Journal of Interdisciplinary Mathematics
ISSN: 0972-0502 (Print), ISSN: 2169-012X (Online)
Vol. 27 (2024), No. 7, pp. 1461–1479
DOI : 10.47974/JIM-1689

The special method with the fake key to attack RSA

Kritsanapong Somsuk *
Department of Computer and Communication Engineering
Udon Thani Rajabhat University
Udon Thani 41000
Thailand

Chalida Sanemueang †
Office of Academic Resources and Information Technology
Udon Thani Rajabhat University
Udon Thani 4100
Thailand

Suchart Khummanee §
Department of Computer Science
Mahasarakham University
Mahasarakham 44150
Thailand

Chanwit Suwannapong ‡
Department of Computer Engineering
Nakhon Phanom University
Nakhon Phanom 48000
Thailand

Sarutte Atsawaraungsuk @
Department of Computer Education
Udon Thani Rajabhat University
Udon Thani 41000
Thailand

* *E-mail:* kritsanapong@udru.ac.th (Corresponding Author)
† *E-mail:* Chalida.sa@udru.ac.th
§ *E-mail:* suchart.k@msu.ac.th
‡ *E-mail:* schanwit@npu.ac.th
@ *E-mail:* sarutte@udru.ac.th

© Taru Publications

Abstract

Because RSA is the cryptographic algorithm that is still extensively employed today, several attack techniques against RSA are continually being developed. These algorithms are divided into two distinct groups. The first group is a general group unaffected by RSA parameters. The other group is unique due to at least one characteristic that represents a vulnerability. The aim of this research is to reveal the new vulnerabilities in RSA and to provide ways for recovering the original plaintext. The first vulnerability occurs when the sum of the private key and Euler totient function is near to the modulus but lower than the modulus. If this result is found, it can be selected as the exponent which is called the fake key. The second situation occurs whenever the key is larger than the modulus. The experimental results indicate that if a new weak point is found, the fake key may be disclosed quickly despite the modulus's size. In addition, when the sum of the public key and the Euler totient function is relatively near to the modulus, the proposed method can be also used to break the modulus. Additionally, the other vulnerability is demonstrated. It occurs when the product of the unknown integer j and the modulus is near to the addition of the key and the product of the other unknown integer and the Euler totient function. However, the issue with the second weak point is the mechanism for finding j.

Subject Classification: *94A60, 00A69, 68R01.*

Keywords: *RSA, Computation Time, Decryption process, The fake key, FK, FK2.*

1. Introduction

Cryptography [1] is the technique for protecting information communicated across an insecure channel by using encryption and decryption techniques, and it is categorized into two categories. The first category is known as symmetric key cryptography. The advantage is the procedure's celerity. However, developing a secure method for exchanging the secret key is challenging. The second method is referred as asymmetric key or public key cryptography [2]. This strategy can solve the problem of key exchange since it requires the selection of two distinct keys. The public key is shared with everyone in the group, whereas the private key is kept hidden by the owner. RSA [3] is the robust public key cryptography. In fact, the modulus's size must be at least 1024 bits [4] to avoid intrusions. However, there are two distinct groups of algorithms that can break RSA. The first group is a general group whose size relies only on the modulus such as Number Field Sieve (NFS) [5]. The second group is referred as a special group. In general, an algorithm's effectiveness is defined by its weakest point. Therefore, at least one parameter must be a weak point for one of the methods in the special group to fast break RSA. The examples of weak parameters and accompanying problem-solving strategies are

shown. Trial division algorithm (TDA) [6] is suited for addressing the first weakness, which is a prime factor that is too small. The small private key is also the vulnerable spot, since Wiener's attack [7] may easily break it. Pollard's $p-1$ can rapidly recover two prime factors when all prime factors of $p-1$ [8] are small, where p is one of prime factors. Fermat's Factorization [9] becomes an efficient procedure when the difference between two prime factors is small. The purpose of this study is to identify the new RSA's vulnerabilities and appropriate ways for breaking RSA. The key is to identify either the private key or an integer that is not the private key but may be used as an exponent in place of the private key. In fact, the fake key is the sum of the private key and the Euler totient function. In addition, the first proposed technique can identify the fake key quickly when it is close to the modulus. The second approach described for calculating the private key when the sum of the public key and Euler totient function is near to the modulus. Moreover, the other weak point is also presented. It occurs when the product of the integer and modulus is near to the sum of the private key (or public key) and the product of the other integer and the Euler totient function. However, the difficulty to attack RSA with this weak point is the way to find the multiplier of the modulus.

2. Related Works

In this part, RSA and other attacks will be discussed. The first section briefly introduces the notion of RSA. The second component is a study of many approaches to break RSA.

