The Optimal Equations with Chinese Remainder Theorem for RSA's Decryption Process

Kritsanapong Somsuk[1*], Sarutte Atsawaraungsuk[2], Chanwit Suwannapong[3], Suchart Khummanee[4] and Chalida Sanemueang[5]

[1*]Department of Computer and Communication Engineering, Faculty of Technology, Udon Thani Rajabhat University, UDRU, Udon Thani, Thailand. kritsanapong@udru.ac.th, Orcid: https://orcid.org/0000-0002-7264-4628

[2] Department of Computer Education, Udon Thani Rajabhat University, Udon Thani, Thailand. sarutte@udru.ac.th, Orcid: https://orcid.org/0000-0002-6853-6480

[3]Department of Computer Engineering, Faculty of Engineering, Nakhon Phanom University, Thailand. schanwit@npu.ac.th, Orcid: https://orcid.org/0000-0002-7970-3088

[4]Department of Computer Science, Faculty of Informatics, Mahasarakham University, Thailand. suchart.k@msu.ac.th, Orcid: https://orcid.org/0000-0002-6078-1203

[5]Office of Academic Resources and Information Technology, Udon Thani Rajabhat University, UDRU, Udon Thani, Thailand. Chali.sa@udru.ac.th, Orcid: https://orcid.org/0009-0001-2223-6909

Received: April 08, 2023; Accepted: June 02, 2023; Published: June 30, 2023

Abstract

This research was designed to provide an idea for choosing the best two equations that can be used to finish the RSA decryption process. In general, the four strategies suggested to accelerate this procedure are competitors. Chinese Remainder Theorem (CRT) is among four rivals. The remains are improved algorithms that have been adjusted from CRT. In truth, the primary building block of these algorithms is CRT, but the sub exponent of CRT is substituted with the new value. Assuming the modulus is obtained by multiplying two prime numbers, two modular exponentiations must be performed prior to combining the results. Three factors are chosen to determine the optimal equation: modular multiplications, modular squares, and modular inverses. In general, the proposed method is always the winner since the optimal equation is selected from among four methods. The testing findings show that the proposed technique is consistently 10-30% faster than CRT.

Keywords: CRT, RSA, Decryption Process, Exponent.

1 Introduction

In the digital world, a great deal of confidential information is transmitted via the Internet because it is a simple and quick method for exchanging the information. However, the Internet is the unsecured medium via which attackers can quickly access information. Numerous approaches have been developed to preserve the information transmitted over the Internet channel. Cryptography (Guang, G., 1999) is

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 14, number: 2 (June), pp. 109-120. DOI: *10.58346/JOWUA.2023.I2.009*

*Corresponding author: Department of Computer and Communication Engineering, Faculty of Technology, Udon Thani Rajabhat University, UDRU, Udon Thani, Thailand.

one of the algorithms used to protect a secret message by encrypting and decrypting data. In truth, there are two distinct cryptographic techniques. The first method is symmetric key cryptography. The secret key is chosen once and used for both encryption and decryption. When symmetric key cryptography is used for implementation, there are two benefits. The first one is regarding process completion time. In general, this strategy requires less time than the alternative method. The difficulty of intruder attempts is the second advantage. Specifically, AES (Christian, E., 2021) (Issam, H., 2010) and one-time pad (Guanglou, Z., 2015) are examples of powerful algorithms in this group. Moreover, both algorithms are still applicable today. The disadvantage is the difficulty in finding the hidden route for the exchange of the public key between senders and recipients. Asymmetric key cryptography (Diffie, W., 1976) or public key cryptography was introduced to address the problem with symmetric key encryption. In fact, two separate keys are necessary. The public key is the first key. In general, it is shared publicly with the entire group. The second key is the private key, which must be kept in strict confidence. In practice, the secret channel is unnecessary for public key cryptography. However, all algorithms in this group have extremely high computation costs. The utilization of public key cryptography for the protection of secret information is inappropriate. On the other hand, the secret key that will be used to protect the secret information is exchanged using public key cryptography.

