

An Efficient Decryption Method for RSA Cryptosystem

Ren-Junn Hwang, Feng-Fu Su

*Department of Computer Science and Information Engineering,
TamKang University
Tamsui, Taipei, Taiwan 251, R.O.C.
victor@mail.tku.edu.tw*

Yi-Shiung Yeh, Chia-Yao Chen

*Degree Program of Electrical Engineering and Computer Science,
National Chiao Tung University
Hsinchu, Taiwan 300, R. O. C.
ysyeh@csie.nctu.edu.tw*

Abstract

This paper proposes an efficient method to implement RSA decryption algorithm. RSA cryptosystem is the most attractive and popular security technique for many applications, such as electronic commerce and secure internet access. It has to perform modular exponentiation with large exponent and modulus for security consideration. The RSA cryptosystem takes great computational cost. In many RSA applications, user uses a small public key to speed up the encryption operation. However, the decryption operation has to take more computational cost to perform modular exponentiation by this case. This paper proposes an efficient decryption method not only based on Chinese Remainder Theorem (CRT) but also the strong prime of RSA criterion. The proposed decryption method only takes 10% computational costs of the traditional decryption method. It also reduces 66% computational costs than that of decryption methods based on CRT only. In a word, the speed of our proposed method is almost 2.9 times faster than the decryption method based on CRT only. The proposed method enhances the performance of the RSA decryption operation.

1. Introduction

As the rapid progressing of modern information technology, security is an important technique of many applications including virtual private networks,

electronic commerce and secure internet access. RSA algorithm is the most popular and well-defined security primary technique. There are many good security protocols based on RSA [1] cryptosystem. RSA is widely used for digital signature and digital envelope, which provide privacy and authentication. However, the RSA operation has to take great computation cost for security consideration. In order to include RSA cryptosystem efficiently in many protocols, it is desired to devise faster encryption and decryption operations.

In many applications, we usually select a small value such as 3, 17, or 65537 to be the public key to speed up the encryption operation [2]. However, by this way, the corresponding decryption operation costs more computational time because of the larger secret key. We can speed up the decryption operation based on the Chinese Remainder Theorem (CRT) [3, 4] if we get the prime factors of modulus. It is reasonable that somebody who holds the secret key can get the factors of modulus. In addition, Hayashi proposed a new modular exponentiation method [5] to improve the computation time of RSA. In his method, the modular exponentiation with the modulus n transforms to two substitute operations with moduli $n + 1$ and $n + 2$. If moduli $n + 1$ and $n + 2$ can be factoring, user can apply CRT to these modular operations modulo to $n + 1$ and $n + 2$ for each. The final result can be generated by Hayashi's formula. However, n is an odd number, we cannot factor $n + 2$ easily in his method. This method is not practical.

We propose an efficient method to implement RSA decryption operation in this paper. This method is not

only based on CRT but also the strong prime of RSA criterion. The security of RSA is based on the difficulty of factoring problem. So, the prime factors of modulus of RSA algorithm must be strong primes. The large modular exponentiation result can be generated from small exponents and moduli. The proposed method enhances the performance of RSA algorithm.

The rest of this article is organized as follows: we will briefly review RSA algorithm in Section 2. Section 3 introduces our implementation method. Section 4 analyzes the computational complexity. Finally, we make some conclusions in Section 5.

2. RSA algorithm

RSA algorithm is a typical public-key cryptosystem. It is a basic technique of many security protocols. It can be described briefly as follows:

1. Choose two large strong primes, p and q . Let $n = p \cdot q$.
2. Compute Euler value of n : $\Phi(n) = (p - 1)(q - 1)$.
3. Find a random number e satisfying $1 < e < \Phi(n)$ and $\gcd(e, \Phi(n)) = 1$.
4. Compute a number d such that $d = e^{-1} \bmod \Phi(n)$.
5. Encryption: Given a message m satisfying $m < n$, then the cipher text $c = m^e \bmod n$.
6. Decryption: $m = c^d \bmod n$.

The security of RSA is based on the hard of factoring n . To enhance the hardness of factoring n , Ogiwara [6] suggests the following constraints on p and q :

1. Both $(p - 1)$ and $(q - 1)$ should contain a large prime factor such that $r_1|(p - 1)$ and $t_1|(q - 1)$, where r_1 and t_1 are two large primes.
2. Both $(p + 1)$ and $(q + 1)$ should contain a large prime factor such that $s_1|(p + 1)$ and $u_1|(q + 1)$, where s_1 and u_1 are two large primes.