2.1 *RSA*

RSA is the best well – known public key cryptography. The first step in implementing RSA is to randomly generate two prime numbers, p and q. Then, the modulus, $n = p*q$ and Euler totient function, $\Phi(n) = (p-1)*(q-1)$ are calculated. The next step is to choose the public key, e, where $1 < e < \Phi(n)$ and $\gcd(e, \Phi(n)) = 1$. The last parameter is the private key, d, which is computed from $d = e^{-1} \bmod \Phi(n)$. After finishing the last step, the disclosed parameters are e and n and the secreted parameters are p, q, $\Phi(n)$ and d. Given m is the original plaintext, the encryption process is $c = m^e \bmod n$, where c is the ciphertext. Moreover, the recipient with the private key can retrieve m using the decryption equation: $m = c^d \bmod n$. In fact, the secreted parameters are the target of intruders. If one is found, the other values can be simply restored. However, it is extremely difficult to identify

one of them, when all parameters are strong, and n is allocated at least 1024 bits.

2.2 *Attacking Methods*

Several methods to break RSA will be explored in this section. However, if all parameters are resilient, these methods cannot destroy the system in polynomial time. In addition, they are separated into two categories: general group and special group.

2.2.1 General Group

The efficacy of these algorithms is completely determined by the modulus size. In other words, it implies that the small size of the modulus is the weak point that enables all techniques in this group to rapidly recover prime factors.

1) Quadratic Sieve (QS) [9] is one of the algorithms in this group. When all digits of the modulus in the decimal system are less than 100, QS is an effective algorithm. Assuming $m = \lfloor \sqrt{n} \rfloor$ and $f(a) = (a + m)^2 - n$, where $a \in \mathbb{Z}$. Then, $f(a) \equiv (a + m)^2 \bmod n$. The key is to find $f(a_1)*f(a_2)* \cdots *f(a_k)$, where $k \in \mathbb{Z}$, that the multiplication of their prime factors can be rewritten as $(p_1*p_2*p_3*\cdots *p_l)^2$, where $l \in \mathbb{Z}$. Therefore,

$$X^2 \equiv Y^2 \bmod n \tag{1}$$

Where $X^2 = f(a_1)*f(a_2)* \cdots *f(a_k)$ and $Y^2 = (p_1*p_2*p_3*\cdots *p_l)^2$

Then, p can be compute from: $p = \gcd((X - Y), n)$

2) Number Field Sieve (NFS) [10] is the most efficient factoring algorithm. Two types of this approach are identified, General Number Field Sieve (GNFS) and Special Number Field Sieve (SNFS). Moreover, the efficiency of NFS remains excellent even though that the modulus is greater than 10^{100}. However, NFS cannot discover both p and q in polynomial time when a modulus of at least 1024 bits is employed.
3) Square Forms Factorization (SQUFOF) [11] is one of general group factoring techniques. It was introduced by D. Shanks. In fact, SQUFOF is improved from Fermat Factorization Algorithm. In addition, SQUFOF may be represented in terms of continued fractions or quadratic forms. The process to find a prime factor is divided into two parts. The first step is to identify the integer that is

a perfect square. The second step is to identify the integer that equals the one from the previous step.

2.2.2 Special Group

The algorithms in the special group differ from those in the general group in that their performance depends on at least one parameter's weakness. Normally, there are several types of weak points, each of which is suited to one of these strategies.

1) Trial Division Algorithm (TDA) is a simple algorithm for identifying p and q by locating the divisor of n that the remainder of the division is 0. Normally, TDA is divided into two techniques. The starting divisor for the first technique [12] is 3 and it is increased when the residual does not equal 0. In fact, the first weak point is a small prime factor. The second method [13] is to assign the first divisor as $\lfloor\sqrt{n}\rfloor$ and it is decreased when the remainder does not equal to 0. Therefore, the second weak point is the prime factor which is close to $\lfloor\sqrt{n}\rfloor$.
2) The Fermat's Factorization Algorithm (FFA) is the factoring algorithm that Pierre de Fermat discovered in 1604. He found that the modulus can be written as the difference of two perfect square numbers.

$$n = x^2 - y^2 \tag{2}$$

Where, $x = \frac{p+q}{2}$ and $y = \frac{p-q}{2}$

Therefore, two prime factors are calculated whenever both of x and y are found.

In fact, FFA is an effective method when p and q are quite close. However, enhanced FFA algorithms that minimize computing time have been developed such as [14], [15], [16].

3) Elliptic Curve Factorization Method (ECM) [17] is the factoring algorithm. In fact, Elliptic Curve Cryptography (ECC) is the other public key cryptography [18], [19], [20]. However, in 1987 H.W. Lenstra modifies ECC to enhance the modulus factoring capability, ECM. The Elliptic Curve equation is being used to get the prime factor. However, ECM can rapidly factor the modulus when its decimal digits are not extremely large, or the smallest prime factor is small. In addition, point addition and point doubling are the

fundamental operations. Nevertheless, both use modular Inverse, which pays the most computing expense. In 2021, the Fast-Elliptic Curve Factorization Method (F-ECM) [21] was presented to reduce the quantity of modular Inverses. It implies that F-ECM is faster than ECM while requiring additional modular multiplications.