RSA (Rivest, R.L., 1978) represents one of the algorithms for public key cryptography. It is recognized as one of the most powerful algorithms in the digital age. RSA is based on the integer factorization (Gupta, S.C., 2021) (Nedal, T., 2018) issue. It will be broken upon the disclosure of all prime factors. Although numerous integer factoring techniques, such as (Somsuk, K., 2018) (Somsuk, K., 2021) (Somsuk, K., 2022) (Somsuk, K., 2020) (Pollard, J.M., 1978) (Omar, K., 2008) (Wu, M.E., 2014), have been presented, there is no effective method to break RSA in polynomial time with at least 1024 bits of modulus. In 1994, P. Shor (Shor, P.W., 1994) introduced the factoring method to break RSA, a novel concept. Shor's algorithm is distinct from other algorithms since it is based on the quantum computer. In addition, he demonstrated that this method on a quantum computer will factor the large modulus in polynomial time. However, quantum computer development is still in progress. In fact, nobody can confirm when the fully quantum computer will be created. Therefore, RSA is still a robust technique that is extensively employed today.

Typically, the private key has a high numerical value. Since the private key is the exponent for computing modular exponentiation, it has an effect on how long it takes to decipher a message. Chinese Remainder Theorem (CRT) (Sung, M.Y., 2003) (Atsushi, M., 2011) is the primary method for accelerating the RSA decryption procedure by dividing the private key into two sub keys. Before combining the results, two modular exponentiations with the new keys are performed. Moreover, modified methods were developed to accelerate this procedure. However, they are a specific algorithm that is appropriate in some conditions.

In this study, the best equation for computing modular exponential with sub exponent is determined. In this work, the modulus is assumed to be the product of two primes. Therefore, two modular exponentiations with two sub exponents are necessary. For each modular exponentiation, the best equation is determined using one of four algorithms. Consequently, it is feasible that the equation chosen to compute modular exponentiation with the first sub exponent is distinct from the equation with the second sub exponent. Moreover, there are three parameters to consider the best equation, modular multiplications, modular squares and modular inverses. In addition, it means that the proposed method for selecting the two best equations requires the shortest computing time in the decryption process.

2 Related Works

This section provides an overview of RSA and discusses strategies for accelerating RSA's decryption process.

RSA

RSA is one of the most widely known algorithms for public key cryptography. In 1978, three scholars, R. Rivest, A. Shamir, and L. Adleman, created this algorithm. In addition, this method is compatible with both data encryption and digital signature. Although it has already been demonstrated that RSA can be broken rapidly using Shor's algorithm (Shor, P.W., 1994), which was proposed by P. Shor in 1994, Shor's method is based on the under-development quantum circuit. Thus, RSA is still typically utilized today. To implement RSA for data encryption, there are three techniques.

Process 1 (Key Generation Process): The purpose of this procedure is to generate all RSA implementation parameters. The five steps are as follows:

Step 1: Generating two random prime numbers, p and q
Step 2: Computing the modulus, n, from $n = p*q$
Step 3: Computing Euler's totient function, $\Phi(n)$, from $\Phi(n) = (p-1)*(q-1)$
Step 4: Selecting pubic key, e, from the following conditions, $1 < e < \Phi(n)$ and $\gcd(e, \Phi(n)) = 1$
Step 5: Computing the private key, d, from $d = e^{-1} \bmod \Phi(n)$

After finishing the Key Generation Process, all parameters are separated into two distinct groups. The first category consists of published parameters e and n. Both are sent to senders who wish to transmit a secret message to recipients (owners' key). The second group is for the private parameters which are p, q, d and $\Phi(n)$.

Process 2 (Encryption Process): It is the method by which senders transmit a secret message to receivers by encrypting the plaintext prior to transmission through an insecure channel. The equation for encryption is as follows:

$$c = m^e \bmod n \tag{1}$$

where m is original plaintext and c is ciphertext

Process 3 (Decryption Process): Using this method, the recipient is able to recover the plaintext. The equation for decryption is as follows:

$$m = c^d \bmod n \tag{2}$$

In practice, the disadvantage is the time required to complete the encryption and decryption processes. Because it is the case that all parameters are available on this side, the decryption process can be sped up by employing numerous improved methods. In section 2.3, these algorithms will be discussed.

Fast Exponent

In fact, modular exponentiation is required to solve both above equations. However, this procedure has a very high computational cost, especially due to the exponent's high value. Fast exponent (Abdalhossein, R., 2015) (Chia, L.W., 2006) is the method for accelerating modular exponentiation. The exponent is converted to a binary value for modular multiplication computation. In fact, modular multiplication is necessary when a bit equals one. The other essential operation in Fast Exponent is modular square, which is based on the exponent's bit length. Assigning $b = \sum_{i=0}^{len-1} b_i * 2^i$ where len is bits length of b, Algorithm 2.1 is the procedure of Fast Exponent.