There are many methods have been proposed to generate primes for RSA [6, 7, 8, 9]. These methods first use some primes r_1 and s_1 to find the strong prime p . The bit lengths of r_1 and s_1 are half of the bit length of p . Figure 1 shows the strong prime structure for RSA cryptosystem. Another prime q also can be generated by this method. In addition, the numbers $p - 1$, $p + 1$, $q - 1$ and $q + 1$ can be factored into three factors at least.

3. Our efficient decryption method

By the security consideration, the prime factors p and q of modulus n in RSA cryptosystem must be strong primes. It is reasonable that the secret key

holder knows the prime factors of $p - 1$, $p + 1$, $q - 1$ and $q + 1$. The proposed decryption method is based on the strong primes of RSA criterion and Chinese Remainder Theorem (CRT). Figure 2 shows the diagram of our decryption method. The secret key holder performs the decryption procedure: $c^d \bmod n$ by our method as the following steps.

Step 1 Factor $p - 1$, $p + 1$, $q - 1$ and $q + 1$ to get their prime factors. We assume that moduli $p - 1$, $p + 1$, $q - 1$ and $q + 1$ be expressed as follows.

$$p - 1 = \prod_{i=1}^h r_i^{\alpha_i},$$

$$p + 1 = \prod_{i=1}^j s_i^{\beta_i},$$

$$q - 1 = \prod_{i=1}^k t_i^{\gamma_i},$$

$$q + 1 = \prod_{i=1}^l u_i^{\delta_i}.$$

Step 2 Compute the modular exponentiations with prime factors of $p - 1$ as the modulus. Then, apply the CRT to generate $y_1 = c^d \bmod (p - 1)$ based on these results. By the same way, the secret key holder generates $y_2 = c^d \bmod (p + 1)$, $y_3 = c^d \bmod (q - 1)$ and $y_4 = c^d \bmod (q + 1)$ individually. The detail steps are as follows:

2.1 Compute $c_{(p-1),i} = c \bmod r_i^{\alpha_i}$, $i = 1, \dots, h$.

2.2 Compute $d_{(p-1),i} = d \bmod \Phi(r_i^{\alpha_i})$, $i = 1, \dots, h$.

2.3 Compute the modular exponentiation $m_{(p-1),i} = c_{(p-1),i}^{d_{(p-1),i}} \bmod r_i^{\alpha_i}$, $i = 1, \dots, h$.

2.4 Apply the CRT to generate $y_1 = c^d \bmod (p - 1)$ based on $m_{(p-1),i}$, $i = 1, \dots, h$.

Step 3 Compute $X_1 = 2^{-1}(y_1 + y_2 - z) \bmod p$ where $z = 0$ if $y_1 \geq y_2$; otherwise $z = 1$ and $X_2 = 2^{-1}(y_3 + y_4 - z) \bmod q$ where $z = 0$ if $y_3 \geq y_4$; otherwise $z = 1$.

Step 4 Apply the CRT to generate the plaintext $m = c^d \bmod n$ based on X_1 and X_2 .

The numbers $p - 1$, $p + 1$, $q - 1$ and $q + 1$ can be individually decomposed into three prime factors at least in step 1. The bit lengths of $r_i^{\alpha_i}$ ($i = 1, \dots, h$) are smaller than $p - 1$. The bit lengths of $d_{(p-1),i}$ ($i = 1, \dots, h$) are smaller than d . In general, the complexity of modular exponentiation depends on the bit length of exponent and modulus. The totally computation time of Step 2 to get $y_1 = c^d \bmod (p - 1)$ is smaller than to compute y_1 by $c^d \bmod (p - 1)$ directly. The efficient results are similar to $y_2 = c^d \bmod (p + 1)$, $y_3 = c^d \bmod (q - 1)$ and $y_4 = c^d \bmod (q + 1)$. The proposed decryption method is more efficient than the traditional method. Section 4 will demonstrate these results detail.

In Step 3, we use (y_1, y_2) to compute X_1 , and (y_3, y_4) to compute X_2 . Theorem 1 demonstrates that X_1 and X_2 are generated as Step 3 correctly.

