Enhanced Algorithm for Recovering RSA Plaintext when Two Modulus Values Share at least One Common Prime Factor

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

Received 15 July 2024; Accepted 27 April 2025

Abstract

RSA is a significant cryptographic technique depending on public key cryptography. In fact, if this technique does not have any vulnerabilities, it is considered as a robust algorithm. Nevertheless, if the system contains vulnerabilities, it can be compromised through the utilization of algorithms that target those weaknesses. Currently, RSA is vulnerable to multiple potential weaknesses. This research identifies a novel vulnerability in RSA and presents a technique to exploit this vulnerability. The proposed method, known as Common Prime Attack (CPA), aims to exploit this vulnerability to recover the original message. The vulnerability arises when at least two key producers generate their modulus that inadvertently have at least one prime factor in common. It is divided into three cases. Case 1 relates to two moduli created from two prime numbers, with one prime being common. Case 2 involves the two moduli generated from at least three primes, with one prime common to both values. Case 3 concerns two moduli produced from at least three primes, with a minimum of two primes in common. The evidence and examples show that all situations can effectively retrieve the original message. The datasets have been generated to support three cases in which

Journal of Cyber Security and Mobility, Vol. 14_2, 433–456.
doi: 10.13052/jcsm2245-1439.1427

each modulus shares a common prime factor. Therefore, the experimental results demonstrate that all situations can rapidly compromise RSA with modulus lengths of 1024, 2048 and 4096 bits. CPA can successfully recover plaintexts in all cases, with a 100% success rate under simulated conditions. Moreover, the average computation time is extremely low, ranging from 1.6 ms to 17.2 ms. Because of the large bit length of the modulus, other methods are not suitable for comparison with the proposed method in terms of computational time. In addition, they require a significant amount of time to execute. However, CPA will instead be compared to other algorithms with respect to security aspects. In conclusion, all generated moduli must be mutually coprime to avoid vulnerabilities associated with CPA.

Keywords: RSA, common modulus attack, common prime attack, modulus.

1 Introduction

Cryptography [1] is a field associated with protecting the confidentiality and integrity of information transmitted over the internet through the utilization of encryption and decryption techniques. The fundamental process for data security is encrypting the original plaintext to be a ciphertext, an unreadable message, generated by the sender. Then, the ciphertext is transmitted to the recipient, who will decrypt this value to recover the original plaintext. According to a case where an unauthorized individual can intercept the transmitted data, they will be unable to recover the original plaintext unless they have the secret key, or the private key. In fact, both are maintained under strict confidentiality. Currently, in the scope of digital computer processing, cryptography can be divided into two distinct groups. Group 1 is symmetric key cryptography [2] utilizing the secret key for both the encryption and decryption processes. This system provides significant benefits for securing the secret information. One advantage is a high level of security. Two examples of highly secure and still usable algorithms are Triple DES [3] and AES [4, 5]. Furthermore, it provides excellent processing speed. Nevertheless, an important challenge lies in the establishment of a secure communication channel for the exchanging of the secret key between the sender and receiver. Group 2 is asymmetric key cryptography [6], often known as public key cryptography, handles the problem of key exchange by using two mathematically related keys for encrypting and decrypting processes. The key pair includes a public key, which is freely available within the group, and a private key, which is confidential to protect against any intrusions. In addition, some algorithms

from this group can be utilized for authentication through the procedure of digital signature. In this process, the private key is used to sign the digital signature, while the public key is used to verify the signature.

RSA [7] is the widely used public key cryptography algorithm because of its high level of security. This approach is applicable for both data security and digital signature [8, 9]. Examples of applications that use RSA to secure information include online transactions, email encryption, digital signatures, software licensing, and blockchain technology. In addition, the security of the system relies on the difficulty of factoring the large modulus [10–13]. Efficiently factorizing the modulus could potentially allow an unauthorized individual to calculate the private key, therefore affecting the security of the ciphertext. If the prime factors of the modulus are assigned securely and the length of the modulus is at least *2048* bits, no algorithm available that can break RSA in polynomial time [14, 15].

In addition, there are other ways that can be employed to break RSA. Common Modulus Attack (CMA) [16] is a technique employed to ascertain the original plaintext when several receivers possess different public key and private key but share the same modulus. Hastad's Broadcast Attack [17] is a method utilized to determine the original plaintext when the single public key is used for several recipients with the different modulus. Blinding Attack [18] involves manipulating the plaintext before it is signed by the key owner, and then adjusting it to acquire the digital signature of the original message. This approach can be utilized when the key owner declines to sign the digital signature, requiring the message to be altered to get their approval for signing.

This paper presents a new vulnerability that can be exploited to break RSA. The vulnerability arises when two key generators inadvertently generate two moduli containing common prime numbers. Therefore, this study presents a technique known as the Common Prime Attack (CPA) to compute the original plaintext when two moduli have at least one common prime factor. Furthermore, this study proposes targeted solutions for each of the three primary instances of vulnerability challenges. Therefore, the major contributions of this paper are as follows:

- A set of new identified weak points in RSA key generation is presented.
- These weaknesses are categorized into three cases:
    - Case 1: At least two moduli are generated from two prime numbers, sharing one common prime factor.
    - Case 2: At least two moduli are generated from at least three prime numbers, with one shared prime among them.

  - Case 3: At least two moduli are generated from at least three prime numbers, sharing at least two common primes.
- Effective methods to recover the original plaintext from all three cases are proposed and mathematically demonstrated.

2 The Related Works

This section reviews the literature related to the research problem, especially focused on RSA, the system that the proposed methods aim to employ for determining the original plaintext in attacks. Furthermore, the algorithms for integer factorization and other techniques capable of breaking RSA are also evaluated. Other important area of study is CMA, which is essential to be employed in conjunction with the proposed methods.

2.1 RSA Cryptography

RSA is the public key cryptography developed by R. Rivest, A. Shamir and L. Adleman in 1977. This technique possesses the incredible capability to simultaneously ensure security and perform digital signatures. The procedure entails three primary stages. Firstly, key generation begins with the selection of random prime numbers $p_1, p_2, \ldots, p_i$ to construct the modulus, $n = \prod_{j=1}^{i} p_j$ and Euler's Totient Function $\Phi(n) = \prod_{j=1}^{i}(p_j - 1)$.