Algorithm 2.1 Fast Exponent

INPUT: $a, c, b_0, b_1, \cdots, b_{len-1}$

OUTPUT: $d = a^b \bmod c$

```
r ← 1, t ← a, i ← 1 and x ←0
IF b0 = 1 Then
    r ← a
EndIF
While i < len Do
    t ← t^2 mod c
    IF bi = 1 Then
        r ← r * t mod c
    EndIF
    i ← i + 1
End While
d ← r
```

Algorithms to Speed Up RSA's Decryption Process

Given that all RSA parameters are known, this section discusses the enhanced methods that speed up the decryption process while using RSA.

a) **Chinese Remainder Theorem (CRT):** CRT is a method for accelerating RSA that divides d into two numbers that are both less than d. Although two modular exponentiations are necessary to complete the process, each equation can be solved in less time than it takes to explicitly retrieve m from equation (2). Given that d_p and d_q are two exponents derived from d, the equations (3) and (4) are chosen to determine d_p and d_q, respectively.

$$d_p = d \bmod p - 1 \tag{3}$$

$$d_q = d \bmod q - 1 \tag{4}$$

In fact, they are the exponents needed to determine m_p and m_q, given the equations (5) and (6).

$$m_p = c^{d_p} \bmod p \tag{5}$$

$$m_q = c^{d_q} \bmod q \tag{6}$$

Assigning $y_p = p^{-1} \bmod q$ and $y_q = q^{-1} \bmod p$, m can be recovered by using equation (7)

$$c = (m_p y_q q + m_q y_p p) \bmod n \tag{7}$$

Because d_p and d_q are very tiny compared to d, this indicates that the time required to recover m is minimized.

b) **The efficient method for the large private key (Somsuk, K., 2017) (Somsuk, K., 2018):** It is a modified approach appropriate for a large private key. However, when d is small, this strategy becomes inefficient. Consequently, it should only be taken for implementation when d is small. Assuming d_p and d_q are large, CRT can be used to minimize calculation time on the decryption side.
Assuming $x_p = p - d_p$ and $x_q = q - d_q$ are assigned, then m_p and m_q may be calculated using equations (8) and (9).

$$m_p = (c^{-1})^{x_p - 1} \bmod p \tag{8}$$

$$m_q = (c^{-1})^{x_q - 1} \bmod q \tag{9}$$

In fact, m_p may be determined quite rapidly when x_p is little. This event will occur when d_p has a tremendous value. The same reason why the procedure to find m_q is quick when x_q is small.

c) **The efficient method when the cluster of all bits which are equal to one is large (Somsuk, K., 2021):** Before applying Algorithm 2.2 to get m, d will be converted into the difference of two integers, d = a - b, where $a = 2^{len}$ and len is the bit length of d. Assuming $f = f_{len-1}f_{len-2}\cdots f_2f_1f_0$ is computed from the relation between a and b, see in (Somsuk, K., 2021), the process to recover m is shown in Algorithm 2.2:

Algorithm 2.2 The Improved Fast Exponent

INPUT: n, c, $f = f_{len-1}f_{len-2}\cdots f_1f_0$

OUTPUT: $m = c^d \bmod n$

```
l ← Length of f, t ← c, m_a ← 1, m_b ← t, i ← 1
While i <= len - 1 Do
    t ← t^2 mod n
    IF f_i = 1 Then
        m_a ← m_a * t mod n
    Else IF f_i = 2 Then
        m_b ← m_b * t mod n
    EndIF
    i ← i + 1
End While
r ← (m_b)^-1 mod n
m ← m_a * r mod n
```

Moreover, this approach is highly effective when the number of clusters is limited. In fact, each cluster is comprised of neighboring bits 0 that are grouped together. Additionally, this approach may be utilized with CRT to reduce computing time. Assigning len_p is bits length of d_p, len_q is bits length of d_q, $d_p = a_p - b_p$ and $d_q = a_q - b_q$ where $a_p = 2^{len_p}$ and $a_q = 2^{len_q}$, then m_p and m_q can be computed by using the equation (10) and (11) respectively.

$$m_p = c^{a_p} * (c^{b_p})^{-1} \bmod p \tag{10}$$

$$m_q = c^{a_q} * (c^{b_q})^{-1} \bmod q \tag{11}$$

When the number of clusters is minimal, the equation (10) can compute m_p extremely quickly. In addition, m_q is quickly determined using equation (11) when the number of clusters is minimal.