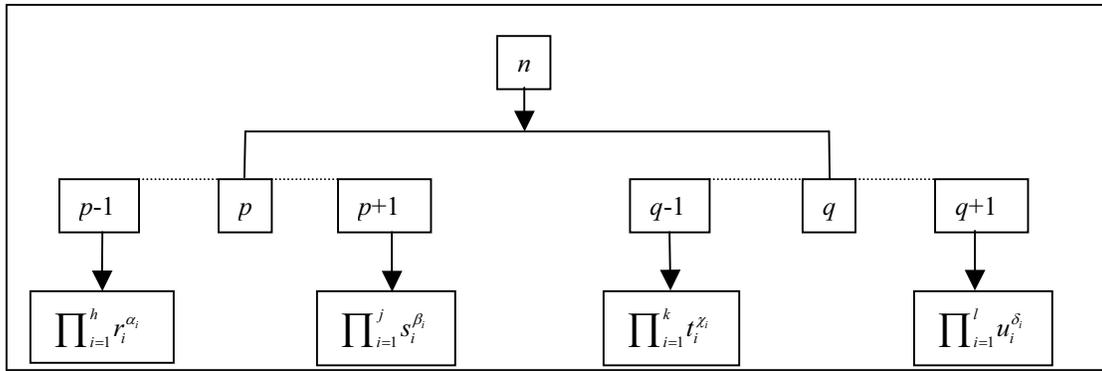


Figure 1 Strong primes for RSA cryptosystem

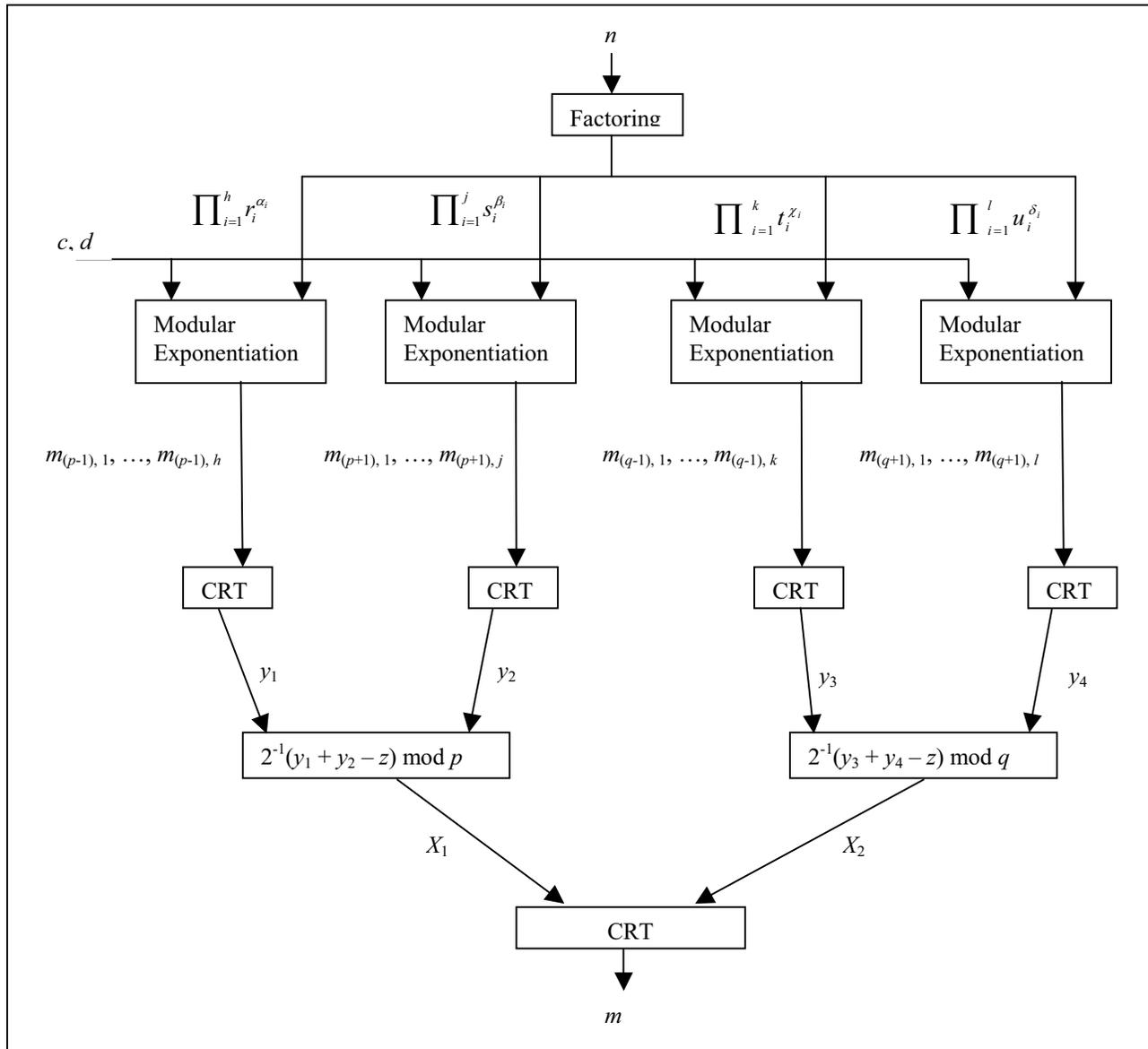


Figure 2 Our proposed decryption method for RSA cryptosystem

[Lemma 1] Given $y_1 = X \bmod (p - 1)$ and $y_2 = X \bmod (p + 1)$ such that $0 \leq X < (p^2 - 1)/2$ and p is a prime, then

$$X = (p + 1)y_1/2 - (p - 1)y_2/2 + (p - 1)(p+1)z/2,$$

where z is either 0 or 1.

Proof:

By $y_1 = X \bmod (p - 1)$ and $y_2 = X \bmod (p + 1)$, we get

$$y_1 + (p - 1)z_1 = X, \quad (1)$$

$$y_2 + (p + 1)z_2 = X, \quad (2)$$

z_1 and z_2 are two positive integers.

Equation (1) \times $(p+1)$:

$$(p + 1)y_1 + (p + 1)(p - 1)z_1 = (p + 1)X. \quad (3)$$

Equation (2) \times $(p-1)$:

$$(p - 1)y_2 + (p + 1)(p - 1)z_2 = (p - 1)X. \quad (4)$$

Equation (3) – Equation (4):

$$2X = (p+1)y_1 - (p-1)y_2 + (p-1)(p+1)(z_1 - z_2). \quad (5)$$

Let $z = z_1 - z_2$, then Equation (5) can be rewriting as

$$X = (p+1)y_1/2 - (p-1)y_2/2 + (p-1)(p+1)z/2. \quad (6)$$

We will proof z is either 0 or 1 in Equation (6) by contradiction as follows:

Case 1:

Assume $z < 0$. Since $y_1 \leq p - 2$ and $y_2 \geq 0$, we get

$$\begin{aligned} X &\leq (p + 1)(p - 2)/2 - (p - 1)(p + 1)/2 \\ &= -(p + 1)/2 < 0, \end{aligned}$$