Next, the public key, e, is chosen from the following conditions, $1 < e < \Phi(n)$ and $gcd(e, \Phi(n)) = 1$. In fact, if e is determined, the private key can be computed from $d = e^{-1} \bmod \Phi(n)$. After all parameters are generated, key creators will keep d, p, q and $\Phi(n)$ secretly, while publicly announcing e and n. When the sender wishes to send the plaintext, m, to the receiver (the key owner), the sender utilizes e and n for encrypting m by using the Equation (1).

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

In the Equation (1), c represents as the ciphertext and it will be sent to the receiver who uses d and n to decrypt c for recovering the original message, m, by using the Equation (2).

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

In fact, RSA is still acknowledged as a secure cryptographic scheme, provided that the modulus possesses a minimum size of *2048* bits and is derived from robust prime numbers. In addition, it is important to ensure

that both e and d must be not too small to prevent potential attacks, such as brute force attack and Wiener's attack. It is especially essential to choose a significantly large number for the value of d. If the parameters are not generated in accordance with these limitations, RSA becomes vulnerable to simple attacks. Presently, there are many types of techniques that have been suggested to break RSA. Moreover, each technique displays a different level of efficiency. The next part focuses on the existing techniques utilized to exploit vulnerabilities in RSA when specific parameters are poorly generated.

2.2 Integer Factorization Algorithms

Currently, there are several methods available for factoring the composite integers. In fact, the factoring algorithms can be classified into two groups: Special-Purpose Integer Factoring and General-Purpose Integer Factoring.

Group 1 (Special-Purpose Integer Factoring): This group includes techniques ideally appropriate for factoring modulus that contain vulnerabilities. Utilizing an algorithm designed to exploit the weakness can result in efficient factorization, as these weaknesses may appear in several forms. Assuming n is generated from two prime numbers, p and q, some examples of algorithms in this category are reviewed as follows:

1. Trial Division Algorithm (TDA): TDA [19, 20] is an algorithm that examines whether a given integer is a divisor of n. If the division does not result in an integer, it means that the number under investigation is not a divisor of n. In this case, another number must be chosen to test as a divisor. This process continues until a number in the condition is found. If the divisor is found, it can be concluded that this number and the quotient are factors of n. Currently, TDA is divided into two methods. The first method involves choosing *3* as the first divisor for the test. If the result is not an integer, this value is increased to test with a new number. This method is suitable for cases where a small integer is a prime factor of n. The second method involves dividing the modulus by an integer that is smaller than $\sqrt{n}$. If the division does not result in an integer, the value is reduced to attempt another division. This method is most effective when the factors of n are close to $\sqrt{n}$.

2. Fermat's Factorization Algorithm (FFA): FFA [21, 22] was introduced by Pierre de Fermat, who observed that n can be represented as the subtraction of two perfect squares as follows: Let $x = \frac{p+q}{2}$, and $y = \frac{p-q}{2}$, then n can be expressed according to the Equation (3):

$$n = x^2 - y^2 \tag{3}$$

By utilizing the Equation (3), $p = x + y$ and $q = x - y$ can be calculated. In fact, there are two approaches to calculate p and q. The first method begins by assigning the value of $\lceil \sqrt{n} \rceil$ as x, then finds the integer value of $y = \sqrt{x^2 - n}$. If the outcome is an integer, p and q can be calculated by using the equations mentioned above. On the other hand, the value of x is increased by 1, and y is recalculated again until an integer result is achieved. Nevertheless, a disadvantage of this approach is the large number of iterations caused by incrementing x by a small value on each iteration. Therefore, several methods have been suggested to minimize the frequency of repetitions by eliminating values of x that are definitively not solutions. This leads to a significant decrease in the amount of time required for computing. The second method requires the process of revising the Equation (3) in the following approach: Given that $u = p + q$ and $v = p - q$, then it is implied that $u^2 - v^2 = 4n$ or $u^2 - v^2 - 4n = 0$. In addition, this algorithm begins by initializing $u = 2\lceil \sqrt{n} \rceil$, and $v = 0$. Then, r is computed by using the Equation (4).

$$r = u^2 - v^2 - 4n \tag{4}$$

When $r = 0$, the values of p and q can be calculated by using the formulas $p = \frac{u+v}{2}$ and $q = \frac{u-v}{2}$. On the other hand, if $r \neq 0$, two parts must be considered: if $r < 0$, u is too small and must to be increased before recalculating r. However, if $r > 0$, v is too small and must be increased before recalculating r. Nevertheless, several studies have been suggested the enhancements to the second method, concentrating on two main areas. First, the process to identify better initial values for u and v to minimize the gap between their relation and the target values. The second technique is to exclude irrelevant values of u and v from the calculation to further decrease the number of iterations.

FFA's primary benefit resides in its capacity to efficiently factorize numbers when the two factors have equivalent sizes. In contrast, FFA exhibits poor results and necessitates significantly more computation time when the two components disappear considerably from this standard.

3. Pollard's $p-1$: Pollard's $p-1$ [12] is a factorization algorithm introduced by J. Pollard in 1974. This method is especially valuable when all prime factors of $p - 1$ or $q - 1$ are small. The process is begun by choosing two integers, b and k, to calculate $gcd(b^{k!} - 1, n)$. If the result is equal to 1, the value of k must be incremented, and the computation should be repeated until a non-trivial greatest common divisor is discovered. In addition, recent research has demonstrated advancements in the Pollard's $p - 1$, particularly

in 2022, where an improvement was proposed by introducing the concept of B-smooth value [13]. If all prime factors of $p-1$ or $q-1$ are members in B-smooth, then this enhanced approach is considerably more efficient than Pollard's $p-1$.