d) **The efficient method by using the new private key with low Hamming Weight (Somsuk, K., 2022):** The private key is often allocated a big value to prevent quick attacks by attackers. In addition, there is a significant probability that the huge private key will have a High Hamming Weight, which is the amount of bits that equal 1. The number of modular multiplications is determined by Hamming Weight. The purpose of the procedure in (Somsuk, K., 2022) is to offer a new private key that is mathematically distinct from the standard private key, and which may have a lower Hamming Weight. Therefore, the computation time is reduced if a new key with a low Hamming Weight is constructed. In addition, if a new private key with a low Hamming Weight is generated, this technique can be used with CRT to accelerate the RSA decryption procedure (Kim, H., 2013).

Assigning $d_{lp} = d_p + a*(p-1)$ and $d_{lq} = d_q + b*(q-1)$ where $a, b \in \mathbb{Z}^+$, then m_p and m_q can be computed by using the equation (12) and (13) respectively.

$$m_p = c^{d_{lp}} \bmod p \tag{12}$$

$$m_q = c^{d_{lq}} \bmod q \tag{13}$$

Assume that the bit length of d_{lp} is extremely near to d_p and d_{lp}'s Hamming Weight is less than d_p, the cost of computing modular multiplication using equation (12) is reduced. Moreover, analyzing the cost with the equation (13) is equivalent to considering the equation (12).

3 The Proposed Method

The purpose of this research is to offer a method that considers the best equations for computing both m_p and m_q to speed up the decryption procedure. The algorithm is separated into two sections. The first step is to select the best equation to compute m_p; the competitors are Equation (5), Equation (8), Equation (10) and Equation (12). In addition, the competitors for the second part to find the winner to compute m_q are the equation (6), the equation (9), the equation (11) and the equation (13). In fact, after the winners for the first and second parts have been determined, both of them are selected to complete RSA's decryption procedure, which indicates that these equations save the most on computation costs.

Given that the costs to complete modular exponentiation are modular squares, modular multiplications, and modular inverses, Algorithm 3.1 and Algorithm 3.2 demonstrate how to choose the best equations to compute m_p and m_q, respectively.

Algorithm 3.1 The best equation to find m_p

INPUT: n, c, p, q, d_p, x_p, a_p, b_p and d_{lp}

OUTPUT: the equation to compute m_p

1. $C_{p\_1} \leftarrow$ Costs to compute m_p by using the equation (5)
2. $C_{p\_2} \leftarrow$ Costs to compute m_p by using the equation (8)
3. $C_{p\_3} \leftarrow$ Costs to compute m_p by using the equation (10)
4. $C_{p\_4} \leftarrow$ Costs to compute m_p by using the equation (12)
5. $C \leftarrow 1$
6. IF $C < C_{p\_1}$
7. $C \leftarrow C_{p\_1}$
8. IF $C < C_{p\_2}$
9. $C \leftarrow C_{p\_2}$
10. IF $C < C_{p\_3}$
11. $C \leftarrow C_{p\_3}$
12. IF $C < C_{p\_4}$
13. $C \leftarrow C_{p\_4}$
14. The Equation to Find m_p is based on C

After completing Algorithm 3.1, the selected equation to compute m_p is the optimal approach.

In addition, Algorithm 3.2 is selected for the computation of m_q.

Algorithm 3.2 The best equation to find m_q

INPUT: n, c, p, q, d_q, x_q, a_q, b_q and d_{lq}

OUTPUT: the equation to compute m_p

1. $C_{q\_1} \leftarrow$ Costs to compute m_q by using the equation (6)
2. $C_{q\_2} \leftarrow$ Costs to compute m_q by using the equation (9)
3. $C_{q\_3} \leftarrow$ Costs to compute m_q by using the equation (11)
4. $C_{q\_4} \leftarrow$ Costs to compute m_q by using the equation (13)
5. $D \leftarrow 1$
6. IF $D < C_{q\_1}$
7. $D \leftarrow C_{q\_1}$
8. IF $D < C_{q\_2}$
9. $D \leftarrow C_{q\_2}$
10. IF $D < C_{q\_3}$
11. $D \leftarrow C_{q\_3}$
12. IF $D < C_{q\_4}$
13. $D \leftarrow C_{q\_4}$
14. The Equation to Find m_q is based on D

After completing the procedure, the most efficient equation to compute m_q is chosen. Therefore, it suggests that the time required to complete the work on the decryption side has decreased.

