PARALLEL COMPUTATION OF THE GENERATING KEYS FOR RSA CRYPTOSYSTEMS

Chin-Chen Chang †
Email: ccc@cs.ccu.edu.tw

Min-Shiang Hwang ‡
Email: m24@ms.tl.gov.tw

Department of Computer Science and Information Engineering †
National Chung Cheng University
Chiayi, Taiwan 621, R.O.C.

Department of Information Management ‡
Chao-Yang Institute of Technology
P.O. Box 55-67, Taichung, Taiwan 404, R.O.C.

Abstract

This paper presents a parallel implementation for generating RSA keys without the use of Euclidean algorithm. Our method is based on a unique factorization theorem, which combinesd with Derôme's method, and allows generation of RSA keys without the use of Euclidean algorithm.

Indexing terms: Cryptography, Number theory, Algorithms

Introduction: The RSA system [1] is one of the most popular asymmetric cryptosystems and offers high security and easy implementation. In the RSA cryptosystem, the plaintext M is encrypted by the following formula:

$$C = M^e \bmod N, \tag{1}$$

where e denotes a public key (or enciphering key), $N = pq$, p and q are large strong primes. In turn, the ciphertext C can be decrypted by the following formula:

$$M = C^d \bmod N, \tag{2}$$

where d denotes a secret key (or deciphering key). Note that, in order to satisfy Equation 2,

$$ed \bmod s = 1, \tag{3}$$

†

‡

†

‡

where s denotes a Euler totient function of N. We now seek a pair of keys (e, d) that satisfy Equation 3. One of the solutions can be obtained using the extended Euclidean algorithm [2]. The other solution is obtained as follows: choose an e such that $gcd(e, s) = 1$, and then compute the inverse of e by the following equation.

$$\begin{aligned} d &= e^{-1} \bmod s & (4) \\ &= e^{\phi(s)-1} \bmod s. & (5) \end{aligned}$$

Obviously, it is very time consuming to compute the above equation because the value of s is usually very large.

Recently, Derôme proposed an efficient method which does not involve the Euclidean algorithm. In this paper, we extend Derôme's method to implement a parallel computation of the generating keys for an RSA cryptosystem.

Review of Derôme's method: Since our method is inspired by Derôme's method, firstly we briefly review his method. Equation 3 implies that there exists an integer $j > 0$ such that

$$de - js = 1. \tag{6}$$

By Derôme's method, the secret key can be generated without the Euclidean algorithm as follows:

$$\begin{aligned} d &= 1 + j\lfloor \frac{s}{e} \rfloor + [j(s \bmod e)] \bmod (e-1), and & (7) \\ j &= (-s^{\phi(e)-1}) \bmod e. & (8) \end{aligned}$$

It is easy to prove that the secret key d in Equation 7 satisfies Equation 3. Note that whenever the public key e is given, the dual key d can be simply computed by using Equation 7. Thus the Euclidean algorithm can be avoided.

Our method: We present a parallel implementation for generating RSA keys without the Euclidean algorithm in the following steps:

1. Decompose the public key e into $e = e_1^{a_1} e_2^{a_2} \cdots e_n^{a_n}$ by the unique factorization theorem, where $e_1, e_2, \cdots, e_n$ are n primes, and $a_1, a_2, \cdots, a_n$ are n positive integers.

2. Compute $d_1, d_2, \cdots, d_n$ by Derôme's method (shown in Equation 7), where d_i is the dual key of e_i.

3. Compute the secret key d by multiplying all $d_i^{a_i}$, i.e., $d = \prod_{i=1}^{n} d_i^{a_i}$ mod s.

The correctness of our method can be shown as follows. By Derôme's method, we know that

$$\begin{cases} e_1 d_1 & \equiv 1 (\mathrm{mod} s), \\ e_2 d_2 & \equiv 1 (\mathrm{mod} s), \\ & \vdots \\ e_n d_n & \equiv 1 (\mathrm{mod} s). \end{cases} \tag{9}$$

Multiplying all of the equations in Equation 9 together, we have

$$(e_1 e_2 \cdots e_n)(d_1 d_2 \cdots d_n) \equiv 1 (\mathrm{mod} s). \tag{10}$$

$$(11)$$

Thus

$$ed \equiv 1(\bmod s). \tag{12}$$

So Equation 3 holds. Since e has been decomposed to n separate e_i's, Equation 9 can be implemented by parallel processing.

Advantages: There are some important advantages to our method.

1. The computation size is small. Since $e_i d_i \bmod s = 1$, it follows that $d_i e_i - j_i s = 1$. By the Derôme's method, $d_i = 1 + j_i \lfloor \frac{s}{e_i} \rfloor + [j_i(s \bmod e_i)] \bmod (e_i - 1)$, where $j = (-s^{\phi(e_i)-1}) \bmod e_i, i = 1, 2, \cdots, n$. Since

$$\begin{aligned}
\phi(e) &= \phi(e_1)\phi(e_2)\cdots\phi(e_n) &(13)\\
&= (e_1 - 1)(e_2 - 1)\cdots(e_n - 1). &(14)
\end{aligned}$$

   Therefore $\phi(e_i) \ll \phi(e)$. The computation size in our method is thus small.

2. Our method provides an efficient approach to assist end user to choose a public key e. Since $gcd(e_i, s) = 1, i = 1, 2, \cdots, n$, $gcd(e, s) = 1$ is also true, where $e = \prod_{i=1}^{n} e_i$. The user thus can select a set of small primes e_i and then multiply these primes together to produce a larger prime e. And then use our parallel implementation to generate the dual key d.

3. Our method can be implemented by parallel processing. If there are n processors in the system, the secret key d can be generated by one computation of d_i and $(logn)$ computations of $d_i \times d_j$, where $i \neq j$.

References

[1] RIVEST, R. L., SHAMIR, A., and ADLEMAN, L.: 'A method for obtaining digital signatures and public key cryptosystems', *Comm. ACM*, 1978, **21,** pp. 120-126.

[2] KNUTH, D. E.: 'The art of computer programming. Vol. 2. Seminumberical algorithms' (Addison Wesley, 1981), 2nd ed., pp. 317-330.

[3] DERÔME, M. F. A.: 'Generating RSA keys without the Euclid algorithm', *Electron. Lett.,* 1993, **29,**(1), pp. 19-21.