4. Pollard's Rho: In 1975, J. Pollard introduced the factoring algorithm which is called Pollard's Rho [23]. The benefit of this algorithm is to require a small processing space. Pollard's Rho is an efficient algorithm when either the modulus or one of its factors has a small size. The procedure of factorization is as follows: the process is begun by selecting $x_0 \in \mathbb{Z}^+$ at random and calculating x_1 by using the formula $x_1 = x_0^2 \ mod \ n$. The next process is to compute x_i, where $i > 1$, by using the Equation (5):

$$x_i = x_{i-1}^2 \ mod \ n \tag{5}$$

The last process will recover find a factor of n by using the Equation (6):

$$r = gcd(x_i - x_i/2, n) \tag{6}$$

According to the Equation (6), if r is in the following condition, $1 < r < n$, then r is a divisor of n. However, if $r = 1$, then i must be increased to calculate x_i. On the other hand, if $r = n$, then modify the value of x_0 and initiate the procedure again.

5. Wiener's Attack: Wiener's Attack [24] refers to an algorithm developed by M.J. Wiener within the 1990s. The main benefit of this method is its ability to efficiently identify factors of n when d is small. This is achieved by utilizing the convergent of the continuous fraction representation of $\frac{e}{n}$. Empirical evidence has demonstrated that if d is less than $\frac{1}{3}n^{0.25}$, RSA can be compromised through the utilization of Wiener's Attack. Furthermore, in 1999, D. Boneh and G. Durfee introduced an algorithm that enhances Wiener's Attack and enables shortly RSA decryption when d is less than $n^{0.292}$.

6. VFactor: VFactor [25] is a method for factorization that was proposed in 2012. This approach has high efficiency in factoring large values of n, particularly when the difference between the two prime factors of n is small. The process is as follows: Initially, a pair of non-even numbers, denoted as x and y, are chosen. Specifically, the value of $x = \lfloor \sqrt{n} \rfloor$ is assigned and then decreased by 1 when it is an even number. However, the variable $y = \lfloor \sqrt{n} \rfloor + 1$ is assigned and it is incremented by 1 when it is an even number. The subsequent step involves calculating the product of x and y,

denoted as $r = xy$. If $r = n$, then x and y are two prime factors of n. Alternatively, one of them must be modified to recalculate r until $r = n$ is found.

7. Elliptic Curve Factorization Method (ECM): ECM [26] is a factorization algorithm introduced by H. Lenstra. This algorithm is a modified version of Elliptic Curve Cryptography (ECC) [27, 28] that is utilized for factorization. The modification involves making small changes to the ECC's equation as follows:

$$y^3 \equiv x^2 + ax + b \, mod \, n \tag{7}$$

Assume $P = (x_p, y_p)$ is a point on the elliptic curve, the process of factorization by using ECM entails point doubling or point addition operations to calculate multiples of a point P, such as $2P, 3P, 4P$, until a point $rP, r \in \mathbb{Z}^+$, is obtained so that it equals the point at infinity. Next, the factors of n can be calculated. The efficiency of ECM is high when the factors are relatively small, typically up to approximately 60 digits.

Group 2 (General-Purpose Integer Factoring): The efficiency of the algorithms in this group is purely dependent on the size of *n*. Given to the efforts of key producers to generate parameters without vulnerabilities, algorithms in the second group are frequently employed for attacking RSA, as opposed to the first group. Some examples of algorithms in this group are as follows:

1. Quadratic Sieve (QS): QS [29] is a factorization technique introduced by C. Pomerance around 1981. The principle of QS is close to FFA but QS utilizes the congruence of squares modulo n, resulting in a greater number of solutions compared to FFA. Let x and y be integers such that n divides $(x^2 - y^2)$, then

$$x^2 \equiv y^2 \, mod \, n \tag{8}$$

Where, $x \neq y \; mod \; n$ and $x \neq -y \; mod \; n$

Because of the availability of multiple solutions for x and y that meet these conditions, QS limits them to be a member in B-smooth. In addition, QS is acknowledged as a remarkably efficient method for factorization when n is approximately 100 digits.

2. General Number Field Sieve (GNFS): GNFS [30] is currently acknowledged as the most efficient approach for factorization in digital computer processing. In fact, this algorithm necessitates proficiency in various mathematical domains, including complex numbers and irreducible polynomial functions. Moreover, GNFS use advanced high-performance digital computers to efficiently factor numbers with about 10^{100} digits. Thus, in order protect

against attacks utilizing GNFS, it is imperative for key generators to produce n with a minimum size of 2048 bits.

2.3 Non-factoring Algorithms for Attacking RSA

This topic will discuss other techniques that do not depend on integer factorization, but they can be employed to exploit RSA. Every method employs a distinct methodology, and they are only applicable in specific contexts. An example that Common Modulus Attack can be employed when the plaintext, m, is encrypted for two recipients. The various non-factoring strategies are as follows:

1. Hastad's Broadcast Attack [17]: If the plaintext, m, is transmitted to several recipients, all of which have a different value of n, but utilizing the same e. If an attacker can intercept all the encrypted messages, they can employ Hastad's Broadcast Attack to retrieve the value of m. Furthermore, Chinese Remainder Theorem (CRT) [31, 32] is an essential computational step in Hastad's Broadcast Attack. Thus, to avoid attacking by using Hastad's Broadcast, it is imperative for each recipient to create a distinct e and n pair.
2. Blinding Attack [18]: Blinding Attack is a technique employed to exploit vulnerabilities in RSA when it is utilized for generating digital signatures. The goal of this method is to trick the key owner into signing the attacker's plaintext. Suppose m represent the plaintext that the attacker intends for the key owner to sign, a request that the owner would naturally refuse to sign directly. Alternatively, the assailant initially chooses a random value r such that $gcd(r, n) = 1$. Then, the attacker calculates $m_r = r^e m \ mod \ n$. Next, attacker gets the signature of the key owner, resulting in $s_r = m_r^d \ mod \ n$. Subsequently, the assailant can compute $s = m^d \ mod \ n$ from $s_r r^{-1} \ mod \ n$, which represents the signature on m.
3. Brute Force Attack: Brute Force Attack [33, 34] is a method for discovering d. The procedure involves encrypting the plaintext, m, to obtain the ciphertext, c, followed by attempting to decrypt c by using every possible integer as the exponent. Assume a is the first integer for validating, if the outcome does not correspond to m, then it is certain that a is not equal to d. On the other hand, if the outcome corresponds to m, then a is equal to d. The attacker can confirm this value by conducting a separate test with a different m. If the outcome is different, then a is not equal to d. Regardless of their lack of success, values can still be maintained to calculate periods. In fact, Brute Force Attack is effective when the value

of d is small, as it begins with small integers and increments them when no match is discovered. For large values of d, begin with large integers and decrease them when no equivalent is discovered.

