generations of large primes are important research field. This paper presented a rapid generation scheme of it, and the techniques for the related software implementation were presented. To speed up the modular multiplication and squaring, Montgomery's algorithms were used with sliding window method. Three pretreatments were also described in details. In view of prime generation of RSA, a series of design methods for software implementation was proposed and give the optimization programs.

Keywords: *Prime Generation, Rabin Miller Test, RSA*

1. INTRODUCTION

The public key cryptography algorithm (also named the asymmetrical crypto-algorithm) uses a pair of keys. The key uses for encryption called public key, which could be gotten by anyone is different from the key for decryption. The decryption key is as the private key, which needed to be kept secret and could not be calculated through the public key (at least in a long time of the reasonable hypothesis). Since 1976, the famous scholars Diffie and Hellman invented public key cryptography; the related technology has gained considerable development and widespread application [1-4]. This idea made the researchers seek for various mathematical difficult problems to construct one kind of function to be called the trapdoor one way function [5]. The trapdoor one way function is the foundation of the public-key cryptography algorithm; the problem to constitute trap-door one-way function is the cornerstone of the security. Currently, there are three types of mathematical problems known can be used to construct the public key cryptography. One is the Integer Factorization Problem (IFP). The second is the Discrete Logarithm Problem in finite fields (DLP), and the other is the Elliptic Curve Discrete Logarithm Problem (ECDLP).

RSA [6] was proposed by Rivest, Shamir and Adleman in 1977. It was the first successful public-key crypto system in theory and enable to both encrypting and signing. RSA's security depends on the IFP problems that it is very easy to calculate the product of two large primes, but the decomposition the product and get the prime factors are very difficult. In order to obtain the public key and private key in RSA, it must first construct two large primes. The security of RSA is closely related to the use of it. Therefore, the study of the rapid generation of large primes is the basis of RSA and of great practical significance.

2. BRIEF INTRODUCTION TO RSA

The RSA algorithm is described as follows:
(1)Select two large random primes: p and q.
(2)Calculate $n = p*q,\ \ \varphi(n) = (p-1)*(q-1)$.
(3)Select a number $1 < e < \varphi(n)$, such that $GCD(e, \varphi(n)) = 1$.
(4) Calculate d, such that $d*e \equiv 1 \bmod \varphi(n)$.

This system can be abbreviated as RSA(n,e) ; where <n,e> are public keys, and could be open for anyone want to use, <n,d> are private keys, which are needed to be kept secret.

When encrypt let the message M, $0 \le M < n$ computed the cipher $C \equiv M^e \bmod n$.

When decrypt, let the cipher C, $0 \le C < n$ Computed $M' \equiv C^d \bmod n$.

By above algorithm, it must first generate two large primes before construct the system parameters such as the public key, the private key and the modulus n. The time of generating primes almost took up most of the time that the key generation has had. In addition, the length of the primes determined the length of the modulus n, which determined the security for the system.

3. ELEMENTARY THEORY

3.1 The Basic Processes of Primes Generation

The general flow of generating prime is as shown in Fig. 1. At first, it needed to generate a random number in a given length, set its top and bottom numbers both to “1”. The first “1” was used to ensure that the length of the random number was as desired, the second “1” could ensure that the random number is odd.

Followed by a method of pre-screening, it controlled the selecting steps of the initial random number, and then put the testing random numbers into the testing until you found a random number so far. According to prime distribution theorem, it could always find a prime within a certain range so long as has assigned the random number seed.

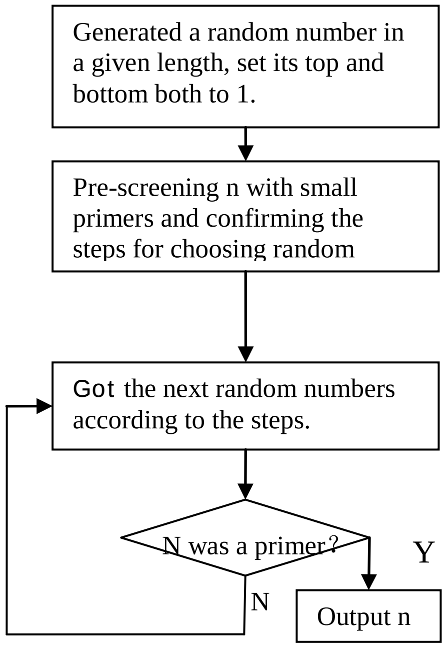
Generated a random number in a given length, set its top and bottom both to 1.
Pre-screening n with small primers and confirming the steps for choosing random
Got the next random numbers according to the steps.
N was a primer?
Y
N
Output n

Figure 1.The Basic Processes Of Generating Primes.