4) Pollard' s $p - 1$ [22] is the method proposed by J. Pollard in 1974. When all prime factors of $p - 1$ or $q - 1$ are small, this approach can recover both p and q very quickly. Assuming $k! = (p - 1)*q$ and from Fermat's theorem that $a^{p-1} \bmod p = 1$, then it implies that $a^{k!} \bmod p = 1$. Hence, $p \mid (a^{k!} - 1)$ or $\gcd(a^{k!} - 1, n) = p$. Consequently, the primary function of Pollard's $p - 1$ is to seek for k. Moreover, the variant of Pollard's $p - 1$ was proposed in 2022 [23]. In fact, the enhanced procedure is more effective when all prime factor of $p - 1$ or $q - 1$ are the member of B-smooth.

5) VFactor [24], [25] is one of integer factorization algorithms. The key is to search for two integers, x and y, that $n = x*y$. Moreover, both must be allocated an odd number. Given that $i = \lfloor \sqrt{n} \rfloor$ is already an odd number, the initial values of x and y are i and i+2 respectively. Assign $t = x*y$; if t is greater than n, y must be decreased by two; and recalculate t until the target is reached. Nonetheless, x must be multiplied by two to determine the new value of t. In fact, VFactor is appropriate if p is close to q.

6) Wiener's attack [7] is the attack technique that Michael J. Wiener suggested in 1990. This strategy focuses on the private key. The fundamental mathematical paradigm is continue fraction. Nonetheless, this approach can rapidly retrieve the private key when $d < \frac{1}{3} n^{\frac{1}{4}}$. Moreover, D. Boneh and G. Durfee [26] improved the effectiveness of Wiener's attack in 1999. The experimental findings demonstrated that when $d < n^{0.292}$, the improved method is still effective.

7) The method to establish the new initial value [27] before searching for the private key by using brute force attack (search from left to right) was proposed in 2021. In general, the initial value for modular exponentiation, which is the exponent, is 3. Nonetheless, if this method is utilized, it may be approximated as $d_{il} = \left\lceil \frac{2n-3}{3e} \right\rceil$, where $d_{il} \leq d$. Consequently, the small value of d is the weak spot that reacts well

to this approach. In 2022, the extension of this technology [28] was also given. This technique will look for d_{ir} rather than d_{il} where $d_{ir} = \left\lfloor \frac{1+(e-1)*(n-2\sqrt{n}-i+1)}{e} \right\rfloor$, $i \in \mathbb{Z}^+$ and $d_{ir} > d$. The brute force attack must search from right to left. The limitation of the method described in [28] is that the last m digits of p and q must be disclosed prior to its use.

8) The method for recovering the original plaintext by modifying the decryption equation was proposed in [29]. Assuming, $\Phi(n) = ad + b$, where $a \mid b$ and $a \mid \Phi(n)$, the original plaintext can be also recovered by using the following equation: $m = (c^{-1})^{\frac{b}{a}} \bmod n$. In fact, this method can find m rapidly when the result of $\frac{b}{a}$ is small.

3. The Proposed Method

This paper presents the specialized techniques for retrieving the plaintext. The first algorithm is referred as the FK, or fake key. In fact, it is capable to recover the plaintext without revealing d, $\Phi(n)$, p, or q. In addition, the special case in which FK may easily recover the plaintext is that the addition result between d and $\Phi(n)$ is extremely close to n. Accordingly, when FK is revealed, this case becomes a risk for RSA.

Assuming, $$a = d + \Phi(n)$$

Then, $$c^a \bmod n = c^{d+\Phi(n)} \bmod n$$

$$= (c^d \bmod n) * (c^{\Phi(n)} \bmod n) \bmod n$$

From, Euler's theorem that $b^{\Phi(n)} \bmod n = 1$, where gcd $(b, n) = 1$
Therefore, if $c^{\Phi(n)} \bmod n = 1$, gcd$(c, n) = 1$, then $c^a \bmod n = c^d \bmod n$
Or,

$$m = c^a \bmod n \tag{3}$$

In fact, m may be reconstructed using equation (3) when a is revealed. Assuming $n = d + \Phi(n)$, then $c^n \bmod n = m$. Normally, there is a small possibility that $d + \Phi(n) = n$. However, it is probable that the sum of d and $\Phi(n)$ will be close to n. Nonetheless, the outcome is split as follows:

The first way: $$d + \Phi(n) < n$$

The second way: $$d + \Phi(n) > n$$

Then, two solutions exist for this issue. When the result of the addition between d and $\Phi(n)$ is in the first way, the first method is presented: a brute force attack from right to left using n as the initial exponent.

