Copyright ⓒ 2022 KSII

The Improved Estimation of the Least Upper Bound to Search for RSA's Private key

Kritsanapong Somsuk*
Department of Computer and Communication Engineering, Faculty of Technology,
Udon Thani Rajabhat University,UDRU, Udon Thani, Thailand.
[e-mail: kritsanapong@udru.ac.th]
*Corresponding author: Kritsanapong Somsuk

Received January 24, 2021; revised February 18, 2022; accepted May 31, 2022; published June 30, 2022

Abstract

RSA is known as one of the best techniques for securing secret information across an unsecured network. The private key which is one of private parameters is the aim for attackers. However, it is exceedingly impossible to derive this value without disclosing all unknown parameters. In fact, many methods to recover the private key were proposed, the performance of each algorithm is acceptable for the different cases. For example, Wiener's attack is extremely efficient when the private key is very small. On the other hand, Fermat's factoring can quickly break RSA when the difference between two large prime factors of the modulus is relatively small. In general, if all private parameters are not disclosed, attackers will be able to confirm that the private key is unquestionably inside the scope [3, n – 2], where n is the modulus. However, this scope has already been reduced by increasing the greatest lower bound to [d_{il}, n – 2], where $d_{il} \geq 3$. The aim of this paper is to decrease the least upper bound to narrow the scope that the private key will remain within this boundary. After finishing the proposed method, the new scope of the private key can be allocated as [d_{il}, d_{ir}], where $d_{ir} \leq n$ – 2. In fact, if the private key is extremely close to the new greatest lower bound, it can be retrieved quickly by performing a brute force attack, in which d_{ir} is decreased until it is equal to the private key. The experimental results indicate that the proposed method is extremely effective when the difference between prime factors is close to each other and one of two following requirement holds: the first condition is that the multiplier of Euler totient function is very close to the public key's small value whereas the second condition is that the public key should be large whenever the multiplier is far enough.

Keywords: RSA, The least upper bound**,** The greatest lower bound**,** The private key**,** prime factors

1. Introduction

At present, data transfer over network channel is the preferred method due to its convenience and speed. However, network channel is referred as the unsecured channel. In other word, if the secret data being transferred across a network channel is very important and there is no securing algorithm in place to protect it, then attackers can simply entrap the data. Prior to transferring any important or private information over an unsecured connection, it should be encased by some securing algorithms. Cryptography [1] is one of the beneficial techniques for protecting data prior to transmission via an unsecured channel by using encryption and decryption processes. The concept of using cryptography to secure the information is as follows: First, the secret message which is called the plaintext is encrypted by using the secret key. In fact, the encrypted message is referred to the ciphertext and it is transmitted to the recipient instead of the secret message. On receiver side, after the ciphertext has arrived, the original plaintext can be recovered using the secret key and a decryption process. In general, cryptography is classified into two broad categories. Each type has its own unique advantages and disadvantages. The first is symmetric key cryptography which uses the same key, referred to the secret key for both the encryption and decryption. The advantage is related to time required to complete the process. On the other hand, the disadvantage refers to the method in which an attacker might avoid exchanging the secret key between senders and receivers. Asymmetric key cryptography, on the other hand, is also known as public key cryptography. The key concept is to the different keys that must be chosen to encrypt the plaintext and decrypt the ciphertext. One key which is kept secretly is called the private key and the other key which is disclosed to everyone is called the public key. In fact, these keys are mathematically related to each other. The advantage is that the secret channel to exchange the key is eliminated. However, the disadvantage is that the procedure is quite time consuming to complete the process. Therefore, both of symmetric key cryptography and public key cryptography are integrated in the real situation in order to increase the performance.

RSA [2] is the best well-known public key cryptography. It is chosen to apply with a variety of applications to secure the information such as [3], [4], [5], [6], [7], [8]. However, to avoid intruder attacks, the modulus bit length (n) should be assigned at least 1024 bits. Therefore, one of the two keys must be a large integer. In fact, in order to ensure that RSA is extremely secure, the private key (d) should be large. Although, RSA is still difficult to be broken (all parameters are strong), many algorithms can break RSA when certain parameters are weak. Moreover, many weak parameters are disclosed such as a low private key [9], a high private key [10], a low prime factor [11] and the small distance between two prime factors [12]. In this paper, two weak parameters are presented. The first weakness is the small public key and the short distance between it and the multiplier of Euler totient function. The other weakness is the large public key and the high distance between the high public key and the multiplier of Euler totient function. In addition, there should be a little difference between two prime factors. Moreover, the new method for estimating initial value of the private key that effectively addresses both weaknesses is also proposed.

Assuming that all private parameters are hidden and if intruders want to recover d, they will clearly recognize that the position of d must be located within the following scope: $d \in [3, n - 2]$. This scope is costly since the length of n is allocated at least 1024 bits. If brute force attack is selected to find d, then the cost of computation is quite high. The method in [13] was proposed to estimate the new greatest lower bound as $[d_{il}, n - 2]$, where $d_{il} \geq 3$. Therefore, this method can recover d rapidly if d is a small integer. However, it is still inefficient when d is a large integer. In fact, the aim of the proposed method is to estimate the new least upper

bound as [3, d_{ir}], when $d_{ir} \leq n - 2$. Therefore, if the proposed method is combined with the method in [13], the scope is narrowed, or $d \in [d_{il}, d_{ir}]$.

2. Related Works

In this section, the overviews of RSA and many techniques to break this algorithm will be mentioned.

2.1 RSA Scheme

RSA [2] is the best well-known public key cryptography. It was proposed by R. Rivest, A. Shamir and L. Adleman in 1977. In fact, RSA's name is derived from the initial letter of their surnames. RSA is administered by three main processes. The first process is key generation which is the process to create a pair of keys. The procedures in this process are as follows: it is begun by generating two large prime numbers, p and q where $p < q$, randomly to find the modulus, $n = p*q$, and Euler totient function, $\Phi(n) = (p-1)*(q-1)$. The next step is to select the public key, e, that must be in the following condition: $1 < e < \Phi(n)$ and $\gcd(e, \Phi(n)) = 1$. After, e is found, the private key, d, can be computed from $e*d \bmod \Phi(n) = 1$ by using Extended Euclidean Algorithm such as [14], [15], [16]. The second process is the encryption process, which is the process of converting the original plaintext into the ciphertext. The equation is $c = m^e \bmod n$, where m is the original plaintext and c is the ciphertext. The last process is the decryption process for recovering the original plaintext by using the following equation: $m = c^d \bmod n$.

Private parameters for RSA scheme are p, q, d and $\Phi(n)$. Therefore, intruders must locate at least one of them in order to recover m. In fact, for the real situation, they are all strongly allocated to prevent easily attacks. However, RSA may be broken when at least one of the private parameters is vulnerable. In the next topic in the related works section discusses an overview of several attacking algorithms is provided, highlighting how the performance of each algorithm is customized to the various weak points.

2.2 Overviews of some algorithms to break RSA

In fact, there are a variety of algorithms that can be chosen to break RSA whenever its vulnerabilities are discovered.