Figure 1 shows that the most critical step was to determine whether the given number n was a prime. The methods for primality testing could be divided into two board categories: deterministic algorithms and probabilistic algorithms. The number generated by the latter are only pseudo-primes, despite the possibility to composite was unlikely but still exists. Their advantages were that there was no regularity, and the speed is relatively fast.

The deterministic primality testing algorithms include original trial division, the Lucas theorem-based approach, AKS, etc. [7]. AKS was to be released in 2002 by three Indian scientists Agrawal, Kayal and Saxena, which were so far the first strict primality proved algorithm in polynomial time. The probabilistic algorithm includes Farmat test, Solovay-Strassen Test, Lehmann Test, Miller Rabin Test and elliptic curve primality testing, etc. [8]. The combination of pre-screening method with Miller Rabin is a more effective means in practical application. This paper’s research and software implementation is also based on it.

3.2 Primes Distribution Theorem

Theorem 1(Primes Distribution Theorem): let $\pi(x)$ the number of primes that not larger than x. $\lim_{x\to\infty}\frac{\pi(x)}{x/\ln x}=1$. When $x\to\infty$, $\pi(x)\approx x/\ln x$.

The Primes Distribution Theorem was discovered by Gauss. It gave the approximate distribution of primes. Since then there had been many mathematicians have made a better estimate [9]. This paper does not make too many descriptions about it, and only estimated that how many steps to find an assigned length’s prime with a given initial value through the Distribution Theorem. The number of the binary numbers of length m is 2^m-2^{m-1} . The number of primes is $\frac{2^m}{\ln 2^m}-\frac{2^{m-1}}{\ln 2^{m-1}}\approx\frac{2^{m-1}}{m\ln 2}$. The distribution density of the primes is about $\frac{2^m}{m\ln 2}/(2^m-2^{m-1})=\frac{1}{m\ln 2}$. Therefore, the prime’s length is 256 bits in the RSA-512. The density of the primes is $256\ln 2\approx 177$, and there is a prime in 177 numbers. If only considering odd numbers, there is a prime in 89 numbers. Corresponding to RSA-1024, there is a prime in 178 numbers. Thus, for any given initial random number, a large prime which’s asset value following it could be found through the effective steps.

3.3 Rabin Miller Test

Rabin Miller test provided an efficient probability method for the detection of a given number n was a prime. For an odd number n, it could be written in the form of $n=2^t s+1$, where s was also an odd number. Let $2\le a\le n-2$. If $a^s\equiv 1$ or $a^{2^j s}\equiv -1(\bmod n)$. Where $0\le j\le r-1$, then n got through the testing. The advantage of this detection method was very fast. The complexity was $(1+\mathrm{O}(1)\log n)$. The drawback was a composite number adopted for a selected “a” of a probability of 1/4, and it belonged to the probabilistic algorithm. If n had passed the testing for N times, the probability of an odd number is smaller than $1/4^N$.

The pseudo-code description of the algorithm is as Figure 2.

```
Find t, q, st. $n = 2^t q + 1$
for i=1 to t
Select random a, st $2 \le a \le n-2$
$x = a^q \bmod n$
if $(x \ne 1)$
i=1
  while $(x \ne n-1)$ do
  if $(i = t)$ return n is a composite number
   $x = x^2 \bmod n$
  if $(x = 1)$ return n is a composite number
    i++
  endwhile
endif
endfor
Return n is a pseudo-prime.
```

Figure 2. The Pseudo-Code Description Of Rabin Miller Algorithm.

4. THE ALGORITHM OF PRIMES GENERATION

4.1 Simple Pretreatment

In this design, it used a combination of getting the system current time and DES encryption to generate enhancement random numbers. Part of its pseudo-code is as Figure 3.

Pretreatment 2: Set 1 to the first and last positions of the generated random number n, part of its pseudo-code is as Figure 4.