4. Davida's Algorithm: An attacker utilizes Davida's Algorithm to recover the plaintext, m, from the ciphertext, c, that has been intercepted during communication. The process includes the subsequent stages: If the attackers successfully intercept c, they tamper with this value and replace it with c' before transferring it to the intended recipient. The recipient, who is unaware of the alteration, incorrectly believes that c' is the actual ciphertext that was correctly transmitted. The recipient performs decryption on c', resulting in m', which is an incoherent message. The recipient thereafter disregards m'. However, if the attackers obtain m', they can use it to retrieve m. The Equation (9) is used to calculate c', while the Equation (10) is used to determine m:

$$c' = cr^e \ mod \ n \tag{9}$$

$$m = m'r^{-1} \ mod \ n \tag{10}$$

where $r \in \mathbb{Z}^+$ and $gcd(r, n) \neq 1$.

2.4 Common Modulus Attack

Common Modulus Attack (CMA) [16] is an alternative method used to compromise the security of RSA. Indeed, this algorithm belongs to the category of non-factoring algorithms. In fact, CMA is a method employed to compute plaintext, m, when m has been encrypted to transfer to several receivers, each utilizing the distinct public keys but the same modulus.

Let e_1, n denote the public key and the modulus of recipient 1, and e_2, n denote the public key and the modulus of recipient 2, with $gcd(e_1, e_2) = 1$. Therefore, c_1 and c_2 represent the values to be sent to recipient 1 and recipient 2, respectively, which can be computed using the Equations (11) and (12).

$$c_1 = m^{e_1} \ mod \ n \tag{11}$$

$$c_2 = m^{e_2} \ mod \ n \tag{12}$$

If c_1, c_2 are disclosed, an intruder can use the Extended Euclidean Algorithm [35] to calculate x and y such that $e_1x + e_2y = 1$. If x and y are known, it follows that

$$c_1^x c_2^y \ mod \ n = (m^{e_1})^x (m^{e_2})^y \ mod \ n$$

$$= m^{e_1 x} m^{e_2 y} \ mod\ n$$

$$= m^{e_1 x + e_2 y} \ mod\ n = m$$

Therefore, it is not recommended to use the same modulus for encrypting original message, even if the private keys are different, to prevent assaults by intruders.

2.5 Contemporary Research Trends in RSA Security

Almost research on RSA has been focused on mathematical cryptanalysis, however current studies have enhanced the emphasis to include behaviours affecting cryptographic security. For example, in 2024, M. Prybylo et al. [36] executed a mixed-method survey with 362 participants from 23 countries to evaluate the privacy views and practices of software development teams. Their results indicate that variations in privacy knowledge among team members may affect the installation and security of cryptographic systems such as RSA. Additionally, in 2023, H.M. Arjomandi et al. [37] investigated low-epsilon adversarial attacks on neural network classifiers processing online image streams. Their study reveals that even minimal perturbations can compromise system integrity, underscoring the requirement for robust security measures in real-time data processing environments, which may also be pertinent to RSA applications in similar contexts. Current studies emphasise the significance of integrating human factors and data-driven risk assessments in the evaluation and improvement of RSA security.

3 The Proposed Method

This study reveals the new weaknesses in RSA and provides an enhanced method known as Common Prime Attack (CPA) to retrieve the plaintext. In fact, CPA can be employed when there are numerous moduli, with at least one sharing a common factor. This idea differs from the concept used in CMA for computing the plaintext, as CMA is employed when the same original messages is sent to multiple recipients by using the same modulus but different public keys. The application of CPA is classified into three cases based on the properties of the moduli.

Case 1: The moduli are generated from two prime numbers, one of which is a common factor. For the case that there are two moduli (designated as n_1 and n_2) generated from two prime numbers, with one prime being a common

Table 1 The Process to find the private keys when common prime is disclosed

	The Private Keys of the Key Creators	
No.	Creator 1	Creator 2
1	$q = \frac{n_1}{p}$	$r = \frac{n_2}{p}$
2	$\Phi(n_1) = (p-1)(q-1)$	$\Phi(n_2) = (p-1)(r-1)$
3	$d_1 = e_1^{-1}\ mod\ \Phi(n_1)$	$d_2 = e_2^{-1}\ mod\ \Phi(n_2)$

factor, it follows that an adversary can compute the private keys for both n_1 and n_2. This can be accomplished by using Theorem 1.

Theorem 1. Let $\{e_1, n_1\}$ represent the public key and the modulus of key generator 1, where $n_1 = pq$, and $\{e_2, n_2\}$ denote the public key and the modulus of key generator 2, where $n_2 = pr$, and p, q and r are prime numbers. In this case, it is possible to compute d_1 and d_2, which are the private keys of key generator 1 and key generator 2, respectively.

Proof: From $gcd(n_1, n_2) = gcd(pq, pr)$, because both q and r are prime numbers, it follows that both are coprime to each other, or $gcd(q, r) = 1$. Therefore, $gcd(n_1, n_2) = p$.

Therefore, if an adversary knows n_1 and n_2, they can compute the private keys of both key generators as follows:

Example 1. Let $n_1 = 6169, e_1 = 13, n_2 = 14353$ and $e_2 = 17$, where n_1 and n_2 are generated from two prime numbers and $gcd(n_1, n_2) \neq 1$. Determine d_1 and d_2 by using CPA

Sol: Because $gcd(6169, 14353) = 31, p = 31$ and other secret parameters can be calculated as follows:

	The Private Keys of the Key Creators	
No.	Creator 1	Creator 2
1	$q = \frac{6169}{31} = 199$	$r = \frac{14353}{31} = 463$
2	$\Phi(n_1) = (30)(198) = 5940$	$\Phi(n_2) = (30)(462) = 13860$
3	$d_1 = 13^{-1}\ mod\ 5940 = 457$	$d_2 = 17^{-1}\ mod\ 13860 = 8153$

In addition, case 1 demonstrates that the attacker can decipher the ciphertext and reliably compute the original plaintext.