Wiener's attack [17] is the technique which was presented by Michael J. Wiener in 1990 to recover d. His theorem relies mostly on the continuous fraction as its mathematical basis. In addition, Wiener's attack is particularly effective when $d < \frac{1}{3}n^{\frac{1}{4}}$. Therefore, it implies that the small private key is the weak point of RSA.

Besides, in 1999, D. Boneh and G. Durfee [18] presented the modified version of Wiener's attack. Although, $d > \frac{1}{3}n^{\frac{1}{4}}$, d is still rapidly recovered whenever $d < n^{0.292}$. In order to avoid trapping d by using this method, d should be assigned larger than $n^{0.292}$.

In 2017, the technique [10] was presented to accelerate the decryption process of RSA by utilizing the new exponent. In fact, this method has highly performance when d is large, because the new exponent which is mathematically related with d and $\Phi(n)$ becomes small. On the other hand, if d is too large, this method may be selected to find d.

Trial division algorithm (TDA) is the method for discovering p and q by for factoring n. It is divided into two techniques. The first way [19] is to select 3 as the initial divisor which is always increased by two when the remainder exists. This approach can quickly discover p and q when p is a small integer, as it is particularly efficient when p is small. Therefore, the small prime factor is also a weakness of RSA. However, $\lfloor \sqrt{n} \rfloor$ is chosen as the first divisor for the second technique [20] and it is always decreased when there is the remainder. Therefore, if p is extremely close to $\lfloor \sqrt{n} \rfloor$, it can be recovered very fast.

Fermat's Factorization algorithm (FFA) [21] is the factoring method which was proposed by Pierre de Fermat. A composite integer with two prime factors can be represented as the difference between two perfect square integers. In fact, when two prime factors are close in proximity to one another, they can be computed rapidly. Furthermore, other FFA-improved algorithms such as [22], [23], [24], [25], [26] were proposed to increase computation speed.

Pollard' s p -1 [27] is the method that was presented by J. Pollard in 1974. Assuming that all prime factors of $p - 1$ or $q - 1$ are a small integer. This method is extremely efficient in recovering both p and q. That is, another weak point is that all prime factors of p - 1 and $q - 1$ must be small.

In 2020, the new methodology [28] to recover d was proposed. Assuming that $\Phi(n) = ad + b$ where $a \mid b$ and $a \mid \Phi(n)$, then the original plaintext can be computed by using the following equation: $m = (c^{-1})^{\frac{b}{a}} \bmod n$. Moreover, this equation can be chosen to apply with brute force attack to find d. The experimental results in the paper showed that when a is large and $\frac{b}{a}$ is small, a short time to complete the operation is required.

The simplest technique is brute force attack. It can be selected to find d directly to locate one of two prime factors. In general, to limit the scope of d, it must first be analyzed. Due to the fact that $\Phi(n)$ is the hidden parameter, the scope of d will be assigned as $[3, n - 2]$ which is a very large range. Moreover, brute force attack can be selected to find d in two ways. The first way is to assign the initial value of d which is equal to the greatest lower bound and it must be increased when it is not certainly the real private key. That mean, this method is suitable for the small private key. On the other hand, the other way is to assign the initial value of d as $n - 2$ and it is decreased when it is not the target. Therefore, this method is suitable for the high private key.

In the beginning of 2021, the method [13] to estimate the new greatest lower bound which is always equal or larger than 3 was proposed. Assuming that d_{il} is represented as the new greatest lower bound, it can be calculated by using the following equation: $d_{il} = \left\lceil \frac{2n-3}{3e} \right\rceil$. Therefore, scope of d is reduced to $[d_{il}, n - 2]$.

2.3 Considering the patterns of p + q

In fact, many methods to analyze the patterns of $p + q$ were proposed. Once all patterns are found, the computation time to recover d is reduced.

Assigning $\mathrm{LSG}_m(z)$ is represented as the last m digits of z and $\mathrm{LSG}(z)$ is represented as the last digit of z when $z \in ¢$. In 2017, the methodology [29] to search for all possible values of $\mathrm{LSG}_m(p)$ and $\mathrm{LSG}_m(q)$ was proposed to find all possible values of $\mathrm{LSG}_m(p + q)$ and $\mathrm{LSG}_m(p -$

q). However, only $LSG_m(p + q)$ is focused in this paper. Assuming that a pair of $LSG_m(p)$ and $LSG_m(q)$ which $LSG_m(LSG_m(p) * LSG_m(q)) = LSG_m(n)$ is found, either rule 1 or rule 2 is selected to find other pairs.

Rule 1: If $LSG(p) = LSG(q)$, $LSG_m((p + 10^{m-1})(q + 9*10^{m-1})) = LSG_m((p + 9*10^{m-1})(q + 10^{m-1})) = LSG_m(n)$.

Rule 2: If $LSG(p) \neq LSG(q)$, there is two odd integers, k_1 and k_2 where $k_1, k_2 = 1, 3, 7$ or 9 and $(LSG(p)*k_2 + LSG(q)*k_1) \bmod 10 = 0$, that $LSG_m((p + k_1*10^{m-1})(q + k_2*10^{m-1})) = LSG_m(n)$.

Example 1: After using Rule 2, all pairs of $LSG_2(p)$ and $LSG_2(q)$ which $LSG_2(LSG_2(p)*LSG_2(q)) = LSG_2(n) = 73$ are found as follows: (07, 39), (17, 69), (27, 99), (37, 29), (47, 59), (57, 89), (67, 19), (77, 49), (87, 79), (97, 09), (01, 73), (11, 43), (21, 13), (31, 83), (41, 53), (51, 23), (61, 93), (71, 63), (81, 33), (91, 03)

After all pairs of $LSG_m(p)$ and $LSG_m(q)$ are found, all possible values of $LSG_m(p + q)$ are also calculated by using $LSG_m(p + q) = LSG_m(LSG_m(p) + LSG_m(q))$.

Example 2: After all pairs of $LSG_2(p)$ and $LSG_2(q)$ which $LSG_2(LSG_2(p)*LSG_2(q)) = LSG_2(n) = 73$ are found, all possible values of $LSG_2(p + q)$ are as follows:

$$S = \{06, 14, 26, 34, 46, 54, 66, 74, 86, 94\}$$

Where S is set of all possible values of $LSG_2(p + q)$ that $LSG_2(LSG_2(p)*LSG_2(q)) = LSG_2(n) = 73$

The benefit of S is that it eliminates irrelevant values of this set from the computation, as $p + q$ may not be the actual result.

In fact, the patterns of $p + q$ can be thoroughly analyzed when m is larger.

Table 1. Patterns of $\frac{p+q}{2}$ that are considered from 3 forms of n

Case	*a*	*b*	*c*	Odd / Even		*x* mod 3 = 0		LSG(*x*)
				Odd	Even	Yes	No	
1	1	1	1	✓			✓	1, 5, 9
2	1	1	9	✓			✓	3, 5, 7
3	1	1	13	✓			✓	3, 7
4	1	1	17	✓			✓	1, 9
5	1	5	1	✓		✓		1, 5, 9
6	1	5	9	✓		✓		3, 5, 7
7	1	5	13	✓		✓		3, 7
8	1	5	17	✓		✓		1, 9
9	3	1	3		✓		✓	2, 8
10	3	1	7		✓		✓	4, 6
11	3	1	11		✓		✓	0, 4, 6
12	3	1	19		✓		✓	0, 2, 8
13	3	5	3		✓	✓		2, 8
14	3	5	7		✓	✓		4, 6
15	3	5	11		✓	✓		0, 4, 6
16	3	5	19		✓	✓		0, 2, 8

Furthermore, in 2016, the patterns of $\frac{p+q}{2}$ [30] are analyzed by considering the forms of n that are divided into three forms. Assuming that $a = n \bmod 4$, $b = n \bmod 6$ and $c = n \bmod 20$, there are 16 patterns of $x = \frac{p+q}{2}$ which are shown in **Table 1**.