when $y_1 = p - 2, y_2 = 0$ and $z = -1$.

This result contradicts the given condition $X \geq 0$.

Hence, z is not smaller than 0.

Case 2:

Assume that $z > 1$. Since $y_1 \geq 0$ and $y_2 \leq p$, we get

$$\begin{aligned} X &\geq -(p - 1)p/2 + (p - 1)(p + 1) \\ &= (p - 1)(p + 2)/2 \\ &> (p - 1)(p + 1)/2 \\ &= (p^2 - 1)/2, \end{aligned}$$

when $y_1 = 0, y_2 = p$ and $z = 2$.

This result contradicts the given condition $X < (p^2 - 1)/2$. Hence, z is not larger than 1.

By Cases 1, 2 and z_1, z_2 are integers, z must be either 0 or 1. Q.E.D.

[Theorem 1] Given $y_1 = X \bmod (p - 1), y_2 = X \bmod (p + 1)$, such that $0 \leq X < (p^2 - 1)/2$ and p is a prime, then $X = 2^{-1}(y_1 + y_2 - z) \bmod p$, where $z = 0$ if $y_1 \geq y_2$; otherwise $z = 1$.

Proof:

By Lemma 1,

$$X = (p + 1)y_1/2 - (p - 1)y_2/2 + (p - 1)(p+1)z/2,$$

where z is either 0 or 1.

We get

$$X = 2^{-1}(y_1 + y_2 - z) \bmod p, \quad (7)$$

where z is either 0 or 1.

In addition, we will demonstrate the conditions of $z = 0$ and $z = 1$ in Equation (7).

By Equation (2) – Equation (1):

$$y_2 - y_1 = (p-1)z_1 - (p+1)z_2 = (p-1)(z_1 - z_2) - 2z_2,$$

where $y_1 \geq 0, y_2 \geq 0, p > 0$ and both z_1 and z_2 are two positive integers.