Pretreatment 3: For the random number "a", it would be the numeral arbitrary in the Rabin Miller test. In the actual selection, we could control the length of "a" to speed up the operation of Modular exponentiation. Part of its pseudo-code is as Figure 5.

4.2 Pre-screening Primes

Primality testing is generally more time-consuming. The random number sequence needed to be pre-filtered first. It could filter out most of the composite number in the sequences. It usually uses the trial division to filter. That is, with some small primes $S(k)$. $S(k) = \{p_1, p_2, ..., p_n, p_n < k\}$. Such as 3, 5,7,11, etc. It needed to test if the testing number n is divisible by these primes. If it is, then n is a composite number, and need to label it that could not to test it in the primality testing. The idea of pre-screening method is from the famous Eratosthenes sieve method. This algorithm is very simple, and invented by Eratosthenes, who was the Greek astronomer, mathematician and geographer in the third century BC. The algorithm is based on the following Theorem 2.

```
void RandomBigNumber (BigNumber
*bn,int e)
{
   time(&now);
   srand( (unsigned)time( NULL ) );
   for(i=0;i<e;i++)
     buf[i] = (BYTE) rand();
   for(i=0;i<e;i=i+2)
   Random_des_crypt(bn, buf, buf );
}
```

Figure 3. The Pseudo-Code Description Of Pretreatment 1.

```
void PRIME_GENinit(BigNumber * bn, int n)
 {
    RandomBigNumber(bn,n/WordLEN);
    bn ->Data[0]= bn ->Data[0]| 0x00000001;
    bn ->Data[n/WordLEN-1]=
 bn ->Data[n/WordLEN-1]| 0x80000000;
 }
```

Figure 4. The Pseudo-Code Description Of Pretreatment 2.

```
Random bn (&a,n/WordLEN-5);
```

Figure 5. The Pseudo-Code Description Of Pretreatment 3.

Theorem 2: Let n be a positive integer. If for all the prime number $p \le \sqrt{n}$, there is $p \perp n$, and then n must be a primer.

In this paper, the random numbers need to be primality testing would be pretreatment by the bit array algorithm. The composite number that did not comply with the requirements could be pre-marked. Therefore the steps of the primality testing could be accurately determined. The specific algorithm could be found in [10]. As belonging to the software implementation in the PC, it basically does not have the problem of resource constraint. So if it needed to generate a large prime, it could appropriately expand the small prime numbers selected, to get a larger step during the pre-screening. The number of primes we pre-selected is about 1000 in the bit array algorithm. It would successfully find a 512-bit prime number in almost one round. As shown in Table 1, there are certain statistical regularities between the number of small primes, which used as divisors and the composite numbers could be screened out.

Table 1: The Percentage Of 512-Bit Random Numbers That Was Not Be Divisible By Small Primes.

The count of random numbers	Percentage
29	30.9%
256	20.0%
512	17.8%
2560	14.3%
5120	13.1%

We could know from Table 1, the more small primes, the slower the rear retention ratio to the number reduced. Therefore the appropriate small prime number was chosen by the time that the 512-bit's large number modular a small number on personal computers.

4.3 The Design of Primality Discriminant

Seen from 3.3, the main operator of pre-screening is large numbers modular small primes. The computing in Rabin Miller test included modular exponentiation and modulus square. The key operator was modular multiplication.

4.3.1 Modular

In order to quick complete the modular operator with small primes, the size of them could be controlled in 32 bits in programming and stored with the long-type data. To order to borrow in the computing process, the dividend could be divided into two parts to be calculated in the design. Part of its pseudo-code is as Figure 6.

4.3.2 Modular multiplication

Modular multiplication is one of the basic arithmetic in public key cryptographic operations. Intuitively, for $\forall x, y \in F_p$, the calculation process of $xy \bmod p$ could be seen as the calculation of r which satisfy $xy = q \times p + r \quad r \in F_p$, Where $q = \left\lfloor \frac{xy}{p} \right\rfloor$ is the quotient that xy divided by q, r is the remainder. Proceeding from this, the modular multiplication algorithm could generally be divided into three types [11-13].

```
void BigNumber_ModPTo(BigNumber * bn,
BYTE w, int e, BYTE *Result)
{
   Word _result;
   int i;
   DWord tmp;
     tmp =((DWord) bn ->Data[wordlen-
1])<<WordLEN | bn ->Data[wordlen-2];
     _result =(Word) tmp % w;
      for(i=wordlen-3; i>=0; i--)
     {
        tmp =((DWord)_result)<<WordLEN |
bn ->Data[i];
        _result = (Word)tmp % w;
     }
          *Result = (Word)_result;
}
```

Figure 6. The Pseudo-Code Description Of Modular Small Primes.