After computing three forms of n, the solution will be found in just one of the cases in **Table 1**. Therefore, many steps can be eliminated from the computation. In fact, the type of an even or odd number is disclosed from a, the result of $x \bmod 3$ is disclosed from b and $\mathrm{LSG}(x)$ is known from c.

In [31], the other pattern of $p + q$ is also discovered. It is based on the result of $(n + 1) \bmod 8$. In fact, if the result of $(n + 1) \bmod 8$ is equal to 0, then the result of $(p + q) \bmod 8$ must be also equal to 0. This concept is applicable to a wide variety of brute force attack, as many loops may be omitted when the result of $(n + 1) \bmod 8$ is equal to 0.

3. The Proposed Method

In this paper, the new greatest lower bound, d_{ir}, to find d is proposed. In fact, after the method in [13] was proposed, the position of d must be inside the scope $[d_{il}, n - 2]$. That is, the initial value can be assigned as d_{il} in order to initiate a brute force attack to find d from left to right. On the other hand, this method becomes inefficient when d is large. In this case, using brute force attack to search for d from right to left is more appropriate. That is, the initial value is started as $n - 2$. However, $n - 2$ is still far from d which is always less than $\Phi(n)$. Therefore, in this paper, d_{ir} which is always equal to or less than $n - 2$ is presented.

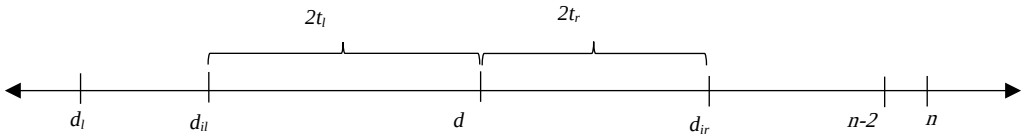

Fig. 1. The position of each parameter on number line

The information in **Fig. 1** shows the parameters associated with RSA on the number line. It implies that d_{ir} is always equal to or less than $n - 2$. Therefore, if this value is found, the scope of d is narrowed in which the position of d is certainly at $[d_{il}, d_{ir}]$.

Before estimating d_{ir}, the integer which is equal or larger than $\Phi(n)$ must be found. Due to the fact that $\Phi(n) = (p - 1)*(q - 1) = n - (p + q) + 1$, the patterns of $p + q$ should be analyzed to estimate the integer which is close to $\Phi(n)$ and is still larger than this value.

From **Table 1**, it implies that the result of $n \bmod 4$ can disclose the result of $\frac{p+q}{2}$ which is an even number or odd number. However, the result of $n \bmod 20$ can also disclose this type. Moreover, the patterns of $\mathrm{LSG}_m(p + q)$ are deeper than the patterns analyzed from $n \bmod 20$. Therefore, three techniques for analyzing the patterns of $p + q$ from $n \bmod 6$, $n + 1 \bmod 8$ and $\mathrm{LSG}_m(n)$ are selected to find the integer in the condition.

In [31], it shows that the result of $(p + q) \bmod 8$ is equal to 0 when $(n + 1) \bmod 8 = 0$. However, if $(n + 1) \bmod 8 \neq 0$, it must be also analyzed to find all possible value of $(p + q) \bmod 8$.

Theorem 1: The results of $(p + q) \bmod 8$ are shown in **Table 2**, The expansion of [31].

Table 2. The result of $(p + q) \bmod 8$

$(n + 1) \bmod 8$	$(p + q) \bmod 8$
0	0
2	2, 6
4	4
6	2, 6

Proof: There are 15 cases as follows:

Case 1: $p \bmod 8 = 1$ and $q \bmod 8 = 1$

Then, $(p*q + 1) \bmod 8 = ((p \bmod 8) * (q \bmod 8) + 1 \bmod 8) \bmod 8$
$= (1*1 + 1) \bmod 8 = 2$

And, $(p + q) \bmod 8 = ((p \bmod 8) + (q \bmod 8)) \bmod 8$
$= (1 + 1) \bmod 8 = 2$

Case 2: $p \bmod 8 = 1$ and $q \bmod 8 = 3$ (The same result with $p \bmod 8 = 3$ and $q \bmod 8 = 1$)

Then, $(p*q + 1) \bmod 8 = (1*3 + 1) \bmod 8 = 4$

And, $(p + q) \bmod 8 = (1 + 3) \bmod 8 = 4$

Case 3: $p \bmod 8 = 1$ and $q \bmod 8 = 5$ (The same result with $p \bmod 8 = 5$ and $q \bmod 8 = 1$)

Then, $(p*q + 1) \bmod 8 = (1*5 + 1) \bmod 8 = 6$

And, $(p + q) \bmod 8 = (1 + 5) \bmod 8 = 6$

Case 4: $p \bmod 8 = 1$ and $q \bmod 8 = 7$ (The same result with $p \bmod 8 = 7$ and $q \bmod 8 = 1$)

Then, $(p*q + 1) \bmod 8 = (1*7 + 1) \bmod 8 = 0$

And, $(p + q) \bmod 8 = (1 + 7) \bmod 8 = 0$

Case 5: $p \bmod 8 = 1$ and $q \bmod 8 = 9$ (The same result with $p \bmod 8 = 9$ and $q \bmod 8 = 1$)

Then, $(p*q + 1) \bmod 8 = (1*9 + 1) \bmod 8 = 2$

And, $(p + q) \bmod 8 = (1+ 9) \bmod 8 = 2$

Case 6: $p \bmod 8 = 3$ and $q \bmod 8 = 3$

Then, $(p*q + 1) \bmod 8 = (3*3 + 1) \bmod 8 = 2$

And, $(p + q) \bmod 8 = (3+ 3) \bmod 8 = 6$

Case 7: $p \bmod 8 = 3$ and $q \bmod 8 = 5$ (The same result with $p \bmod 8 = 5$ and $q \bmod 8 = 3$)

Then, $(p*q + 1) \bmod 8 = (3*5 + 1) \bmod 8 = 0$

And, $(p + q) \bmod 8 = (3+ 5) \bmod 8 = 0$

Case 8: $p \bmod 8 = 3$ and $q \bmod 8 = 7$ (The same result with $p \bmod 8 = 7$ and $q \bmod 8 = 3$)

Then, $(p*q + 1) \bmod 8 = (3*7 + 1) \bmod 8 = 6$

And, $(p + q) \bmod 8 = (3+ 7) \bmod 8 = 2$

Case 9: $p \bmod 8 = 3$ and $q \bmod 8 = 9$ (The same result with $p \bmod 8 = 9$ and $q \bmod 8 = 3$)