Case 2: The moduli are computed from many prime integers, one of which is a common factor.

For the case that n_1 and n_2 are generated from at least 3 prime numbers, with one prime being a common factor, it follows that if $m < p$, where p is a

prime factor of both n_1 and n_2, then the adversary can compute m by using Theorem 2.

Theorem 2. Let $\{e_1, n_1\}$ represent the public key and the modulus of key generator 1, $n_1 = pqr, \{e_2, n_2\}$ denote the public key and the modulus of key generator 2, $n_2 = pst$, where p, q, r, s and t are prime numbers and $gcd(e_1, e_2) = 1$. It follows that if $c_1 = m^{e_1}$ *mod* n_1 and $c_2 = m^{e_2}$ *mod* n_2 are disclosed, then it is possible to compute m when $1 < m < p$.

Proof: Suppose the attacker still does not know the factors of n_1 and n_2,

The proof begins by finding $gcd(n_1, n_2) = gcd(pqr, pst) = p$

Then, $n_1 = pX$ and $n_2 = pY$, where $X = qr$ and $Y = st$

From

$$c_1 = m^{e_1} \textit{ mod } n_1$$

Then,

$$c_1 = m^{e_1} \textit{ mod } pqr$$

Or,

$$\begin{aligned} m^{e_1} &= pqrk + c_1, \quad \text{when } k \in \mathbb{Z} \\ &= lp + c_1, \quad \text{when } l = qrk \end{aligned}$$

Therefore,

$$c_1 = m^{e_1} \textit{ mod } p \tag{13}$$

From,

$$c_2 = m^{e_2} \textit{ mod } n_2$$

Then,

$$= m^{e_2} \textit{ modpst}$$

Or,

$$\begin{aligned} m^{e_2} &= pstu + c_2, \quad \text{when } u \in \mathbb{Z} \\ &= vp + c_2, \quad \text{when } v = stu \end{aligned}$$

Therefore,

$$c_2 = m^{e_2} \textit{ mod } p \tag{14}$$

From Equations (13) and (14), it follows that if the attacker obtains c_1 and c_2, CMA can be chosen to compute m.

Although q, r, s, and t are not given in case 2, if p, a common factor of n_1 and n_2, is released, it is feasible to calculate m.

Case 3: The moduli are generated from several prime numbers, with a minimum of 2 primes being common factors.

For the case that n_1 and n_2 are generated from at least 3 prime numbers, with at least 2 primes being common factors for both n_1 and n_2, if $m < n$, where n is the product of the common factors of n_1 and n_2, it follows that an attacker can always compute m by using Theorem 3.

Theorem 3. Let $\{e_1, n_1\}$ represent the public key and the modulus of key generator 1, where $n_1 = nr$, and $\{e_2, n_2\}$ denote the public key and the modulus of key generator 2, where $n_2 = ns$, with p, q, r, and s being prime numbers, $n = pq$ and $gcd(e_1, e_2) = 1$. If $c_1 = m^{e_1}$ *mod* n_1 and $c_2 = m^{e_2}$ *mod* n_2 are disclosed, it is possible to calculate m when $1 < m < n$.

Proof: Suppose the attacker still does not know the factors of n_1 and n_2,

The proof begins by finding $gcd(n_1, n_2) = gcd(nr, ns) = n$

Then,

$$n_1 = nr \quad and \quad n_2 = ns$$

From,

$$c_1 = m^{e_1} \ mod \ n_1$$

Then,

$$c_1 = m^{e_1} \ mod \ nr$$

Or,

$$\begin{aligned} m^{e_1} &= nrk + c_1, \quad \text{when } k \in \mathbb{Z} \\ &= ln + c_1, \quad \text{when } l = rk \end{aligned}$$

Therefore,

$$c_1 = m^{e_1} \ mod \ n \tag{15}$$

From,

$$c_2 = m^{e_2} \ mod \ n_2$$

Then,

$$= m^{e_2} \ mod \ ns$$

Or,

$$\begin{aligned} m^{e_2} &= nsu + c_2, \quad \text{when } u \in \mathbb{Z} \\ &= vn + c_2, \quad \text{when } v = su \end{aligned}$$

Therefore,

$$c_2 = m^{e_2} \ mod \ n \tag{16}$$

From Equations (15) and (16), it is observed that the equations resemble those in Equations (13) and (14), indicating that m can be calculated using CMA.

In Cases 2 and 3, if all common prime factors of the moduli are identified, m can be recovered by using Extended Euclidean Algorithm [35]. This procedure follows the same computational principle as employed in CMA.

Example 2. Given that $n_1 = 476227$, $e_1 = 17$, $n_2 = 905057$ and $e_2 = 13$ where n_1 and n_2 are generated from 3 prime numbers and $gcd(n_1, n_2) \neq 1$, suppose the attacker knows $c_1 = 408007$ and $c_2 = 785533$, Find m by using CPA.

Sol: Because, $gcd(n_1, n_2) = gcd(476227, 905057) = 2257 = n$, then $r = \frac{476227}{2257} = 211$ and $s = \frac{905057}{2257} = 401$.

From,

$$c_1 = m^{e_1} \ mod \ n$$

Then,

$$408007 = m^{e_1} \ mod \ 2257 \ or \ 408007 \ mod \ 2257 = 1747$$

From,

$$c_2 = m^{e_2} \ mod \ n$$

Them,

$$785533 = m^{e_2} \ mod \ 2257 \ or \ 785533 \ mod \ 2257 = 97$$

After using Extended Euclidean Algorithm to compute x and y to get the result of $e_1 x + e_2 y = 1$, then $x = -3$ and $y = 4$.