The first algorithm is estimated quotient to calculate the remainder. The specific process is first calculated xy , calculated the estimated quotient $\hat{q} \approx q$, and then computed $r = xy - \hat{q}p$. Finally, after several subtractions, r would approximately reduce to $[0, p-1]$. The classic algorithm and Barrett algorithms are typical representatives of this algorithm.

The second algorithm is interleaved modular multiplication. The process that calculated xy was intertwined with the reduction. Every step of the intermediate and final results was reduced to $[0, p-1]$.

The third algorithm is Montgomery modular multiplication, which was proposed by Montgomery in 1985. This algorithm could be seen as an interleaved modular multiplication that from low to high (LSB First), and made the appropriate conversion of input parameters x and y, so that more suitable for software and circuit implementation.

Through programming we knew the classic algorithm was a more efficient algorithm for an arbitrary length of the x, y of the modular multiplication. When it needed to calculate $xy \bmod p$ for only one time and the module p is less than 768 bits, the Barrett algorithm would be faster. Montgomery's algorithm would be a better choice if the modular was long, and it needed to do a continuous modular multiplication, for example, modular exponentiation.

In this paper, the design would choose the Montgomery's algorithm to implement. The sliding window method would also be used in modular exponentiation. The zero sequence would as long as possible by efficient encoding, thereby reducing the calculation of modular multiplication and to improve the efficiency of the whole operation.

5. CONCLUSION

This paper discusses the processes and critical steps to be quickly generate large primes in RSA. The main content includes the basic flow, pretreatment, modular and modular multiplication, etc. The design and related software implementation are completed. The specific details of the pseudo-code are proposed. The speed tests of the primes' generation are completed, and the test programs are written in vc6.0.

REFERENCES:

[1] W. Diffie, M.E, "Hellman, New directions in cryptography", *IEEE Transactions on Information Theory*, Vol. 22, No. 6, 1976, pp. 644-654.

[2] R.L. Rivest, A. Shamir, L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120-126.

[3] H.S. Rhee, J.H. Park, D.H. Lee, "Generic construction of designated tester public-key encryption with keyword search", *Information Sciences*, Vol. 205, 2012, pp. 93-109.

[4] S.H Su, S.W Lü, "A public key cryptosystem based on three new provable problems", *Theoretical Computer Science*, Vol. 426-427, 6 April 2012, pp. 91-117.

[5] A.C. Yao, "Theory and applications of trapdoor functions", Proceedings of the 23rd Annual Symposium on the Foundations of Computer Science, IEEE, 1982, pp. 80-91.

[6] Ç. K Koç , "High-Speed RSA Implementation", Technical Report TR-201, version 2.0, RSA Laboratories, November 1994.

[7] M. Agrawal, N. Kayal, N. Saxena, "Prime is in P", *Annals of Mathematics*, 2004, pp. 781-793.

[8] M.O. Rabin, "Probabilistic algorithm for testing primalit", *Journal of Number Theory*, Vol. 12, 1980, pp. 128-138.

[9] A. Granville, G. Martin, "Prime Number Races", *American Mathematical Monthly*, Vol. 113, No. 1, 2006, pp. 1-33.

[10] L.N. ZHANG, J.H. CHEN, J.H. ZHANG, "Compare and study of RSA Key Generation on Smart Cards", *Journal of Computer Applications*, Vol. 26, 2006, pp. 149-150.

[11] P. BARRETT, "Implementing the Rivest, Shamir and Adleman public-key encryption algorithm on a standard digital signal processor", Proceedings of Advances in Cryptology: CRYPT0 '86, Berlin, Germany 1987.

[12] J. Chao, N. Matsuda and S. Tsujii, "Efficient construction of secure hyperelliptic discrete logarithm problems", Proceedings of ICICS'97 INCS 1334, Springer-Verlag 1997, pp. 292-301

[13] P. Montgomery, "Modular Multiplication Without Trial Division", *Mathematics of Computation*, vol. 44, 1985, pp. 519-521.