Then, $(p*q + 1) \bmod 8 = (3*9 + 1) \bmod 8 = 4$

And, $(p + q) \bmod 8 = (3+ 9) \bmod 8 = 4$

Case 10: $p \bmod 8 = 5$ and $q \bmod 8 = 5$

Then, $(p*q + 1) \bmod 8 = (5*5 + 1) \bmod 8 = 2$

And, $(p + q) \bmod 8 = (5+ 5) \bmod 8 = 2$

Case 11: $p \bmod 8 = 5$ and $q \bmod 8 = 7$ (The same result with $p \bmod 8 = 7$ and $q \bmod 8 = 5$)

Then, $(p*q + 1) \bmod 8 = (5*7 + 1) \bmod 8 = 4$

And, $(p + q) \bmod 8 = (5+ 7) \bmod 8 = 4$

Case 12: $p \bmod 8 = 5$ and $q \bmod 8 = 9$ (The same result with $p \bmod 8 = 9$ and $q \bmod 8 = 5$)

Then, $(p*q + 1) \bmod 8 = (5*9 + 1) \bmod 8 = 6$

And, $(p + q) \bmod 8 = (5+ 9) \bmod 8 = 6$

Case 13: p mod 8 = 7 and q mod 8 = 7
Then, $(p*q + 1) \bmod 8 = (7*7 + 1) \bmod 8 = 2$
And, $(p + q) \bmod 8 = (7+ 7) \bmod 8 = 6$
Case 14: p mod 8 = 7 and q mod 8 = 9 (The same result with p mod 8 = 9 and q mod 8 = 7)
Then, $(p*q + 1) \bmod 8 = (7*9 + 1) \bmod 8 = 0$
And, $(p + q) \bmod 8 = (7+ 9) \bmod 8 = 0$
Case 15: p mod 8 = 9 and q mod 8 = 9
Then, $(p*q + 1) \bmod 8 = (9*9 + 1) \bmod 8 = 2$
And, $(p + q) \bmod 8 = (9+ 9) \bmod 8 = 2$

Therefore, all results of $(p + q) \bmod 8$ are matched with the information in **Table 2**.

In fact, considering the patterns of $(p + q) \bmod 8$, $(p + q) \bmod 3$ and $LSG_m(p + q)$ together are the key to find i which must be selected as the variable in theorem 2.

Theorem 2 Assigning i is represented as the distance between the traditional initial value of $p + q$ which is equal to $2\lceil\sqrt{n}\rceil$ and the improved initial value, then

$$d \le \left\lfloor \frac{1 + (e-1) * (n - 2\lceil\sqrt{n}\rceil - i + 1)}{e} \right\rfloor$$

Proof:
From
$$\Phi(n) = (p-1)*(q-1)$$
$$= pq - (p+q) + 1$$
$$= n - (p+q) + 1$$

In general, $p + q \ge 2\lceil\sqrt{n}\rceil$, see in [12]

Then, $\Phi(n) \le n - (2\lceil\sqrt{n}\rceil) + 1$

However, after using the technique to analyze all patterns of $LSG_m(p + q)$, $(p + q) \bmod 3$ and $(p + q) \bmod 8$, then $p + q \ge 2\lceil\sqrt{n}\rceil + i$, therefore

$$\Phi(n) \le n - (2\lceil\sqrt{n}\rceil + i) + 1$$

Then, $\Phi(n) \le (n - 2\lceil\sqrt{n}\rceil - i + 1)$

From, $ed \bmod \Phi(n) = 1$
Hence $ed = 1 + k\Phi(n)$
Or, $ed \approx k\Phi(n)$

Because, both of e and d is less than $\Phi(n)$, it implies that $k < e$ or $k \le e - 1$

Then, $ed < 1 + e\Phi(n)$
That means, $ed \le 1 + (e-1) * \Phi(n)$

From, $\Phi(n) \le (n - 2\lceil\sqrt{n}\rceil - i + 1)$, then

$$ed \leq 1 + (e-1) * (n - 2\lceil \sqrt{n} \rceil - i + 1)$$

Or,
$$d \leq \frac{1 + (e-1) * \left(n - 2\lceil \sqrt{n} \rceil - i + 1\right)}{e}$$

In fact, d is always an integer,

Therefore,
$$d \leq \left\lfloor \frac{1 + (e-1) * (n - 2\lceil \sqrt{n} \rceil - i + 1)}{e} \right\rfloor$$

Therefore, assigned d_{ir} is the new initial value to find d from right to left, it can be estimated by using the following equation:

$$d_{ir} = \left\lfloor \frac{1 + (e-1) * (n - 2\lceil \sqrt{n} \rceil - i + 1)}{e} \right\rfloor \quad (1)$$

Where, $d \leq d_{ir} \leq n - 2$

Example 3: Assuming n = 167556124362173, e = 3499 and all pairs of $LSG_2(p + q)$ are disclosed as follows: S = {06, 14, 26, 34, 46, 54, 66, 74, 86, 94} (using theorems in [29]), Finding the least upper bound of d by using theorem 2. (The private parameters are p = 24163669, q = 6934217, $\Phi(n)$ = 167556093264288 and d = 134178957795523)

Sol.

Before using the equation in theorem 2, the variable, i, must be found,

First, two forms of n must be checked,

Because (167556124362173 + 1) mod 8 = 6, then the result of $(p + q)$ mod 8 = 2, 6.

Furthermore, $(p + q)$ mod 3 must be equal to 0, because 178719335848973 mod 6 = 5.

Next, the initial value of $p + q$ which is equal to $g = 2\lceil \sqrt{n} \rceil$ must be calculated, $2\lceil \sqrt{167556124362173} \rceil$ = 25888696.

Because 25888696 mod 8 = 0, then g = 25888696 + 2 = 25888698

Because 25888698 mod 3 = 0, then g= 25888698

Because LSG_2(25888698) = 98 is not a member in S, then this value is certainly not the real value of $p + q$. Therefore, it can be reassigned as 25888706. However, 25888706 mod 3 = 2 $\neq$ 0, then the next value of g should be 25888714. In addition, 25888714 mod 3 = 1 $\neq$ 0, then the next value of g should be 25888726. However, 25888726 mod 3 = 1 $\neq$ 0, then the next value of g should be 25888734. In fact, 25888734 mod 3 = 0 and 25888734 mod 8 = 6. Therefore, it is chosen as the new initial value. Furthermore, i can be calculated from i = 25888734 – 25888696 = 38.