Assuming, $f = n$, the equation $x = c^f \bmod n$ is computed. If x equals m, there is a probability that $f = d + \Phi(n)$. Alternatively, f must be reduced by two until the target is reached. In truth, f may be reduced by two because the sum of d and $\Phi(n)$ is always an even number.

Assuming $x = c^f \bmod n \neq m$, the next equation for searching for m is $c^{f-2} \bmod n$. Thus, modular exponentiation must be calculated whenever the target cannot be found. To reduce the cost of calculation, modular multiplication is used to determine the result rather than modular exponentiation.

$$\text{Assigning } r = c^{-1} \bmod n, \text{ then}$$

$$c^f * r^2 \bmod n = c^f * (c^{-1})^2 \bmod n$$

$$= c^{f-2} \bmod n$$

Therefore, $c^{f-2} \bmod n$ can be computed by using the following equation:

$$c^{f-2} \, mod \, n = c^f * r^2 \bmod n \tag{4}$$

The second strategy is chosen because the result of $d + \Phi(n)$ is greater than n. In fact, n is still picked as the first exponent for this strategy. Moreover, since the target exponent is greater than n, the implementation must perform a brute force attack searching from left to right.

Assuming, $s = n$, the equation $y = c^s \bmod n$ is computed. If y equals to m, it is possible that $s = d + \Phi(n)$. Otherwise, s must be raised by two until the target is discovered. The same reason with the first technique, modular multiplication can be implemented instead of computing modular exponentiation directly to reduce costs by using the following equation:

$$c^{s+2} \bmod n = c^s * c^2 \bmod n \tag{5}$$

Nevertheless, FK is the outcome of the combining of two ways, as seen by algorithm 1. However, it is possible that the result may not be the target. If this condition is satisfied, the method must proceed to the next outcome. After obtaining this result, two values that may be the fake key are revealed, and the period value that is selected to rapidly find the fake key may be computed.

Example 1: Assuming $n = 572753$ and $e = 466807$ ($p = 953$, $q = 601$, $\Phi(n) = 571200$ and $d = 1543$), Finding the fake key to recover the plaintext by using FK

Solution: The steps of FK are as follows:

Steps 1 - 9: $m = 15$, $c = 550399$, $r = 357708$, $r_2 = 274805$, $c_2 = 260700$, $f = s = 572753$ and $x = y = 390003$

Step 10: Because, $390003 \neq 15$, then jump to steps 13 – 24

Steps 13 – 18: Process in Loop until x or y which equals to 15 is found

(Loop 1) Steps 14 – 17: $x = 87549$, $y = 387799$, $f = 572751$ and $s = 572755$

(Loop 2) Steps 14 – 17: $x = 413180$, $y = 276258$, $f = 572749$ and $s = 572757$

(Loop 3) Steps 14 – 17: $x = 229674$, $y = 207368$, $f = 572747$ and $s = 572759$

(Loop 4) Steps 14 – 17: $x = 473982$, $y = 400189$, $f = 572745$ and $s = 572761$

(Loop 5) Steps 14 – 17: $x = 15$, $y = 22338$, $f = 572743$ and $s = 572763$

Because $x = 15$, then continue to step 19

Steps 19-22: Because $x = 15 = m$, then $f = 572743$

From this example, just five rounds are required to determine the fake key, which may also be chosen to retrieve the plaintext. The reason is that the result of the addition between d and $\Phi(n)$ is very close to n, $|n-(d+\Phi(n))| = 10$. Additionally, when f is revealed, this number may be chosen as the other exponent to retrieve the plaintext without employing FK again. The example is as follows: assuming, an attacker who has already known f can entrap $c = 439052$ on the network channel. The plaintext can be recovered by using the following equation: $439052^{572743} \bmod 572753 = 21$.

Algorithm 1:	FK
Input:	n, e
Output:	The result which may be equal to $d + \Phi(n)$
1.	Assigning m as the plaintext for testing
2.	$c \leftarrow m^e \bmod n$
3.	$r \leftarrow c^{-1} \bmod n$
4.	$r_2 \leftarrow r^2 \bmod n$
5.	$c_2 \leftarrow c^2 \bmod n$
6.	$f \leftarrow n$
7.	$s \leftarrow f$
8.	$x \leftarrow c^f \bmod n$
9.	$y \leftarrow x$
10.	IF $x = m$ then

Contd...

```
11.         x equals to d + Φ (n)
12.     Else
13.         While x ≠ m and y ≠ m Do
14.             x ← x*r2 mod n
15.             y ← y*c2 mod n
16.             f ← f – 2
17.             s ← s + 2
18.         End While
19.         IF x = m then
20.             f equals to d + Φ (n)
21.         Else
22.             s equals to d + Φ (n)
23.         EndIF
24.     EndIF
```