Assigning M is number of modular multiplications

S is number of modular squares

I is number of modular inverses

Example 1 and Example 2 demonstrate that when the best equations for computing m_p and m_q are revealed, the cost of calculation is lowered.

Example 1 Assigning p =215893, q = 191027, n = 41241392111, $\Phi(n)$ = 41240985192, e = 31287190865 and d = 18643694825, find the best equations to compute m_p and m_q

Sol: First, the equations to compute m_p are consider

1.1 Choosing the equation (5), $d_p = 125273 = 11110100101011001_2$, then all costs to compute m_p are 9M and 16S
1.2 Choosing the equation (8), $x_p - 1 = 90619 = 10110000111111011_2$, then all costs to compute m_p are 10M 16S and 1I
1.3 Choosing the equation (10), f_p = 100021201212102012, then all costs to compute m_p are 11M 17S and 1I, using Algorithm 2.2
1.4 Choosing the equation (12), the new private key which is the best exponent is, $d_{lp} = 557057 = 10001000000000000001_2$ with a = 2, then all costs to compute m_p are 2M and 19S

Table 1: Computation Costs to Compute mp for Example 1

Equation to find m_p	**Computation Costs**			**Total**
	M	**S**	**I**	
Equation (5)	9	16	-	25
Equation (8)	10	16	1	27
Equation (10)	11	17	1	29
Equation (12)	2	19	-	**21**

Table 1 displays the costs for each equation used in Example 1 to calculate m_p. It suggests that the equation (12) with d_{lp} as the exponent is the best equation for calculating m_p.

The following step is to consider the best equation to compute m_q.

2.1 Choosing the equation (6), $d_q = 130303 = 11111100111111111_2$, then all costs to compute m_q are 14M and 16S
2.2 Choosing the equation (9), $x_q - 1 = 60723 = 1000011000000110 1_2$, then all costs to compute m_q are 9M 15S and 1I
2.3 Choosing the equation (11), f_q = 100000020100000002, then all costs to compute m_q are 3M 17S and 1I, using Algorithm 2.2
2.4 Choosing the equation (13), the new private key which is the best exponent is, $d_{lq} = 321329 = 1001110011100110001_2$ with b = 1, then all costs to compute m_q are 9M and 18S

Table 2: Computation Costs to Compute mq for Example 1

Equation to find m_q	**Computation Costs**			**Total**
	M	**S**	**I**	
Equation (6)	14	16	-	30
Equation (9)	9	15	1	25
Equation (11)	3	17	1	**21**
Equation (13)	9	18	-	27

Table 2 displays the costs for each equation used to determine m_q in Example 1. It indicates that equation (11) with f_q as the exponent is the best equation for computing m_q.

Consequently, the decryption procedure takes just 5M, 36S, and 1I if the proposed method is used to select the best equations for m_p and m_q. However, 23M and 32S are required when CRT is used.

In addition, the following example illustrates the calculation time required by each strategy to complete the RSA decryption process with a modulus length of 1024 bits.

Example 2 Assigning

p = 6835952817751931056288719508748250083888114156174801897722150005506243706772022163 3652432327037202609350019219452966481218745314536697897520826196695305561,

q = 8282951339977371559557116862936904963731156092285127429793792505171552713870940944 769284423758721851632533334123609971098438483697490486995960015535271683,

n = 5662186455182044617906740594633781463793059283654028550826283333830870632788608396 1808895635420415159791204480512204616368922595132360925475875917037696182748869934 5854084178933948099880131041351193344938955604891413475324742713753346690309076671 03717989042746914486625711825006310437901015162076942206404163,

$\Phi(n)$ = 5662186455182044617906740594633781463793059283654028550826283333830870632788608396 1808895635420415159791204480512204616368922595132360925475875917037696167629965776 8561058020475584383028580565158490860339662329731988368546778507323715608963800106 41275876475211658417719092604693295286740738414034307001601920,

e = 1327736579018352958045134474888367291083082145048327158995115515139773807483097603 2637033988947012540711803646999696972630925170489558220153151547718188397414403643 9175315923192600321302183208370868131320630919246598973076878530168184759211069909 47117319034514172302776978479151914781725335621552189140484449 and

d = 1842184205992115214210473082324629961272904739257012898984490107133717161910923963 4160790541776065994676073483482706709256343672940973616349646825006881589903758186 6249183790361515690932723867605503207740396301601165063255853765574262832221276052 16769126310742318158415501664530000915032528664645672079759809, find the best equations to compute m_p and m_q