From the equation (1),

$$d_{ir} = \left\lfloor \frac{1 + (e - 1) * (n - 2\lceil\sqrt{n}\rceil - i + 1)}{e} \right\rfloor$$

$$d_{ir} = \left\lfloor \frac{1 + (3499 - 1) * (167556124362173 - 2\lceil\sqrt{167556124362173}\rceil - 38 + 1)}{3499} \right\rfloor$$
$$= 167508211620489$$

Usually, the tradition initial value to find the private key from right to left is begun as $d_r = n - 2 = 167556124362171$, that means the distance (s) to find d is decreased as:

$$s = \frac{d_r - d_{ir}}{2}$$
$$= \frac{167556124362171 - 167508211620489}{2}$$
$$= 23956370841$$

Therefore, the distance (t_r) between d_{ir} and d is as follows:

$$t_r = \frac{d_{ir} - d}{2}$$
$$t_r = \frac{167508211620489 - 134178957795523}{2}$$
$$= 16664626912483$$

Assuming that the method in [13] is selected to find d, d_{il} has to be computed from the following equation:

$$d_{il} = \left\lceil \frac{2n - 3}{3e} \right\rceil$$
$$= \left\lceil \frac{2 * 167556124362173 - 3}{3 * 3499} \right\rceil$$
$$= 31924573567$$

Therefore, the distance (t_l) between d_{il} and d is as follows:

$$t_l = \frac{d - d_{il}}{2}$$
$$= \frac{134178957795523 - 31924573567}{2}$$
$$= 67073516610978$$

Then total loops to find d by using the proposed method is less than the method in [13]. In fact, the reason is that k = 2802 (ed = 2802* $\Phi(n)$ + 1) in this example is quite large that is more suitable for the proposed method. On the other hand, it becomes unsuitable for the method in [13] which is highly efficient when k is a small integer.

In addition, the information in example 4 will show the case of e and d that the proposed method can be selected to find d very fast.

Example 4: Assuming n = 1907322773, e = 7603 and all pairs of $LSG_2(p + q)$ are disclosed as follows: S = {06, 14, 26, 34, 46, 54, 66, 74, 86, 94} (using theorems in [29]), Finding the least upper bound of d by using theorem 2. (The private parameters are p = 41981, q = 45433, $\Phi(n)$ = 1907235360 and d = 1906984507)

Sol.

Before, using the equation in theorem 2, the variable, i, must be found,

First, two forms of n must be checked,

Because (1907322773 + 1) mod 8 = 6, then the result of ($p + q$) mod 8 = 2, 6.

Furthermore, ($p + q$) mod 3 must be equal to 0, because 1907322773 mod 6 = 5.

Next, the initial value of $p + q$ which is equal to $g = 2\lceil \sqrt{n} \rceil$ must be calculated, $\lceil 2\sqrt{1907322773} \rceil$ = 87346.

Because 87346 mod 8 = 2, then g = 87346

Because 87346 mod 3 = 1, then g should be increased. However, g = 87354 is the minimum integer that is larger than 87346 and this value is still in the case of ($p + q$) mod 8 = 2 or 6 and ($p + q$) mod 3 = 0. Therefore, g = 87354.

Because $LSG_2(87354)$ = 54 is already a member in S, therefore, it is chosen as the new initial value. Furthermore, i can be calculated from i = 87354 – 87346 = 8.

From the equation in theorem,

$$d_{ir} = \left\lfloor \frac{1 + (e - 1) * (n - 2\lceil \sqrt{n} \rceil - i + 1)}{e} \right\rfloor$$

$$d_{ir} = \left\lfloor \frac{1 + (7603 - 1) * (1907322773 - 2\lceil \sqrt{1907322773} \rceil - 8 + 1)}{7603} \right\rfloor$$

= 1906984566

Because d is always an odd integer, then d_{ir} = 1906984566 – 1 = 1906984565

Therefore, the distance (t_r) between d_{ir} and d is as follows:

$$t_r = \frac{d_{ir} - d}{2}$$

$$t_r = \frac{1906984565 - 1906984507}{2}$$

= 29

Although, d which is in this example is very large, the distance between the new initial value and the target is only 29. Therefore, it consumes only a little time to recover d. In fact, the reason that the proposed method can recover d very fast is k = 7602 ($e*d = 1 + 7602*\Phi(n)$) that is the maximum value.

Moreover, if $2\lceil\sqrt{n}\rceil + 1 = p + q$ and $k = e - 1$, d_{ir} is certainly equal to d. Therefore, d can be recovered by using the equation (1).

In addition, after d_{ir} is found, d can be recovered by using algorithm 1.

Algorithm 1 Finding d' by using brute force attack from right to left with the new initial value

Input: d_{ir}, e, n

Output: d' (d' is possible to be equal to d)

```
1.  Selecting m, 1 < m < n
2.  c ← m^e mod n
3.  a ← c^-1 mod n
4.  a ← a^2 mod n
5.  t ← d_ir
6.  h ← c^t mod n
7.  i ← 0
8.  While h ≠ m do
9.      h ← h*a mod n
10.     i ← i + 1
11. End While
12. d' ← d_ir - 2*i
```

In fact, d' is very high possible to be equal to d. However, if d' is not still equal to d, then the loop in line 8 to 10 is required again with the present value of i to find the period value which will be occurred when h becomes to m again. In addition, various unrelated integers will be removed when the period value is found.

Nevertheless, if the position of d is at the middle of [d_{il}, d_{ir}], both of the proposed method and the method in [13] are not suitable to recover d, because computation costs are still high.

However, if this event is occurred, d_{mid} should be selected as the initial value instead of d_{il} and d_{ir}. This value can be estimated by using the equation (2).

$$d_{mid} = \frac{d_{ir} + d_{il}}{2} \tag{2}$$

Although, d_{mid} is closer to d than both of d_{ir} and d_{il} (when the position of d is at the middle of [d_{ir}, d_{il}]), it cannot be confirmed that d_{mid} is larger or less than d. Therefore, after d_{mid} is found, both of two ways for brute force attack, (left to right) and (right to left), must be selected to recover d.

Example 5: Assuming n = 1907322773, e = 1441 and all pairs of $LSG_2(p + q)$ are disclosed as follows: S = {06, 14, 26, 34, 46, 54, 66, 74, 86, 94} (using theorems in [29]), Finding the least upper bound of d by using theorem 2. (The private parameters are p = 41981, q = 45433, $\Phi(n)$ = 1907235360, k = 675 and d = 893396161)

Sol.

Because i = 8 is already calculated is example 4, d_{ir} can be estimated as follows:

$$d_{ir} = \left\lfloor \frac{1 + (e - 1) * (n - 2\lceil\sqrt{n}\rceil - i + 1)}{e} \right\rfloor$$

$$d_{ir} = \left\lfloor \frac{1 + (1441 - 1) * (1907322773 - 2\lceil\sqrt{1907322773}\rceil - 8 + 1)}{1441} \right\rfloor$$

= 1905911870, because d is always an odd number, therefore, d_{ir} = 1905911869

Then,
$$t_r = \frac{d_{ir} - d}{2}$$
$$t_r = \frac{1905911869 - 893396161}{2}$$
$$= 506257854$$

However, if d_{il} is selected, it can be estimated as follows:

$$d_{il} = \left\lceil \frac{2n - 3}{3e} \right\rceil$$
$$= \left\lceil \frac{2 * 1907322773 - 3}{3 * 1441} \right\rceil$$
= 882408

Because d is always an odd number, therefore, d_{il} = 882409

Then,
$$t_l = \frac{d - d_{il}}{2}$$
$$= \frac{893396161 - 882409}{2}$$
= 446256876

In fact, d_{mid} can be estimated by using equation (2)

$$d_{mid} = \frac{d_{ir} + d_{il}}{2}$$
$$= \frac{1905911869 + 882409}{2}$$
= 953397139

In this example, $d_{mid} > d$, then distance between d_{mid} and d is as follows:

$$t_{mid} = \frac{d_{mid} - d}{2}$$
$$= \frac{953397139 - 893396161}{2}$$
$$= 30000489$$

Where, t_{mid} is the distance between d_{mid} and d

However, for the real situation, if d_{mid} is selected, the process to find d has to execute two ways for brute force attack. Therefore,

$$t_{mid} = 2*30000489$$
$$= 60000978$$

Table 3. Comparison about total distance during three techniques from example 5

Algorithm	The initial value	Total distance
Brute Force Attack (left to right)	d_{il} = 882409	446256876
Brute Force Attack (right to left)	d_{ir} = 1905911869	506257854
Brute Force Attack (two ways)	d_{mid} = 953397139	60000978

The information in example 5 shows that if the position of d is close to the middle of [d_{il}, d_{ir}], d_{mid} which is the center between d_{ir} and d_{il} is closer to d than both of d_{ir} and d_{il}.