Therefore,

$$\begin{aligned}(1747)^{-3}(97)^4 \ mod \ 2257 &= (1747^{-1})^3(97)^4 \ mod \ 2257 \\ &= (2049)^3(97)^4 \ mod \ 2257 \\ &= (2004)(713) \ mod \ 2257 \\ &= 171\end{aligned}$$

Therefore, after implementing CPA, then $m = 171$,

Checking, 171^{17} *mod* $476227 = 408007$ and 171^{13} *mod* $905057 = 785533$

4 Experimental Results

This section will examine experiments related to the computation time for three distinct cases, utilizing the moduli of 1024, 2048, and 4096 bits. The selected moduli utilized for experimentation will possess qualities that align with the benefits of each case. In Case 1, two values will have a prime factor in common. Therefore, no other algorithms are considered because of their significantly greater processing time, in contrast to the proposed method which exhibits exceptional speed. The common prime factor is always equal to half the size of the modulus. For example, if the modulus is 2048 bits, the common factor will be 1024 bits. When the modulus is formed using more than two primes, it may be noticed that the remaining factors are consistently lower than the shared factor.

In addition, this method uses the BigInteger class in Java, which can represent variables of unlimited size. The experiments were conducted on a 2.53 GHz Intel® Core i5 processor with 8 GB of RAM to maintain consistent control over the parameters. All experiments were implemented in Java 11 and executed on a Windows 10 64-bit operating system. The programs were run in single-threaded mode without parallel processing or hardware acceleration. Random values were generated by using SecureRandom class, with a fixed seed to ensure consistency across repeated runs.

Figure 1 displays the experimental findings for Case 1, in which n_1 and n_2 are formed by two prime numbers that have a single prime factor in common. The average computation time to locate the answer is 1.2 milliseconds, 1.6 milliseconds, and 3.6 milliseconds for modulus sizes of 1024 bits, 2048 bits, and 4096 bits, respectively, based on the results.

Figure 2 displays the experimental outcomes for Case 2, in which n_1 and n_2 are formed by combining three prime integers, with one prime factor that

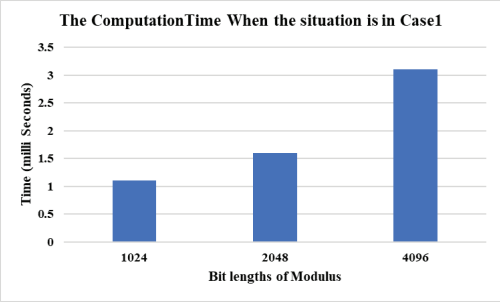

Figure 1 The computation time when the situation is in case 1.

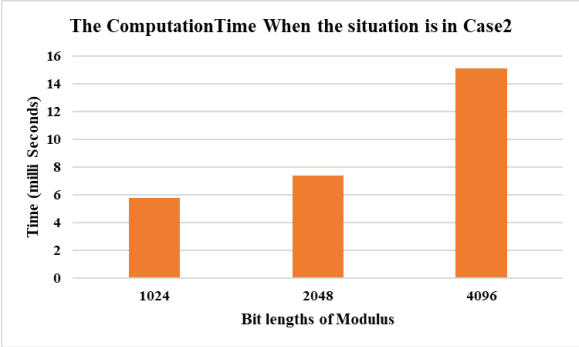

Figure 2 The computation time when the situation is in case 2.

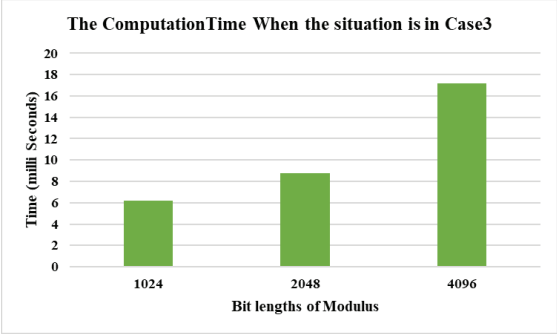

Figure 3 The computation time when the situation is in case 3.

is shared by n_1 and n_2. The plaintext used for testing must be smaller than the shared divisor. The average time required to get the answer for modulus sizes of 1024 bits, 2048 bits, and 4096 bits is 5.8 milliseconds, 7.4 milliseconds, and 15.1 milliseconds, respectively.

Figure 3 displays the experimental results obtained in Case 3, where n_1 and n_2 are formed by combining three prime numbers that share two prime components. In this experiment, the size of the plaintext was intentionally set to be smaller than the product of the two commonly shared prime factors. The findings indicate that for modulus values of 1024 bits, 2048 bits, and 4096 bits, the average time required was 6.2 milliseconds, 8.8 milliseconds, and 17.2 milliseconds, respectively.

When comparing all three situations, it is shown that Case 3 requires the highest computation time. CMA must be conducted to reach the final solution after locating the common factors in Cases 2 and 3. Given that n

is equal to the product of two prime numbers, p and q, it follows that n is always greater than p. In addition, Case 3 requires more computation time than Case 2 because of the bigger magnitude of the divisor. Nevertheless, each of the three cases necessitates a minimal amount of time to discover the solution. Hence, to thwart attacks employing CPA, it is imperative to steer clear of circumstances that yield outcomes aligning with these three cases while generating RSA parameters.

5 Analysis

From three cases above, it can be observed that case 1 can be used to compute m in all situations because d_1 and d_2 are disclosed. However, cases 2 and 3 cannot directly compute d_1 and d_2 because it is not possible to calculate the factors of n_1 and n_2 completely. However, after the common prime factors of n_1 and n_2 are disclosed, CPA can calculate m, when m is lower in size than the revealed common factors of n_1 and n_2.

Table 2 provides a comparative analysis of the effectiveness of CPA, CMA, and Hastad's Broadcast Attack, highlighting key aspects such as Core Principle, Required Information, Key Assumptions, Decryption Feasibility, CPA Advantage, and Flexibility.

6 Limitations

Although CPA is proven that it is an efficient algorithm under specific situations, CPA has limitations. In fact, the applicability of CPA strictly depends on three cases in this study which are about the characteristic of moduli. These conditions include the reuse of at least one common prime factor across two or more moduli, either among two primes or within larger composite structures. If the moduli are generated by using completely distinct prime numbers without any overlap, CPA becomes inapplicable. Furthermore, in Cases 2 and 3, the technique requires that the original plaintext, m, be smaller than the shared prime factor(s) to ensure successful recovery. This concept causes further limitations on practical applicability.