Sol: First, the equations to compute m_p are consider

1.1 Choosing the equation (5), d_p is the exponent, then all costs to compute m_p are 255M and 510S
1.2 Choosing the equation (8), $x_p - 1$ is the exponent, then all costs to compute m_p are 260M 506S and 1I
1.3 Choosing the equation (10), f_p is the parameter for Algorithm 2.2, then all costs to compute m_p are 255M 511S and 1I, using Algorithm 2.2
1.4 Choosing the equation (12), d_{lp} with a = 1 is t the best exponent, then all costs to compute m_p are 6M and 512S

Table 3: Computation Costs to Compute mp for Example 2

Equation to find m_p	**Computation Costs**			**Total**
	M	**S**	**I**	
Equation (5)	255	510	-	765
Equation (8)	260	506	1	767
Equation (10)	255	511	1	767
Equation (12)	6	512	-	**518**

Table 3 displays the costs associated with each equation used in Example 2 to determine m_p. It suggests that the equation (12) with d_{lp} as the exponent is the best equation for calculating m_p and that just 518 expenses are necessary.

The next process is to consider the best equation to compute m_q

2.1 Choosing the equation (6), d_q is the exponent, then all costs to compute m_q are 278M and 509S
2.2 Choosing the equation (9), $x_q - 1$ is the exponent, then all costs to compute m_q are 255M 510S and 1I
2.3 Choosing the equation (11), f_q is the parameter for Algorithm 2.2, then all costs to compute m_q are 15M 510S and 1I, using Algorithm 2.2
2.4 Choosing the equation (13), d_{lq} with b = 2 is the best exponent, then all costs to compute m_q are 251M and 512S

Table 4: Computation Costs to Compute mq for Example 2

Equation to find m_q	**Computation Costs**			**Total**
	M	**S**	**I**	
Equation (6)	248	509	-	757
Equation (9)	255	510	1	766
Equation (11)	15	510	1	**525**
Equation (13)	251	512	-	763

Table 4 contains the costs for each equation used to determine m_q in Example 2. It indicates that equation (11) with f_q as the exponent is the best equation for computing m_q.

Hence, the decryption procedure takes just 21M, 1022S, and 1I if the proposed method is used to select the best equations for m_p and m_q. However, 503M and 1019S are required when CRT is used.

4 Experimental Results

Table 5: Time to Finish Decryption Process in Example 2

Algorithm	**Time (mSec)**
CRT	38.47
Applying CRT with the Method in (Somsuk, K., 2017)	39.15
Applying CRT with the Method in (Somsuk, K., 2018)	29.87
Applying CRT with the Method in (Somsuk, K., 2021)	30.15
The Proposed Method	**21.21**

In this section, the computation time required by each technique to complete RSA decryption process is shown. CRT is performed to all comparison techniques to determine the best time. Fast Exponent is also provided for computing modular exponentiation. In fact, the Improved Fast Exponent, Algorithm 2.2, is the only way in (Somsuk, K., 2018) that Fast Exponent cannot include to complete modular exponentiation. Furthermore, the BigInteger class in Java, which can represent an unbounded size variable, is selected for the implementation. All experiments were conducted on a 2.53 GHz Intel® Core i5 with 8 GB of RAM so that the same parameters could be controlled.

The experiment consists of two distinct parts. Consider the computation time required to compute the RSA decryption process in Example 2. The second experiment involves randomly generating all RSA parameters with bit lengths between 1024 and 4098 and evaluating the time required to do the same task.

Table 5 compares the time to complete modular exponentiation based on the parameters in Example 2. The experimental findings demonstrate that the proposed method imposes the least amount of computation costs. Because the most effective equations for computing m_p and m_q are chosen, the proposed method becomes the most effective algorithm. In practice, the equation used to calculate m_p may differ from the equation used to determine m_q.