4. Experimental Results

In this section, the experimental results will be mentioned. It is divided into two parts. The first part is to analyze many pairs of e and d which are generated for the single value of n. This part is also divided into two experiments. One is for n which is generated from the same size of p and q. The other is the experiment of n which is generated from the different size of p and q. The second part is the comparison about loops computation to find d during the proposed method and the other algorithms. However, in this part, the weak parameters that respond well to the proposed method are chosen to strongly demonstrate that this method is highly efficient under these conditions.

Table 4. Considering d_{ir} from each pair of (e, d) generating from n = 1907322773

Row	k	(Public Key) e	e - k	(Private Key) d	d_{ir}	$(d_{ir} - d)/2$	Decreasing (%)
1	7602	7603	1	1906984507	1906984565	58	99.98
2	250852	250853	1	1907227757	1907227815	58	99.93
3	21	23	2	1741388807	1824312139	82923332	50.02
4	238	241	3	1883493841	1899321579	15827738	33.57
5	3620	3623	3	1905656087	1906708995	1052908	36.82
6	6550	6553	3	1906362217	1906944371	582154	39.39
7	3	7	4	817386583	1634773217	817386634	25
8	247	251	4	1876841171	1899636871	22795700	25.21
9	873	877	4	1898536453	1905060693	6524240	25.74
10	62	67	5	1764904363	1878769219	113864856	20.04
11	428	433	5	1885211857	1902830719	17618862	20.31
12	457	463	6	1882519567	1903116121	20596554	16.96
13	520	527	7	1881902063	1903616377	21714314	14.58
14	411580	411587	7	1907202923	1907230786	27863	76.75
15	1005540	1005547	7	1907222083	1907233523	11440	88.63
16	819	827	8	1888785683	1904929210	16143527	12.91
17	182	191	9	1817365631	1897249894	79884263	11.19
18	3	13	10	440131237	1760525003	1320393766	10
19	149529	149539	10	1907107819	1907222665	114846	46.57
20	1658023	1658033	10	1907223857	1907234269	10412	89.47
21	75	89	14	1607220809	1885805807	278584998	7.16
22	2957	2971	14	1898248051	1906593469	8345418	8.03
23	100967	100981	14	1906970941	1907216532	245591	30.19
24	59	77	18	1461388133	1882466127	421077994	5.57
25	521	539	18	1843542899	1903696949	60154050	5.68
26	3755	3773	18	1898136437	1906729923	8593486	6.45
27	229	269	40	1623631589	1900145325	276513736	2.53
28	2919	2959	40	1881453199	1906590865	25137666	2.82
29	1547	1597	50	1847522293	1906041157	58518864	2.14
30	30293	30343	50	1904092567	1907172563	3079996	4.65
31	3142743	3142793	50	1907205017	1907234813	29796	74.69
32	9	109	100	157478149	1889737847	1732259698	1
33	178987	179087	100	1906170383	1907224769	1054386	8.5
34	234	337	103	1324311793	1901575967	577264174	0.98
35	12366	12469	103	1891480669	1907082461	15601792	1.51
36	863	1013	150	1624821437	1905025413	280203976	0.12
37	10993	11143	150	1881561367	1907064259	25502892	1.01
38	25673843	25673993	150	1907224217	1907235345	11128	88.71
39	1137	1337	200	1621934633	1905808916	283874283	0.54
40	5954607	5954807	200	1907171303	1907235099	63796	42.11

The information in Table 4 that p and q have the same size (16 bits) shows that the distance to find d can be decreased about 99% whenever $k = e - 1$, because this value is selected for the proposed equation. Furthermore, for the same value of $e - k$ and this value may be large, the distance will be more reduced when e is larger. The clearly example is shown in 18th row, 19th row and 20th row that the result of $e - k = 10$. For the 18th row, that $e = 13$ is the smallest in this group, the distance is decreased only 10%. However, the distance is decreased 46.57% in 19th row that the ratio is larger than 18th row. The reason is that $e = 149539$ is very larger than $e = 19$. Furthermore, the distance can be reduced 89.47% in 20th row, because $e = 1658033$ is the largest value in comparison to the other values that are generated from the same result of $e - k = 10$. In fact, the reason that the distance can be more decreased when e is large and the result of $e - k$ is stable will be shown in theorem 3.

Theorem 3 Assuming $e_1*d_1 = 1 + k_1*\Phi(n)$, $e_2*d_2 = 1 + k_2*\Phi(n)$ where e_1 and e_2 are very large, $e_1 > e_2$ and $e_1 - k_1 = e_2 - k_2 = s$ then d_1 is always larger than d_2.

Proof: From,

$$d = \frac{1+k\Phi(n)}{e}$$

Then, $$d_1 = \frac{1+k_1\Phi(n)}{e_1} \text{ and } d_2 = \frac{1+k_2\Phi(n)}{e_2}$$

Because, $e_1 - k_1 = e_2 - k_2 = s$, then $k_1 = e_1 - s$ and $k_2 = e_2 - s$

Therefore,

$$d_1 = \frac{1+(e_1-s)\Phi(n)}{e_1} \text{ and } d_2 = \frac{1+(e_2-s)\Phi(n)}{e_2}$$

Because e_1 and e_2 are very large, then $\frac{1}{e_1} \approx \frac{1}{e_2} \approx 0$

That mean, $$d_1 \approx \frac{(e_1-s)\Phi(n)}{e_1} \text{ and } d_2 \approx \frac{(e_2-s)\Phi(n)}{e_2}$$

Because, $e_1 > e_2$, then $\frac{(e_1-s)}{e_1} > \frac{(e_2-s)}{e_2}$

Therefore, d_1 is always larger than d_2 when $e_1 - k_1 = e_2 - k_2 = s$

In addition, for the proposed equation, if e is very large, then $\frac{(e-1)}{e} \approx 0.999xxx$. That mean, the scope of all possible values of d_{ir} is quite narrow. In **Table 4**, d_{ir} is during 1900145325 to 1907235359 when $e > 269$. Therefore, the proposed method is suitable for the case that the distance between e and k is quite far and e is a large number.

However, the ratio of decreased distance becomes small when k is very close to e and e is a large integer. Therefore, if k is very close to e, the proposed method has very high performance when e is small.

Therefore, it implies that the proposed method is suitable to be selected to recover d when one of two following conditions occurs.

