File encryption and decryption system based on RSA algorithm

Suli Wang

School of Information Engineering
Jingdezhen Ceramic Institute
Jingdezhen, Jiangxi Province, China
sallysur@163.com

Ganlai Liu

Support Center
Jingdezhen Telecom
Jingdezhen, Jiangxi Province, China
liugl@189.cn

Abstract-***This paper describes a complete set of practical solution to file encryption based on RSA algorithm. With analysis of the present situation of the application of RSA algorithm, we find the feasibility of using it for file encryption. On basis of the conventional RSA algorithm, we use C + + Class Library to develop RSA encryption algorithm Class Library, and realize Groupware encapsulation with 32-bit windows platform. With reference of this Groupware on Net platform, you can realize the window application of encryption operation on any files with RSA algorithm.***

Keywords- *RSA algorithm; file encryption and decryption;Portable Components*

I. INTRODUCTION

RSA public key encryption algorithm is developed by Ron rivest, Adishamir and Leonard adleman in 1977, is the most influential public-key encryption algorithm, and has been recommended for ISO public key data encryption standard. RSA algorithm is an asymmetric cryptographic algorithm, the asymmetric, meaning that the algorithm requires a key pair, use one of the encryption, you need to be decrypted with another.

RSA algorithm can be simply described as follows：

- <Generate keys >
- Take the prime number p,q, make n=p×q
- Taken an integers e which coprime with (p-1) × (q-1)
- Solution of the variant d by the equation d×e=1 (mod (p-1)×(q-1))
- Tuple (e, n) as a public key
- Tuple (d, n) as the private key
- < Encryption and decryption >
- $b=a^e \bmod n$， $a=b^d \bmod n$.（A is expressly, b is cipher text）

RSA technology has formed a relatively complete international norm in all aspects of electronic security field. On the hardware side, it is used in a variety of sophisticated consumer electronics products with the mature IC technology. In terms of software applications, mainly in the Internet, RSA is widely used in encrypted connection, digital signatures and digital certificates core algorithms.

II. PROJECT DESIGN

After detailed needs analysis, software requirements can be summarized as follows:

- Can generate the asymmetric keys with required digits
- You can save and load the key, the key is saved as plain text
- Can use specified key to encrypt any file with RSA algorithm, encrypted data generated as plain text
- Encrypted files can be loaded and decrypted with the specified key to restore the original file

978-0-7695-4501-1/11 $26.00 © 2011 IEEE
DOI 10.1109/ICCIS.2011.150

- Message integrity, ease of operation, the graphical interface decency

III. THE SPECIFIC DESIGN

As the design process involves many classes and programs, so the following list only a few key technical designs, the specific encryption and decryption processes are all standard RSA algorithm, so it is skipped to mention.

A. Storage of large number and arithmetic operation

Large number storage is provided by the Flex_unit Class. Large numbers are stored in a linear group with unit named as Unsigned. In the method Void Reserve (unsigned x) by the C++ to give a new open space for pointer variable a, When Flex_unit instance is stored in a larger number than the current stored one, they will call the reserve to increase storage space, but when Flex_unit instance is stored in a smaller number than the current stored one, storage does not automatically tighten to improve the efficiency of computing. With pointer variable a, there are two important unsigned integer to control the storage, which is unsigned variable z and unsigned variable n. Variable z is the number of units assigned space, with the increasing number of larger, not their crunch, and variable n is the share of large numbers currently stored in the number of units, each composed of a large numbers of Unsigned and read into the unit from the Set method, Get method to complete, the variable n is read-only.

Based on the Flex_unit Class storage capabilities, we may have new class of Vlong_value, implement arithmetic functions, and to achieve the cast operator unsigned, to facilitate the large number of each type and ordinary integer assignment. When large number is cast to Unsigned, it will be the lowest four-byte value.

B. Montgomery modulus algorithm

Exponentiation modulus operation is the largest proportion of RSA algorithm; it most directly determines the RSA arithmetic performance. In the code of this software, we directly scan vlong binary By-bits canning.

To improve the speed of exponentiation modulus operation, the key is to improve the speed of modular multiplication. This software is the application of Montgomery algorithm.