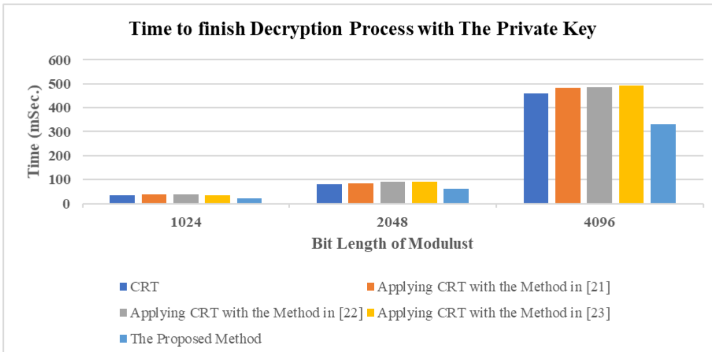

Figure 1: Time to finish Decryption Process with The Private Key

In figure 1, a comparison of the time required to complete the decryption process is presented. In fact, the computation time for each bit length is determined by the mean of 20 modulus values. These experimental findings demonstrate that the proposed method is the most efficient. In general, CRT, Applying CRT with the Method in (Somsuk, K., 2017), Applying CRT with the Method in (Somsuk, K., 2018), and Applying CRT with the Method in (Somsuk, K., 2021) are utilized to get the best equation. Therefore, it implies that the proposed method is always the winner. In addition, the proposed method completes the procedure 10 to 30 percent faster than CRT on average.

5 Conclusion

The private key is analyzed to select the best equations to find m_p and m_q before using Chinese Remainder Theorem (CRT) to recover m. In fact, three parameters are chosen to consider the best equations, modular multiplications, modular squares, and modular inverses. In addition, it implies that computation time to decrypt the ciphertext is certainly the lowest in comparison to the other techniques. The experimental result shows that the time is reduced about 10 to 30 percent when proposed method is compared with CRT.

References

[1] Abdalhossein, R. & Parviz, K. (2015). Design of RSA Processors Based on High-Radix Montgomery Multipliers, *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, 23, 1710 - 1719.

[2] Atsushi, M., Naofumi, H., Takafumi, A. & Akashi, S. (2011). Systematic Design of RSA Processors Based on High-Radix Montgomery Multipliers, *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, *19*, 1136-1146.

[3] Chia, L.W. (2006). Fast Parallel Exponentiation Algorithm for RSA Public-Key Cryptosystem, *INFORMATICA*, 17, 445-462.

[4] Christian, E., Esteban, A., Jorge, L.G., Eduardo, V., Gabriel, S. & Juan, G.A. (2021). A low-cost and highly compact FPGA-based encryption/decryption architecture for AES algorithm. *IEEE Latin America Transactions*., *19*, 1443-1450.

[5] Diffie, W. & Hellman, M.E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*, 644–654.

[6] Guang, G. & Golomb, S.W. (1999). Transform domain analysis of DES. *IEEE Transactions on Information Theory*., *65*, 2065 – 2079.

[7] Guanglou, Z., Gengfa, F., Rajan, S. & Mehmet, A.O. (2015). Encryption for Implantable Medical Devices Using Modified One-Time Pads. *IEEE Access*., *3*, 825 - 836.

[8] Gupta, S.C. & Manju, S. (2021). Matrix Modification of RSA Digital Signature Scheme. *Journal of Applied Security Research*, *16*, 63-70.

[9] Issam, H., Kamal, E.S. & Ezz, E.M. (2010). High-Speed AES Encryptor With Efficient Merging Techniques. *IEEE Embedded Systems Letters*., *2*, 67-71.

[10] Kim, H., Choi, Y., Choi, D., & Ha, J. (2013). A New Exponentiation Algorithm Resistant to Combined Side Channel Attack. *Journal of Internet Services and Information Security*, *3*(3-4), 17-27.

[11] Nedal, T. & Emad, A. (2018). Hybrid Publicly Verifiable Authenticated Encryption Scheme Based on Chaotic Maps and Factoring Problems. *Journal of Applied Security Research*, *13*, 304 - 314.

[12] Omar, K. (2008). Algorithm for factoring some RSA and Rabin moduli, *Journal of Discrete Mathematical Sciences and Cryptography*, *11*, 537 – 543.

[13] Pollard, J. M. (1978). Monte Carlo methods for index computation ($mod p$). *Mathematics of computation*, *32*(143), 918-924.

[14] Rivest, R.L., Shamir, A. & Adleman, L. (1978). A method for obtaining digital signatures and public key cryptosystems. *Communications ACM*, *21*, 120–126.

[15] Shor, P.W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *In IEEE Proceedings 35th annual symposium on foundations of computer science*, 124-134.