1. k is very close to e and e is a small number
2. e and k must be large when the distance between e and k is far.

Furthermore, the information in this table is also shown that all values of d_{ir} are less than $\Phi(n)$ that is demonstrated in **Fig. 2**.

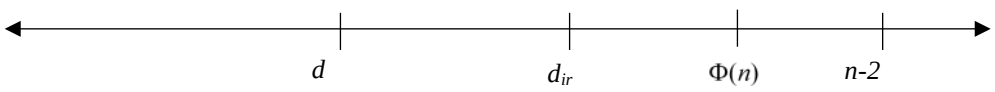

Fig. 2. The position of each parameter in Table 4 on number line

Table 5. Considering d_{ir} from each pair of (e, d) generating from n = 3187657073, p = 7193, q = 443161 (p = 13 bits, q = 19 bits)

Row	k	(Public Key) e	e - k	(Private Key) d	d_{ir}	$(d_{ir} - d)/2$	Decreasing (%)
1	1062	1063	1	3184208407	3184545517	337110	90.22
2	2998312	2998313	1	3187205657	3187543083	337426	25.25
3	41	43	2	3038964547	3113415213	74450666	49.92
4	7367	7369	2	3186341689	3187111585	769896	41.46
5	865029	865031	2	3187199351	3187540463	341112	25.47
6	4	7	3	1821260983	2732180697	910919714	33.33
7	10	13	3	2451697477	2942348443	490650966	33.33
8	88	91	3	3082133971	3152516189	70382218	33.30
9	137707	137713	6	3187067857	3187521001	453144	23.09
10	138857	138863	6	3187069007	3187521193	452186	23.10
11	960	967	7	3164134903	3184247825	20112922	14.49
12	23071810	23071817	7	3187205753	3187544009	338256	25.05
13	10679	10687	8	3184820863	3187245883	2425020	14.49
14	2385849	2385857	8	3187196033	3187542811	346778	24.78
15	2	11	9	579492131	2897767407	2318275276	11.11
16	9	19	10	1509729499	3019778665	1510049166	10
17	39	49	10	2536756369	3122492225	585735856	10.01
18	34234219	34234229	10	3187205789	3187544053	338264	25.04
19	159349	159361	12	3186966721	3187524145	557424	19.25
20	239987	239999	12	3187047359	3187530865	483506	20.69
21	9	23	14	1247167847	3048955271	1801787424	7.14
22	26	41	15	2021155481	3109799167	1088643686	6.67
23	4277	4297	20	3172372153	3186802341	14430188	5.59
24	47247	47267	20	3185858123	3187476711	1618588	10.02
25	1348577	1348597	20	3187159453	3187541783	382330	23.16
26	189	229	40	2630489389	3173624741	543135352	2.51
27	1561	1601	40	3107576321	3185553177	77976856	2.62
28	366589	366629	40	3186858989	3187535453	676464	15.23
29	113	163	50	2209535947	3167988661	958452714	2.01
30	1131	1181	50	3052269941	3184845125	132575184	2.07
31	827833	827783	50	3187014217	3187540297	526080	18.1
32	31	131	100	754224491	3163211749	2408987258	1
33	34353	34453	100	3177955837	3187451629	9495792	2.11
34	65447	65557	110	3181858813	3187495525	5636712	2.78
35	4785551	4785661	110	3187133461	3187543481	410020	21.69
36	89	289	200	981527329	3176514583	2194987254	0.5
37	123	323	200	1213704107	3177675589	1963971482	0.5
38	116088189	116088389	200	3187201229	3187544119	342890	24.77
39	1217	937	280	2453913473	3184142285	730228812	0.47
40	9281913	9282193	280	3187110577	3187543803	433226	20.72

In **Table 5**, although bits length of p and q are different, bits length of p and q are 13 and 19 in order, the proposed method is still high performance when characteristics of e and k are in one of two conditions above. However, d_{ir} may be farther from the target, because $(n-2\lceil\sqrt{n}\rceil-i+1?$ is farther from $\Phi(n)$ when it is compared with the other values of n which are generated from the same size of p and q and are also very close to n.

However, the information in this table is shown that some values of d_{ir} are larger than $\Phi(n)$. Therefore, number of loops to find d are still large when this event is occurred.

Therefore, from the information in **Table 4** and **Table 5**, it implies that d_{ir} can be estimated well when one of two conditions is happened and the difference between p and q is a little.

In fact, to confirm that the proposed method performs well when the weak points are encountered. It is compared to the other methods to recover d. Then, all results are from the small result of e - k and the difference between p and q is rather minor. Furthermore, the compared methods are brute force attack (searching from right to left), the improved FFA [22], the improved TDA [20] and Pollard's $p - 1$.

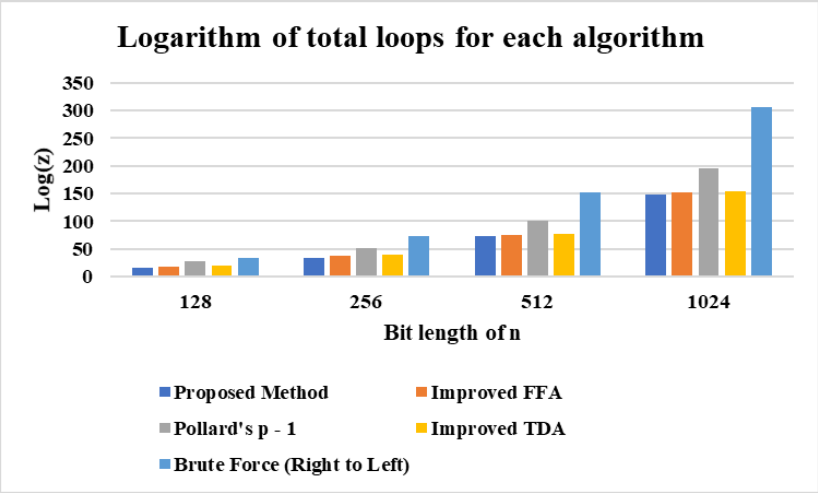

Fig. 3. Logarithm of total loops for each algorithm

In **Fig. 3**, assuming that z is represented as the distance to find d, the information of y-axis is the logarithm of z and x-axis is bits length of modulus. The experimental results show that the distance to recover d by using the proposed method is the smallest. However, the distance from the improved FFA is close to the proposed method, because FFA is also suitable for the small result of $p - q$.

5. Conclusion

This paper presents the new least upper bound on the private key which is always equal or less than the difference between the modulus and 2. In fact, this value is selected as the new initial value for brute force attack by searching for the private key from right to left. That mean, this technique is appropriate for the large private key. Unfortunately, when the difference between the public key and the multiplier of Euler totient function is very small, this method is extremely efficient. However, if the result is high, the public key and the multiplier should be large as well. In addition, this method performs well when p and q is close to each other. Moreover, assuming the position of d is close to scope's center, selecting the center between the new least upper bound and the greatest lower bound in [13] is preferable alternative. The experimental results demonstrate that if one of the weak points occurs, the proposed method can estimate the new initial value that is extremely close to the private key.

References

[1] K. Halunen and O.M. Latvala, "Review of the use of human senses and capabilities in cryptography," *Computer Science Review*, vol. 39, pp. 1 – 10, 2021. Article (CrossRef Link) .