Select a cardinal number $R=2^k$ which coprime with modulus n, variable n satisfy $2^{k-1}\le n<2^k$, variable n should be odd number.Select R^{-1} and n', Satisfy 0<R-1 <n, 0 <n '<n, making the RR^{-1}-nn' = 1. For 0 ≤ m <Rn for any integer, Montgomery multiplication is given modulo mR-1 mod n Fast Algorithm M (m):

M (m)

{

$$\lambda = (m \bmod R)n' \bmod R; 0 \le \lambda \le R$$

$$t = (m + \lambda n) / R$$

if (t≥n) return (t-n);

else return t;

}

Because $\lambda n \equiv mnn' \equiv -m \bmod R$, so variable t is integer, At the same time $tR \equiv m \bmod n$,so $t \equiv mR^{-1} \bmod n$ 。 And because $0 \le m + \lambda n \le Rn + Rn$, variable t results in the range 0 ≤ t <2n, return if variable t is not less than variable n, should return t-n。

In this program, RSA's core operations by using the modulus multiplication algorithm is M (A * B). Although M (A * B) is not really needed by the results of modulus multiplication, but as long as the power mode algorithm to be modified accordingly, you can call this modulus multiplication algorithm to calculate.

Use above modulus multiplication algorithm by combining the above described power modulus algorithm, it can constitute a standard power module Montgomery algorithm, which is the software used by the process described below:

```
M(m)
{
k = ( m * n' ) mod R;
x = (m + k*n ) / R;
      if (x>=n) x -= n;
return x;
}
Exp(C,E,n)
{
      D=R-n;
      P=C*R mod n;
      i=0;
      while(true)
            {
            if(Ei==1)D=M(D*P);// Ei is the current binary bit of E
            i+=1;
            If (i== binary digits of E)
            break;
            P=M(P*P);
           }
      return D*R^-1 (mod n);
}
```

We can use the Mul method and Exp method of class of Monty in the practical realization. The global function Modexp can initialize the object of Monty and call it's exp method, we can directly call Modexp when operation.

C. Eratosthenes to find primes and Fermat primes screening test

Screening integers in the range of integer filter to find prime numbers, with all known to be a composite number of integer excluded. Program constructs an array b [ ], size is a prime number search range, the size of search range is minded variable SS. Array b [0] to b [SS] correspond to the large number Start to Start + SS. First of all elements of b[ ] initialized to 1, corresponding to the large number to determine if a composite number, the corresponding element of b [] is set to 0. Finally, only need to do the exact prime number test for those large number which elements with b [ ] is corresponding to value 1. As long as the number being tested is a prime number reaches a certain threshold, this number is on the sub-prime. This not only ensures the implementation of this program can be completed in a short time, and that makes it possible to obtain relatively high accuracy prime number.

Next, the number of possible prime number (tag array b [] in the value of 1 corresponds to the number of elements) for prime testing.

The software application of Fermat's little theorem directly, take integer variable A which relatively prime with variable p, is for a large prime p should satisfy the A^{P-1}mod p = 1, we put the large prime p into a large integer, the number does meet this relationship may not be a prime number. Then we change variable A, for several tests, if several tests are passed, the probability that this number is prime number is relatively large. By this principle, we write the following test functions of prime numbers.

```
Int is_probable_prime_san ( const vlong &p )
{
   const rep = 4; //Testing times
   const unsigned any[rep] = { 2,3,5,7 };
   for ( unsigned i=0; i<rep; i+=1 )
if(modexp(any[i], p-vlong(1), p ) != vlong(1) )
return 0;
```

```
    return 1;
}
```

If testing passed, the number is a prime number and will pass to previous program to use. And here may be also another problem which can't be neglected, it is to get a composite number which can pass this test. In this case, it is to validate it from mathematical point of view that if RSA encryption can be realized or not. After get a large prime number, that is parameter p and q in the RSA algorithm, we can calculate the key, also the encryption operation.

CONCLUSION

RSA algorithm encryption used in file encryption for small files, any file with asymmetric key encryption into its text can be more convenient to communicate and manage, and it has broad development prospects. The project application was designed to take the efficiency and reusability into account. The whole project opens source code and a variety of development information; it is convenient for the reference and continuous development. Application of this procedure can easily communicate data including arbitrary binary and text files under the environment which demand a high security, such as in public forums.

REFERENCES

[1] Montgomery PL, Modular multiplication without trialdivision[J], Mathematics of Computation, 1985

[2] Oh JH,Moon S J, Modular multiplication method［J］, IEE Proceedings：Computers and Digital Tech-niques, 1998

[3] Shi Xiangdong, Dong Ping, a new core design based on the RSA encryption algorithm, micro-computer information, period 2005 12-3

[4] [AX931] ANSI X9.31-1998 Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), Appendix A, American National Standards Institute, 1998.

[5] [COCK73] Clifford Cocks, A Note on 'Non-Secret Encryption', CESG Research Report，20 November 1973

[6] [KALI93] Burton Kalinski, Some Examples of the PKCS Standards, RSA Laboratories, November 1993