Example 2: Assuming $n = 605911$ and $e = 283253$ ($p = 1511$, $q = 401$, $\Phi(n) = 604000$ and $d = 1917$), Finding the fake key to recover the plaintext by using FK

Solution: The steps of FK are as follows:

Steps 1-9: $m = 35$, c = 34324, $r = 282955$, $r_2 = 270218$, $c_2 = 245992$, $f = s = 605911$ and $x = y = 118263$

Step 10: Because, $118263 \neq 35$, then jump to steps 13 - 24

Step 13 – 18: Process in Loop until x or y which equals to 35 is found

(Loop 1) Steps 14 – 17: $x = 439283$, $y = 147053$, $f = 605909$ and $s = 605913$

(Loop 2) Steps 14 – 17: $x = 573328$, $y = 368965$, $f = 605907$ and $s = 605915$

(Loop 3) Steps 14 – 17: $x = 585558$, $y = \underline{35}$, $f = 605905$ and $s = 605917$

Because $y = 35$, then continue to step 19

Steps 19-22: Because $x = 585558 \neq 35$ ($y = 35 = m$), then $s = 605917$

In Example 2, the search for the fake private key requires just three iterations, $s = 605917$. In other words, even though the actual private key is 1917, RSA whose parameters are generated from this example are easy to break. When either s or f is known, the proposed method for decrypting equations using the fake key as the exponent instead of d can recover m. Furthermore, FK is an efficient method when either s or f is very close to n. On the other hand, FK might become ineffective. In fact, FK is the next

specific method for breaking RSA. Moreover, in FK, r_2 and c_2 are the constant variables. As they serve as multiplicators in modular multiplication, time can be further reduced when they are both small integers. Moreover, if the addition between d and $\Phi(n)$ is not close to n, but the addition between e and the $\Phi(n)$ is close to n, the second proposed method, FK2, is more efficient. Furthermore, d will be retrieved rather than the fake key.

Assuming, $$h = e + \Phi(n)$$

Then, $$m^h \bmod n = m^{e+\Phi(n)} \bmod n$$

$$= (m^e \bmod n) * (m^{\Phi(n)} \bmod n) \bmod n$$

Because, $$m^{\Phi(n)} \bmod n = 1$$

Therefore,

$$m^e \equiv m^h \bmod n \quad (6)$$

However, after h is found, d can be recovered as follows: From, $h = e + \Phi(n)$, then $\Phi(n) = h - e$

Because, both of $\Phi(n)$ and e are disclosed, d can be computed. Therefore, FK2 which is suitable for the small distance between $e + \Phi(n)$ and n is designed as follows:

Algorithm 2: FK2

Input: n, e

Output: The result which may be equal to $e + \Phi(n)$

---

1. Assigning m as the plaintext for testing
2. $c \leftarrow m^e \bmod n$
3. $r \leftarrow m^{-1} \bmod n$
4. $r_2 \leftarrow r^2 \bmod n$
5. $m_2 \leftarrow m^2 \bmod n$
6. $f \leftarrow n$
7. $s \leftarrow f$
8. $x \leftarrow m^f \bmod n$
9. $y \leftarrow x$
10. IF $x = c$ then
11. x equals to $e + \Phi(n)$
12. Else

Contd...

```
13.       While x ≠ c and y ≠ c Do
14.           x ← x*r2 mod n
15.           y ← y*m2 mod n
16.           f ← f – 2
17.           s ← s + 2
18.       End While
19.       IF x = c then
20.           Φ (n) ← f – e
21.       Else
22.           Φ (n) ← s - e
23.       EndIF
24.       d ← e^-1 mod Φ (n)
25.   EndIF
```

For both of FK and FK2, assuming $u = e + \Phi(n)$ and $v = d + \Phi(n)$, if $|n-u|$ is very close to $|n-v|$, then the performance of FK is also close to FK2. However, the aim of FK is the fake key. On the other hand, FK2 is aimed to get d. However, the following equation may also be utilized to retrieve m.

Assuming, $b = j^*n = d + i^*\Phi(n)$, where $j, i \in \mathbb{Z}$

Then, $c^b \equiv c^{j^*n} \equiv c^{d+i^*\Phi(n)} \bmod n$

Or, $c^b \bmod n = c^{j^*n} \bmod n = c^{d+i^*\Phi(n)} \bmod n = (c^d * c^{i^*\Phi(n)}) \bmod n$

Because, $c^{\Phi(n)} \bmod n = 1$, $c^{i^*\Phi(n)} \bmod n = (c^{\Phi(n)})^i \bmod n = 1$, Then

$$m = c^{j^*n} \bmod n \tag{7}$$

From equation (7), it implies that m is recovered rapidly when j^*n is very closed to $d + i^*\Phi(n)$. The example 3 is shown the case in this condition.