Let $z = z_1 - z_2$, then

$$y_2 - y_1 = (p-1)z - 2z_2. \quad (8)$$

By Equation (2), and $0 \leq X < (p^2 - 1)/2, 0 \leq y_2 \leq p$, we get $0 \leq y_2 + (p + 1)z_2 < (p^2 - 1)/2$.

Hence, $0 \leq z_2 < (p - 1)/2$.

By Equation (8),

$$y_2 - y_1 > (p - 1)z - (p - 1) = (p - 1)(z - 1). \quad (9)$$

Case 1: ($y_1 \geq y_2$)

Since $y_1 \geq y_2$, we get $y_2 - y_1 \leq 0$.

By Equation (9), we get $(p - 1)(z - 1) < 0$.

Since $p - 1 > 0$, we have $z - 1 < 0$.

That is $z < 1$.

By Lemma 1, z must be 0 when $y_1 \geq y_2$.

Case 2: ($y_1 < y_2$)

Since $y_1 < y_2$, we get $y_2 - y_1 > 0$.

By Equation (9), we get $(p - 1)(z - 1) \geq 0$.

We get $z \geq 1$.

By Lemma 1, z must be 1 when $y_1 < y_2$.

Hence, $X = 2^{-1}(y_1 + y_2 - z) \bmod p$, where $z = 0$ if $y_1 \geq y_2$; otherwise $z = 1$. Q.E.D.

Although we need to compute the multiplicative inverse of 2 modular p in Step 3, Theorem 2 provides an efficient method to compute the inverse value.

[Theorem 2]: Let p be an odd prime, the multiplicative inverse of 2 modulo p is $(p + 1)/2$.

Proof:

By the equation:

$$2 \times (p + 1)/2 = p + 1 \equiv 1 \bmod p.$$

We get the multiplicative inverse of 2 modulo p is $(p + 1) / 2$. Q.E.D.

4. Computational complexity

This section regularly demonstrates our proposed decryption method is more efficient than the traditional decryption method and decryption method based on CRT only. We define some notations as follows:

- $MOD_E(y, z)$ denotes an operation of modular exponentiation ($x^y \bmod z$).
- $M(w), A(w)$ and $Mod(w)$ denote operations of multiplication, addition and modulus with the bit length of operand is w .
- $l(w)$ denotes the bit length of w .
- S denotes the shift operator.

By the addition chain method [10] the modulo operation $c^d \bmod n$ can be expressed as:

$$MOD_E(d, n) = 1.5 \times l(d)[M(l(n)) + 2 Mod(l(n)) + 1]. \quad (10)$$

The multiplication and addition operations can be expressed as follows [11]:

$$M(w) = 3M(w/2) + 5A(w) + 2S, \quad (11)$$

$$A(w) = w/32. \quad (12)$$

The modulo operation also can be expressed as the following equations based on the divide and conquer concept [12]:

$$Mod(w) = Mod(w/2) + 4M(w/2) + 1.5A(w) + 3S. \quad (13)$$

Without loss of generality, we assume all of $Mod(32)$, $M(32)$, $A(32)$ and S take one clock cycle. By the equations (11) and (12), we get

$$\begin{aligned} M(1024) &= 3M(512) + 5A(1024) + 2S \\ &= 3M(512) + 162 \\ &= 3[3M(256) + 5A(512) + 2S] + 162 \\ &= 9M(256) + 408 \\ &= 9[3M(128) + 5A(256) + 2S] + 408 \\ &= 27M(128) + 786 \\ &= 27[3M(64) + 5A(128) + 2S] + 786 \\ &= 81M(64) + 1380 \\ &= 81[3M(32) + 5A(64) + 2S] + 1380 \\ &= 243M(32) + 2352 \\ &= 2595. \end{aligned}$$

By the equations (11), (12) and (13), we get

$$\begin{aligned} Mod(1024) &= Mod(512) + 4M(512) + 1.5A(1024) \\ &\quad + 3S \\ &= [Mod(256) + 4M(256) + 1.5A(512) + \\ &\quad 3S] + 3295 \\ &= [Mod(128) + 4M(128) + 1.5A(256) + \\ &\quad 3S] + 4294 \\ &= [Mod(64) + 4M(64) + 1.5A(128) + \\ &\quad 3S] + 4577 \\ &= [Mod(32) + 4M(32) + 1.5A(64) + 3S] \\ &\quad + 4646 \\ &= 4657 \end{aligned}$$

By the consideration of security, the bit length of modulus should be 1024 at least. By Equation (10), the traditional decryption method can be repressed as $MOD_E(d, n) = 1.5 \times 1024[M(1024) + 2 Mod(1024) + 1]$. That is, the traditional decryption method should take 18293760 clock cycles.