[2] R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of ACM*, vol. 21, pp. 120 – 126, 1978. Article (CrossRef Link) .

[3] V. Guleria, S. Sabir and D.C. Mishra, "Security of multiple RGB images by RSA cryptosystem combined with FrDCT and Arnold transform," *Journal of Information Security and Applications*, vol. 54, pp. 1 – 13, 2020. Article (CrossRef Link) .

[4] L. Yang, T. Shanyu, L. Ran, Z. Liping and M. Zhao, "Secure and robust digital image watermarking scheme using logistic and RSA encryption," *Expert Systems with Applications*, vol. 97, pp. 95 - 105, 2018. Article (CrossRef Link) .

[5] K. Jiao, G. Ye, Y. Dong, X. Huang and J. He, "Image Encryption Scheme Based on a Generalized Arnold Map and RSA Algorithm," *Security and Communication Networks*, vol. 2020, pp. 1 - 14, 2020. Article (CrossRef Link) .

[6] C.L. Chen and C.C. Chen, "A Verifiable and Traceable Secondhand Digital Media Market Protocol," *KSII Transactions on Internet and Information Systems*, vol. 5, pp. 1472 - 1491, 2011. Article (CrossRef Link) .

[7] K. Somsuk and M. Thakong, "Authentication system for e-certificate by using RSA's digital signature," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, pp. 2948 - 2955, 2020. Article (CrossRef Link) .

[8] K. Sharma, A. Agrawal, D. Pandey, R.A. Khan and S. K. Dinkar, "RSA based encryption approach for preserving confidentiality of big data," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 2088-2097, 2022. Article (CrossRef Link) .

[9] M.E. Wu, C.M. Chen, Y.H. Lin and H.M. Sun, "On the Improvement of Wiener Attack on RSA with Small Private Exponent," *The Scientific World Journal*, vol. 2014, pp. 1 - 9, 2014. Article (CrossRef Link).

[10] K. Somsuk, "The New Equation for RSA's Decryption Process Appropriate with High Private Key Exponent," in *Proc. of International Computer Science and Engineering Conference*, pp. 1-5, November 15 – 18, 2017. Article (CrossRef Link) .

[11] S. Murat, "Generalized Trial Division," *International Journal of Contemporary Mathematical Science*, vol. 6(2), pp. 59 – 64, 2011.

[12] M.E. Wu, R. Tso and H.M. Sun, "On the improvement of Fermat factorization using a continued fraction technique," *Future Generation Computer Systems*, vol. 30(1), pp.162 – 168, 2014. Article (CrossRef Link).

[13] K. Somsuk, "A New Methodology to Find Private Key of RSA Based on Euler Totient Function," *Baghdad Science Journal*, vol. 18(2), pp.338– 348, 2021. Article (CrossRef Link).

[14] K.G. Chol, L.S. Chol and H.H. Cho, "Fast rebalanced RSA signature scheme with typical prime generation," *Theoretical Computer Science*, vol. 830 - 831, pp.1 - 19, 2020. Article (CrossRef Link).

[15] S.M. Sedjelmaci, "On a parallel extended Euclidean algorithm," in *Proc. of ACS/IEEE International Conference on Computer Systems and Applications*, pp. 235 - 241, June 25 – 29, 2001. Article (CrossRef Link).

[16] D. Chandravathi and P.V. Lakshmi, "Privacy Preserving Using Extended Euclidean Algorithm Applied To RSA-Homomorphic Encryption Technique," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, pp.3175 - 3179, 2019. Article (CrossRef Link).

[17] M. Wiener, "Cryptanalysis of short RSA secret exponents," *IEEE Transactions on Information Theory*, vol. 36, pp. 553-558, 1990. Article (CrossRef Link).

[18] D. Boneh, and G. Durfee, "Cryptanalysis of RSA with Private Key d less than $N^{0.292}$," in *Proc. of Advances in Cryptology — EUROCRYPT '99*, pp. 1 – 11, 1999. Article (CrossRef Link) .

[19] N. Lal, A. P. Singh and S. Kumar, "Modified trial division algorithm using KNJ-factorization method to factorize RSA public key encryption," in *Proc. of International Conference on Contemporary Computing and Informatics*, pp. 992-995, November 27 – 29, 2014. Article (CrossRef Link) .
[20] K. Somsuk, T. Chiawchanwattana and C. Sanemueang, "Estimating the new Initial Value of Trial Division Algorithm for Balanced Modulus to Decrease Computation Loops," in *Proc. of International Joint Conference on Computer Science and Software Engineering*, pp. 143-147, July 10 – 12, 2019. Article (CrossRef Link) .
[21] B.R. Ambedkar, A. Gupta, P. Gautam and S.S. Bedi, "An Efficient Method to Factorize the RSA Public Key Encryption," in *Proc. of International Conference on Communication Systems and Network Technologies*, pp. 108 - 111, June 3 – 5, 2011. Article (CrossRef Link) .
[22] K. Somsuk, "The improvement of initial value closer to the target for Fermat's factorization algorithm," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 21, pp. 1573 – 1580, 2018. Article (CrossRef Link) .
[23] J. McKee, "Speeding Fermat's factoring method," *Mathematics of Computation*, vol. 68, pp. 1729 – 1737, 1999. Article (CrossRef Link) .
[24] K. Omar, "Algorithm for factoring some RSA and Rabin moduli," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 11(5), pp. 537 - 543, 2008. Article (CrossRef Link) .
[25] Q. Huang, Y.T. Li, Y. Zhang and C. Lu, "A Modified Non-Sieving Quadratic Sieve For Factoring Simple Blur Integers," in *Proc. of International Conference on Multimedia and Ubiquitous Engineering*, pp. 729 - 732, April 729 – 732, 2007. Article (CrossRef Link) .
[26] H.M. Bahig, M.A. Mahdi, K.A. Alutaibi, A. AlGhadhban and H.M. Bahig, "Performance Analysis of Fermat Factorization Algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 11(12), pp. 340 - 352, 2020. Article (CrossRef Link) .
[27] J. M. Pollard, "Theorems of factorization and primality testing," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 76(3), pp. 521 - 528, 1974. Article (CrossRef Link) .
[28] K. Somsuk, "The new Weakness of RSA and The Algorithm to Solve this Problem," *KSII Transactions on Internet and Information Systems*, vol. 14(9), pp. 3841 - 3857, 2020. Article (CrossRef Link) .
[29] K. Somsuk and K. Tientanopajai, "An Improvement of Fermat's Factorization by Considering the Last m Digits of Modulus to Decrease Computation Time," *International Journal of Network Security*, vol. 19(1), pp. 99 - 111, 2017. Article (CrossRef Link) .
[30] K. Somsuk and K. Tientanopajai, "Improving fermat factorization algorithm by dividing modulus into three forms," *KKU Engineering Journal*, vol. 43, pp. 350 - 353, 2016. Article (CrossRef Link) .
[31] Y.B. Hammad, G. Carter and E. Dawson, "RAK factoring algorithm," *Australasian Journal of Combinatorics*, vol. 33(1), pp. 291 - 305, 2005.

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 integer factorization algorithms and quantum computing.