Example 3: Assuming $n = 572753$ and $e = 289543$ ($p = 953$, $q = 601$, $\Phi(n) = 571200$ and $d = 150007$)

Solution: In fact, the proposed method becomes inefficient with these parameters, because $d + \Phi(n) = 150007 + 571200 = 721207$ which is very larger than $n = 572753$. However, $1200^*n = 1200 * 572753 = 687303600$ and $d + 1203^*\Phi(n) = 150007 + 1203^*571200 = 687303607$, where $j = 1200$, $i = 1203$

Because $|687303600 - 687303607| = 7$, then 1200^*n is very close to $d + 1203^*\Phi(n)$

In fact, $d + 1203^* \Phi(n) = 687303607$ is the fake key to decrypt the encrypted message. In addition, to find the fake key, assume $j = 1200$ is disclosed, then FK can be modified a little to recover m as follows:

Steps 1 - 9: $m = 117$, $c = 224158$, $r = 376266$, $r_2 = 152451$, $c_2 = 333780$, $f = s = 687303601$ and $x = y = 481987$

Step 10: Because, $481987 \neq 117$, then jump to steps 13 - 24

Step 13 – 18: Process in Loop until x or y which equals to 117 is found

(Loop 1) Steps 14 – 17: $x =$ 345014**,** $y =$ 467208, $f = 687303599$ and $s = 687303603$

(Loop 2) Steps 14 – 17: $x =$ 103065**,** $y =$ 81424, $f = 687303597$ and $s = 687303605$

(Loop 3) Steps 14 – 17: $x =$ 29266**,** $y =$ **117**, $f = 687303595$ and $s = 687303607$

Because $y = 117$, then continue to step 19

Steps 19-22: Because $x = 29266 \neq 117$ ($y = 117 = m$), then $s = 687303607$

Given $j = 1200$ is disclosed, finding the fake key requires just three iterations. It can be used as the exponent in the decryption equation to recover the plaintext. Assuming sender still sends the following message, $m = 29$, to the receiver, and both parties are unaware that the attacker has already produced a fake key, then the following attack is possible: before sending the message, it must be encrypted as follows: $c = 29^{289543} \bmod 572753 = 8603$. Given that an attacker can capture c via the network channel, he or she can retrieve $m = 8603^{687303607} \bmod 572753 = 29$ rapidly. Nevertheless, the difficulty to solve in Example 3 to identify the fake key concerns the method for finding j. Moreover, assuming $= j^*n$ is close to $e + i^* \Phi(n)$ and j is disclosed, the private key can be recovered instead of the fake key by using the improvement of FK2.

4. Experimental Results

In this section, the experimental results will be presented in two parts. The first section discusses the average time required to find the target. In addition, n is assigned as 128, 256, 512, 1024 and 2048 bits. Furthermore, the distance between n and $d + \Phi(n)$ which is selected for the experiment is about 10^4, 10^5, 10^6 and 10^7. In fact, the results are shown in Table 1. Moreover, assume j is disclosed, the distance between j^*n and $d + i^* \Phi(n)$ is also considered in Table 2. The second section compares the total loops required to complete the procedure. In this part, the proposed

method is contrasted with different attack methods. Nevertheless, only a tiny gap between n and $d + \Phi(n)$ is chosen to claim that the proposed method is the most efficient way under these conditions. According to the same performance of FK and FK2 when $|n-(d+\Phi(n))|$ is close to $|n-(e+\Phi(n))|$, only FK is chosen for this section. In addition, BigInteger class in Java which can be represented as the unlimited size variable is selected for the implementation. All experiments were conducted on 2.53 GHz an Intel® Core i5 with 8 GB memory to control the same settings.

Table 1

Average time to find the fake key when the distance between n and $d + \Phi(n)$ is a little

Bits Length of n	Computation Time (mSec.) When $\lvert n-(d+\Phi(n))\rvert \approx x$			
	$x = 10^4$	$x = 10^5$	$x = 10^6$	$x = 10^7$
128	26.18	1.47×10^2	1.29×10^3	1.26×10^4
256	41.23	2.82×10^2	2.61×10^3	2.59×10^4
512	79.74	6.52×10^2	6.28×10^3	6.15×10^4
1024	2.05×10^2	1.95×10^3	1.91×10^4	1.90×10^5
2048	7.43×10^2	7.13×10^3	7.14×10^4	6.78×10^5

Table 2

Average time to find the fake key when the distance between $j*n$ and $d + i*\Phi(n)$ is a little

Bits Length of n	Computation Time (mSec.) When $\lvert j*n-(d+i*\Phi(n))\rvert \approx x$			
	$x = 10^4$	$x = 10^5$	$x = 10^6$	$x = 10^7$
128	27.26	1.49×10^2	1.31×10^3	1.37×10^4
256	42.24	2.91×10^2	2.72×10^3	2.71×10^4
512	80.11	6.61×10^2	6.28×10^3	6.32×10^4
1024	2.06×10^2	2.01×10^3	2.13×10^4	2.01×10^5
2048	7.44×10^2	7.32×10^3	7.24×10^4	6.92×10^5