7 Implications for Cryptographic Practices and Standards

The findings from this research emphasise a significant vulnerability in RSA implementations due to the generation of all moduli without the validation

Table 2 The comparison during CPA, CMA and Hastad's Broadcast Attack

Criteria	CPA	CMA	Hastad's Broadcast Attack
Core Principle	Exploits the reuse of at least one common prime factor between two or more moduli	Exploits the use of the same modulus with different public exponents	Exploits the use of the same public exponent with different moduli
Required Information	At least two moduli that share one or more prime factors	Multiple ciphertexts encrypted with the same modulus and different public exponents	Multiple ciphertexts encrypted with the same public exponent and different moduli
Key Assumptions	Moduli share at least one common prime number	$gcd(e_1, e_2) = 1$; identical modulus used across all receivers	Number of intercepted ciphertexts must be larger than public exponent e; relies on CRT
Decryption Feasibility	Does not require full factorization when shared primes are disclosed	Requires knowledge of both exponents and the common modulus	Requires multiple ciphertexts and relies on CRT
CPA Advantage	Does not require the modulus or public exponent to be identical across users	Only applicable when the same modulus is reused	Only applicable when the same public exponent is reused across multiple recipients
Flexibility	Can be applied to moduli composed of two or more primes without requiring to be identical	Restricted to scenarios where the modulus is the same	Restricted to cases with identical public exponents and sufficient ciphertexts

for the originality of prime factors. This problem is especially relevant in distributed settings, where key generation occurs across several systems, or in situations where inadequate randomisation, repetitive seeds, or insufficient entropy might result in the accidental reuse of prime numbers. In fact, CPA indicates that if two moduli share a single prime factor, the original plaintext may be effectively retrieved under certain mathematical settings. While these situations may be infrequent in carefully constructed systems, they still present a credible risk in practical implementations when key generation lacks rigorous validation protocols. To address this risk, cryptographic key generation protocols must be improved to include specific verifications that guarantee the singularity of all prime factors used across various key pairs.

8 Future Work

Although CPA has significant efficacy under specific conditions, the identification of common prime factors among several moduli depends on mathematical analysis. This may become computationally demanding when handling moduli from many and dispersed sources. Future study may investigate the application of machine learning to facilitate the early identification or forecasting of moduli that are likely to contain common prime factors. Unsupervised learning techniques may be implemented to identify anomalous modulus structures without requiring explicit factorisation. This methodology would enhance the efficiency of CPA in extensive contexts and transform it from an elementary mathematical attack technique into a comprehensive system-level cryptographic risk assessment instrument.

9 Conclusion

The study reveals a novel weakness in RSA encryption and proposes a method known as CPA to attack this issue by using the identified faults. The new weakness is due to two key generators producing moduli with different but at least one common prime factor. The process is divided into 3 cases as follows: Case 1 involves both moduli being generated from 2 prime numbers, with at least one shared prime factor. This case allows the use of CPA to compute the private keys of both moduli. Cases 2 and 3 entail generating moduli from more than 2 prime numbers, with some prime factors being shared (Case 2 involves only one shared prime factor, while Case 3 involves at least 2 shared prime factors). In these instances, private keys cannot be calculated directly, but it is feasible to recover m, when m is lower than the shared factors' size. In addition, CPA may be relevant to other cryptographic methods utilising moduli derived from prime numbers. For example, Paillier cryptosystem employs a modulus derived from two large prime numbers, which may exhibit similar weaknesses when prime reuse occurs. Investigating these options may provide significant insights for enhancing the resilience of prime-based public-key systems.

References

[1] M. Hellman, An extension of the Shannon theory approach to cryptography. IEEE Transactions on Information Theory, 23(3) (1977), 289–294, https://doi.org/10.1109/TIT.1977.1055709.

[2] G.J. Simmons, Symmetric and Asymmetric Encryption. ACM Computing Surveys, 11(4) (1979), 305–330, https://doi.org/10.1145/356789.356793.

[3] D. Coppersmith, D.B. Johnson, S.M. Matyas, A proposed mode for triple-DES encryption. IBM Journal of Research and Development, 40(2) (1996), 253–262, https://doi.org/10.1147/rd.402.0253.

[4] C. Sanchez-Avila, R. Sanchez-Reillol, The Rijndael block cipher (AES proposal): a comparison with DES, In Proceedings of IEEE 35th Annual 2001 international carnahan conference on security technology, October 16–19, 2001, United Kingdom: IEEE, pp. 229–234, https://doi.org/10.1109/CCST.2001.962837.

[5] S. Mangard, M. Aigner, S. Dominikus, A highly regular and scalable AES hardware architecture. IEEE Transactions on Computers, 52(4) (2003), 483–491, https://doi.org/10.1109/TC.2003.1190589.

[6] W. Diffie, M. Hellman, New directions in cryptography. IEEE Transactions on Information Theory, 22(6) (1976), 644–654, https://doi.org/10.1109/TIT.1976.1055638.

[7] R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2) (1978), 120–126, https://doi.org/10.1145/359340.359342.

[8] K. Somsuk, M. Thakong, Authentication system for e-certificate by using RSA's digital signature. TELKOMNIKA Telecommunication, Computing, Electronics and Control, 18(6) (2020), 2948–2955, https://doi.org/10.12928/TELKOMNIKA.v18i6.17278.

[9] K. Somsuk, S. Atsawaraungsuk, C. Suwannapong, S. Khummanee, C. Sanemueang, The Variant of Digital Signature Algorithm for Constant Message. Journal of Internet Services and Information Security, 13(2) (2023), 81–95, https://doi.org/10.58346/JISIS.2023.I2.005.

[10] M.E. Wu, R. Tso, H.M. Sun, On the improvement of Fermat factorization using a continued fraction technique, Future Generation Computer Systems, 30(1) (2014), 162–168, https://doi.org/10.1016/j.future.2013.06.008.

[11] R.R.M. Tahir, M.A. Asbullah, M.R.K. Ariffin, Z. Mahad, Determination of a Good Indicator for Estimated Prime Factor and Its Modification in Fermat's Factoring Algorithm. Symmetry, 13(5) (2021), 1–22, https://doi.org/10.3390/sym13050735.