If the decryption method based on CRT only and the bit lengths of p and q are equal, the operation of the decryption method can be repressed as $2MOD_E(d/2, n/2) + A(512) + 2M(512) + Mod(512)$. By this case, the decryption method takes 5434296 clock cycles.

In our proposed method, $p - 1$, $p + 1$, $q - 1$ and $q + 1$ can be factored into at least three numbers. Without loss of generality, we assume the bit length of the largest prime factor is about $l(n)/4$ and others are about $l(n)/8$ [6, 7, 8, 9]. The total number of operations of our proposed method is $4MOD_E(d/4, n/4) + 8MOD_E(d/8, n/8) + 4[A(256) + 2M(256) + Mod(256)] + 4[A(128) + 2M(128) + Mod(128)] + 2[A(512) + M(512) + Mod(512)] + A(512) + 2M(512) + Mod(512)$. It takes 1851806 clock cycles.

The traditional decryption method takes 18293760 clock cycles totally. The decryption method based on CRT takes 5434296 clock cycles. However, our proposed method only takes 1851806 clock cycles. In comparing with the traditional decryption method, our proposed method only takes 10% computational cost of the traditional method, but the decryption method based on CRT should take 30%. In other words, the computational cost of our proposed scheme is only 34% of the decryption method based on CRT. The speed of our proposed method is almost 2.9 times faster than the decryption method based on CRT only.

5. Conclusions

In this paper, we propose an efficient method to implement RSA decryption algorithm. This decryption method is not only based on CRT but also the strong prime of RSA criterion. The 1024 bits RSA original decryption method without any tricks must takes 18293760 clock cycles. The decryption method based on CRT takes 5434296 clock cycles. By our proposed decryption method, it only takes 1851806 clock cycles. The complexity of our proposed method is only 10% of the traditional decryption method. Our proposed method reduces approximately 66% computational costs as comparing with the decryption method based on CRT. In a word, the speed of our proposed method is almost 2.9 times faster than the decryption method based on CRT only. Our method can be applied to not only decryption operation but also signing phase of digital signature. This efficient decryption method can enhance the performance of the RSA algorithm.

6. References

- [1] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signature and Public-Key Cryptosystems", *Communication of the ACM*, Vol.21, No.2, 1978, pp. 120-126.
- [2] B. Schneier, *Applied Cryptography: protocols, algorithms, and source code in C*, 2nd ed. John Wiley & Sons, Inc., 1996.
- [3] J.-J. Quisquater, C. Couvreur, "Fast decipherment algorithm for RSA public-key cryptosystem", *Electronics Letters* 18, (21), 1982, pp. 905-907.
- [4] M. Shand, J. Vuillemin, "Fast Implementation of RSA Cryptography", *Proceedings of 11th Symposium on Computer Arithmetic*, 1993.
- [5] A. Hayashi, "A New Fast Modular Multiplication Method and its Application to Modular Exponentiation-based Cryptography", *Electronics and Communications in Japan*, 83 (12), 2000, pp. 88-93.

- [6] M. Ogiwara, "A Method for Generating Cryptographically Strong Primes", *IEICE Trans.*, E73, (6), 1990, pp. 985-994.
- [7] J. Gordon, "Strong RSA Keys", *Electronics Letters*, 20, (12), 1984, pp. 514-516.
- [8] M. J. Ganley, "Note on the generation of P0 for RSA keysets", *Electronics Letters*, 26, (6), 1990, pp. 369.
- [9] C.-S. Lai, W.-C. Yang and C.-H. Chen, "Efficient Method for Generating Strong Primes with Constraint of Bit Length", *Electronics Letters*, 27, (20), 1991, pp. 1807-1808.
- [10] Knuth DE. *The Art of Computer Programming*, Vol. 2 (Semi numerical Algorithms), 2nd ed. Addison-Wesley: Massachusetts, 1980.
- [11] Davida GI, Wells DL, Kam JB., "A Database Encryption System with Subkeys", *ACM Trans. On Database Systems*, 6, 1981, pp. 312-328.
- [12] M.-S. Hwang, "Dynamic Participation in a Secure Conference Scheme for Mobile Communications", *IEEE Trans. On Vehicular Technology*, Vol. 48, No. 5, 1999, pp. 1469-1474.