[16] Somsuk, K. (2017). The New Equation for RSA's Decryption Process Appropriate with High Private Key Exponent. *In IEEE 21st International Computer Science and Engineering Conference (ICSEC)*, 1-5.

[17] Somsuk, K. (2018). The improvement of initial value closer to the target for Fermat's factorization algorithm. *Journal of Discrete Mathematical Sciences and Cryptography*, *21*(7-8), 1573 – 1580.

[18] Somsuk, K. (2020). Improving the initial values of VFactor suitable for balanced modulus, *International Journal of Electrical and Computer Engineering*, *10*, 6446 - 6452.

[19] Somsuk, K. (2021). The alternative Method to Finish Modular Exponentiation and Point Multiplication Processes. *KSII Transactions on Internet and Information Systems (TIIS)*, *15*(7), 2610-2630.

[20] Somsuk, K. (2021). The Improvement of Elliptic Curve Factorization Method to Recover RSA's Prime Factors. *Symmetry*., *13*, 1–15.

[21] Somsuk, K. (2022). Efficient Variant of Pollard's $p - 1$ for the Case That All Prime Factors of the $p - 1$ in B-Smooth. *Symmetry*., *14*, 1–17.

[22] Somsuk, K. (2022). The alternative method to speed up RSA's decryption process. *Journal of Discrete Mathematical Sciences and Cryptography*, 1-22.

[23] Somsuk, K., Chiawchanwattana, T., & Sanemueang, C. (2018). Speed up RSA's Decryption Process with Large sub-Exponents using Improved CRT. *In IEEE International Conference on Information Technology (InCIT)*, 1-5.

[24] Sung, M.Y., Seungjoo, K., Seongan, L. & Sang, J.M. (2003). RSA speedup with Chinese remainder theorem immune against hardware fault cryptanalysis, *IEEE Transactions on Computers*, *52*, 461-472.
[25] Wu, M.E., Tso, R. & Sun, H.M. (2014). On the improvement of Fermat factorization using a continued fraction technique, *Future Generation Computer Systems*, *30*(1), 162 – 168.

Authors' Biography

Kritsanapong Somsuk is an associate professor at the Department of Computer and Communication Engineering, Faculty of Technology, Udon Thani Rajabhat University, Udon Thani, Thailand. He obtained his M.Eng. (Computer Engineering) from Department of Computer Engineering, Faculty of Engineering, Khon Kaen University, M.Sc. (Computer Science) from Department of Computer Science, Faculty of Science, Khon Kaen University and his Ph.D. (Computer Engineering) from Department of Computer Engineering, Faculty of Engineering, Khon Kaen University. The area of research interests includes computer security, cryptography and integer factorization algorithms. He can be contacted at email: kritsanapong@udru.ac.th.

Sarutte Atsawaraungsuk received the B.Sc. and M. Sc degree in Computer Science, and the Ph.D. degree in Computer Engineering from the Khon Kaen University, Thailand. Currently, he is an assistant professor at the Department of Computer Education, Faculty of Education, Udon Thani Rajabhat University, Udon Thani, Thailand. His research interests in the Machine learning, image processing and its application. He can be contacted at email: sarutte@udru.ac.th.

Chanwit Suwannapong received a B.Eng., M.Eng. and Ph.D. degree in Computer Engineering from Khon Kaen University, Thailand. Currently, he is an assistant professor at the Department of Computer Engineering, Faculty of Engineering, Nakhon Phanom University, Nakhon Phanom, Thailand. He has published many publications in the area of Wireless Sensor Network, Ad Hoc Networks and Smart Agriculture. He can be contacted at email: schanwit@npu.ac.th.

Suchart Khummanee received the B.Eng. degree in Computer Engineering from the King Mongkut's Institute of Technology Ladkrabang, the M.Sc. degree in Computer Science from the Khon Kaen University, and the Ph.D. degree in Computer Engineering from the Khon Kaen University, Thailand. He is currently a full lecturer of Computer Science at the Mahasarakham University, Thailand. He can be contacted at email: suchart.k@msu.ac.th.

Chalida Sanemueang obtained a bachelor of arts in German as a second language from the faculty of Humanities and Social Science, Khon Kaen University and received a master of arts in European Studies (International Disciplinary Program) from Chulalongkorn University, Thailand. Currently, she is a lecturer at Language Center, Udon Thani Rajabhat University. The research interests focus on German as a second language, English as a Medium Instruction and spoken language systems. She can be contacted at email: Chalida.sa@udru.ac.th.