[12] J.M. Pollard, Theorems of factorization and primality testing. Math. Proc. Camb. Philos. Soc., 76 (1974), 521–528, https://doi.org/10.1017/S0305004100049252.

[13] 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, 14(2) (2022), 312, https://doi.org/10.3390/sym14020312.
[14] F.O. Mojisola, S. Misra, C.F. Febisola, O. Abayomi-Alli, G. Sengul, An improved random bit-stuffing technique with a modified RSA algorithm for resisting attacks in information security (RBMRSA). Egyptian Informatics Journal, 23(2) (2022), 291–301, https://doi.org/10.1016/j.eij.2022.02.001.
[15] G.C. Kim, S.C. Li, H.C. Hwang, Fast rebalanced RSA signature scheme with typical prime generation. Theoretical Computer Science, 830–831 (2020), 1–19, https://doi.org/10.1016/j.tcs.2020.04.024.
[16] A. Abubakar, S. Jabaka, B. I. Tijjani, A. Zeki, H. Chiroma, M. J. Usman, S. Raji, M. Mahmud, Cryptanalytic Attacks on Rivest, Shamir, and Adleman (RSA) Cryptosystem: Issues and Challenges. Journal of Theoretical and Applied Information Technology, 61(1) (2014), 1–7.
[17] I. K. Salah, A. Darwish, S. Oqeili, Mathematical Attacks on RSA Cryptosystem. Journal of Computer Science, 2(8) (2006), 665–671, https://doi.org/10.3844/jcssp.2006.665.671.
[18] D. Boneh, Twenty Years of Attacks on the RSA Cryptosystem. Notices of the AMS, 46(2) (2006), 203–213.
[19] 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 International Conference on Contemporary Computing and Informatics, November 27–29, 2014, India: IEEE, pp. 992–995, https://doi.org/10.1109/IC3I.2014.7019588.
[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 Proceedings of International Joint Conference on Computer Science and Software Engineering, July 10–12, 2019, Thailand: IEEE, pp. 143–147, https://doi.org/10.1109/JCSSE.2019.8864218.
[21] K. Somsuk, The new integer factorization algorithm based on fermat's factorization algorithm and euler's theorem. Int J Electr Comput Eng, 10(2) (2020), 1469–1476, https://doi.org/10.11591/ijece.v10i2.
[22] K. Somsuk, K. Tientanopajai, An Improvement of Fermat's Factorization by Considering the Last m Digits of Modulus to Decrease Computation Time. Int. J. Netw. Secur., 19(1) (2017), 99–111, https://doi.org/10.6633/IJNS.201701.19(1).11.

[23] J.M. Pollard, Monte Carlo methods for index computation ($modp$). Mathematics of computation, 32(143) (1978), 918–924, https://doi.org/10.2307/2006496.

[24] M. Wiener, Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, 36 (1990), 553–558, https://doi.org/10.1109/18.54902.

[25] P. Sharma, A.K. Gupta, A.Vijay, Modified Integer Factorization Algorithm using V-Factor Method, In Proceedings of International Conference on Advanced Computing & Communication Technologies, January 7–8, 2012, India: IEEE, pp. 423–425, https://doi.org/10.1109/ACCT.2012.73.

[26] H. W. Lenstra Jr., Factoring integers with elliptic curves. Annals of Mathematics, 126(3) (1987), 649–673, https://doi.org/10.2307/1971363.

[27] N. Koblitz, Elliptic Curve Cryptosystems. Mathematics of Computation, 48 (1987), 203–209, http://dx.doi.org/10.1090/S0025-5718-1987-0866109-5.

[28] V.S. Miller, Uses of elliptic curves in cryptography. Lecture Notes in Computer Science, 218 (1986), 417–428, https://doi.org/10.1007/3-540-39799-X_31.

[29] R. D. Silverman, The multiple polynomial quadratic sieve. Mathematics of Computation, 48(1777) (1987), 329–339, https://doi.org/10.1090/s0025-5718-1987-0866119-8.

[30] T. Kleinjung, On polynomial selection for the general number field sieve. Mathematics of Computation, 75(256) (2006), 2037–2047, https://doi.org/10.1090/S0025-5718-06-01870-9.

[31] S.M. Yen, S. Kim, S. Lim, S.J. Moon, RSA speedup with Chinese remainder theorem immune against hardware fault cryptanalysis. IEEE Transactions on computers, 52(4) (2003), 461–472, https://doi.org/10.1109/TC.2003.1190587.

[32] L. Harn, M. Fuyou, C.C. Chang, Verifiable secret sharing based on the Chinese remainder theorem. Security and Communication Networks, 7(6) (2014), 950–957, https://doi.org/10.1002/sec.807.

[33] K. Somsuk, A New Methodology to Find Private Key of RSA Based on Euler Totient Function. Symmetry, 18(2) (2021), 338–348, http://dx.doi.org/10.21123/bsj.2021.18.2.0338.

[34] K. Somsuk, The Improved Estimation of the Least Upper Bound to Search for RSA's Private key. KSII Transactions on Internet and

Information Systems, 16(6) (2022), 2074–2093, http://doi.org/10.3837/tiis.2022.06.016.

[35] D. Poulakis, An application of Euclidean algorithm in cryptanalysis of RSA. Elemente der Mathematik, 75(3) (2020), 114–120, https://doi.org/10.4171/EM/411.

[36] M. Prybylo, S. Haghighi, S.T. Peddinti and S. Ghanavati, Evaluating privacy perceptions, experience, and behavior of software development teams, In Proceedings of the Twentieth USENIX Conference on Usable Privacy and Security, August 11–13, 2024, Philadelphia, PA, USA: pp. 101–120, https://dl.acm.org/doi/10.5555/3696899.3696905.

[37] H.M. Arjomandi, M. Khalooei and M. Amirmazlaghani, Low-epsilon adversarial attack against a neural network online image stream classifier. Applied Soft Computing, 147 (2023), 110760, https://doi.org/10.1016/j.asoc.2023.110760.

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.