Table 1 demonstrates that, for a given distance, the average time is shorter as the modulus's bit length increases, because the computing cost in each iteration demands larger parameters, particularly n, which is represented by the divisor. However, the fake key can be retrieved rapidly

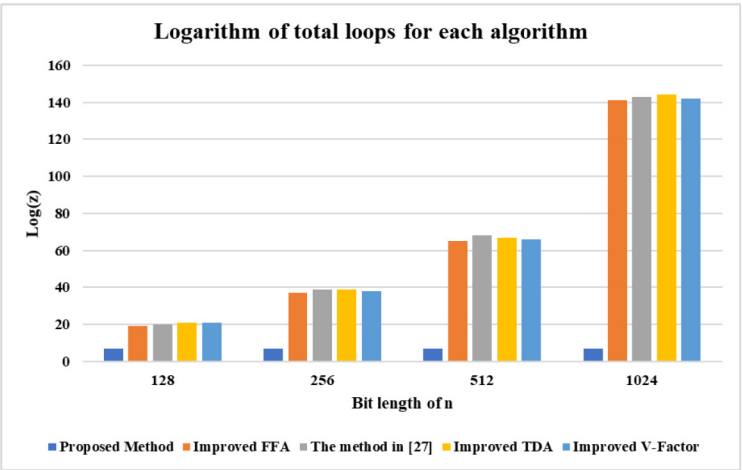


Figure 1

Logarithm of total loops for each algorithm

when n is close to $d + \Phi(n)$. In fact, the average time to recover the key is only 6.92 x 10^5 milli seconds or 11 minutes and 32 seconds when $|n-(d+\Phi(n))| \approx 10^7$, although bits length of n is 2048. Then, it implies that the $d + \Phi(n)$ and $e + \Phi(n)$ must be assigned far from n to avoid attacking easily by using the proposed method.

The results presented in Table 2 indicate that the average time is equivalent to the information in Table 1. However, the average time in Table 2 is higher than in Table 1 due to step 8 in both FK and FK2 where the exponent, $j*n$, is extremely large for modular exponentiation.

Figure 1 compares the total loops required to complete the procedure. The comparators are the improvement of FFA [14], the improvement of TDA [12], the method in [27], the improvement of V-Factor [25]. However, some methods which are not suitable to the modulus in this experiment are ignored such as Pollard's p-1 which is suitable for all small prime factors of p-1 or q -1 and the method in [29] which is suitable for the small result of $\frac{b}{a}$. The purpose of this experiment is to demonstrate that the proposed method is the optimal special algorithm when the distance between n and $d + \Phi(n)$ is low. Therefore, the distance of around 10^7 is chosen for this experiment. The experimental results demonstrated that the proposed method is the fastest algorithm when this criterion is satisfied. Moreover, when the distance between n and $d + \Phi(n)$ is small, the approach in [27] becomes an inefficient algorithm since e becomes a large value and the result of d_i is a little integer.

The example is that assigning n = 1836859944530168790203573448066 71713519 (128 bits), p = 17867764096422414989, q = 10280301075264113771 and $\Phi(n)$ = 18368599445301687899220927963498518476O, then assuming d = 28148065171685528751, then $d + \Phi(n)$ = 18368599445301687902035734 44806670713511 and the distance between n and $d + \Phi(n)$ is 1000008 (about 10^7). However, for this private key, it affects to e = 471821676818622 4187402609635553408065. That means d_i = 25 is a small integer and total loops are about 2.81 x 10^{19}.

5. Conclusion

This paper proposes a modified method for recovering the plaintext. In four cases, the proposed method can rapidly recover the plaintext.

Case 1: The small distance from the difference between the modulus and the addition between the private key and Euler totient function. The fake key is the target.

Case 2: The small distance from the difference between the modulus and the addition between the public key and Euler totient function. The private key is the target.

Case 3: The small distance from the difference between the result from the modulus multiplies with the integer and the result from the addition between the private key and the multiplication result between Euler totient function and the other integer. In this case, the multiplier of the modulus is assumed that it is disclosed. The fake key is the target.

Case 4: The small distance from the difference between the result from the modulus multiplies with the integer and the result from the addition between the public key and the multiplication result between Euler totient function and the other integer. In this case, the multiplier of the modulus is assumed that it is disclosed. The private key is the target.

Normally, if one of these situations occurs, the proposed method can quickly retrieve the original plaintext. Although 2048 bits of modulus are assigned, the findings of the experiments indicate that either the fake key or the private key may be found rapidly when the distance is small.

References

[1] V. K. Vivek and R. G. Rati, "Hybrid symmetric cryptography approach for secure communication in web application," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 24, no. 5, pp. 1179-1187 (2021).

[2] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654 (1976).

[3] 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).

[4] G. C. Kim, S. C. Li, and H. C. Hwang, "Fast rebalanced RSA signature scheme with typical prime generation," *Theoretical Computer Science*, vol. 830-831, pp. 1-19 (2020).

[5] S. M. Hamdi, S. T. Zuhori, F. Mahmud, and B. Pal, "A compare between Shor's quantum factoring algorithm and General Number Field Sieve," in Proceedings of the International Conference on Electrical Engineering and Information & Communication Technology, Dhaka, pp. 1-6, Apr. 10–12 (2014).

[6] S. Murat, "Generalized trial division," *International Journal of Contemporary Mathematical Science*, vol. 6, no. 2, pp. 59–64 (2011).

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

[8] J. M. Pollard, "Theorems of factorization and primality testing," Mathematical Proceedings of the Cambridge Philosophical Society, vol. 76, no. 3, pp. 521-528 (1974).

[9] B. R. Ambedkar, A. Gupta, P. Gautam, and S. S. Bedi, "An efficient method to factorize the RSA public key encryption," in Proceedings of the International Conference on Communication Systems and Network Technologies, Katra, pp. 108-111, Jun. 3–5 (2011).

[10] M. Cosnard and J. Philippe, "The quadratic sieve factoring algorithm on distributed memory multiprocessors," in Proceedings of the Fifth Distributed Memory Computing Conference, Charleston, pp. 254-262, Apr. 8–12 (1990).

[11] F. Lemmermeyer, "Václav Šimerka: quadratic forms and factorization," *LMS Journal of Computation and Mathematics*, vol. 16, pp. 118-129 (2013).

[12] N. Lal, A. P. Singh, and S. Kumar, "Modified trial division algorithm using KNJ-factorization method to factorize RSA public key encryption," in Proceedings of the International Conference on Contemporary Computing and Informatics, Mysore, pp. 992-995, Nov. 27–29 (2014).

[13] K. Somsuk, T. Chiawchanwattana, and C. Sanemueang, "Estimating the new initial value of trial division algorithm for balanced modulus to decrease computation loops," in Proceedings of the International Joint Conference on Computer Science and Software Engineering, Chonburi, pp. 143-147, Jul. 10–12 (2019).

[14] 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, no. 1, pp. 162–168 (2014).

[15] 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, no. 7-8, pp. 1573–1580 (2018).

[16] K. Omar and L. Szalay, "Sufficient conditions for factoring a class of large integers," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 13, pp. 95-103 (2010).

[17] H. W. Lenstra, "Factoring integers with elliptic curves," *Annals of Mathematics*, vol. 126, no. 3, pp. 649–673 (1987).

[18] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 103–109 (1987).

[19] V. S. Miller, "Uses of elliptic curves in cryptography," *Lecture Notes in Computer Science*, vol. 218, pp. 417–428 (1986).

[20] S. D. Sumit, S. Brahmjit, and J. Poonam, "Demystifying elliptic curve cryptography: Curve selection, implementation and countermeasures to attacks," *Journal of Interdisciplinary Mathematics*, vol. 23, no. 2, pp. 463–470 (2020).

[21] K. Somsuk, "The improvement of elliptic curve factorization method to recover RSA's prime factors," *Symmetry*, vol. 13, pp. 1-15 (2021).

[22] J. M. Pollard, "Theorems of factorization and primality testing," Mathematical Proceedings of the Cambridge Philosophical Society, vol. 76, no. 3, pp. 521-528 (1974).

[23] K. Somsuk, "An efficient variant of Pollard's p – 1 for the case that all prime factors of the p – 1 in B-smooth," *Symmetry*, vol. 14, pp. 1-17 (2022).

[24] P. Sharma, A. K. Gupta, and A. Vijay, "Notice of violation of IEEE publication principles: Modified integer factorization algorithm using V-factor method," in Proceedings of the International Conference on Advanced Computing & Communication Technologies, Rohtak, Haryana, pp. 423-425, Jan. 7–8 (2012).

[25] K. Somsuk, "Improving the initial values of V-factor suitable for balanced modulus," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 6, pp. 6446-6452 (2020).

[26] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than N^0.292," *Lecture Notes in Computer Science*, vol. 1592, pp. 1-11 (1999).

[27] K. Somsuk, "A new methodology to find private key of RSA based on Euler totient function," *Baghdad Science Journal*, vol. 18, no. 2, pp. 338-348 (2021).

[28] K. Somsuk, "The improved estimation of the least upper bound to search for RSA's private key," *KSII Transactions on Internet and Information Systems*, vol. 16, no. 6, pp. 2074-2093 (2022).

[29] K. Somsuk, "The new weakness of RSA and the algorithm to solve this problem," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 9, pp. 3841-3857 (2020).

Received October, 2022