Trustworthy Approaches to RSA: Efficient Exploitation Strategies Based on Common Modulus

Mahdi Mahdavi
Universitat Oberta de Catalunya
Shahid Beheshti University
m_mahdavi@uoc.edu

Navid Abapour
Surrey Centre for Cyber Security,
University of Surrey
n.abapour@surrey.ac.uk

Zahra Ahmadian
Electrical Engineering Department,
Shahid Beheshti University
z_ahmadian@sbu.ac.ir

***Abstract*—With the increasing integration of crowd computing, new vulnerabilities emerge in widely used cryptographic systems like the RSA cryptosystem, whose security is based on the factoring problem. It is strongly advised to avoid using the same modulus to produce two pairs of public-private keys, as the cryptosystem would be rendered vulnerable to common modulus attacks. Such attacks can take two forms: one that aims to factorize the common modulus based on one key pair and the other that aims to decrypt certain ciphertexts generated by two public keys if the keys are co-prime. This paper introduces a new type of common modulus attack on the RSA cryptosystem. In our proposed attack, given one public-private key pair, an attacker can obtain the private key corresponding to a given public key in RSA decryption. This allows the adversary to decrypt any ciphertext generated using this public key. It is worth noting that the proposed attack can be used in the CRT model of RSA. In addition, we propose a parallelizable factoring algorithm with an order equivalent to a cyclic attack in the worst-case scenario.**

1. Introduction

The RSA algorithm, invented in 1997 by Rivest, Shamir, and Adelman, is one of the earliest public key encryption schemes known [1]. It is a public key cryptosystem extensively studied and widely implemented for securing the digital realm. It has been shown that a quantum computer is required to factorize the 2048-bit RSA modulus [2]. Some implementations of RSA are vulnerable to attacks even without quantum computation.

As an overview of Euler's theorem and the RSA cryptosystem, according to [3], let $\mathbb{Z}_n^*$ denote the set of all positive integers that are less than and coprime to n, and $\phi(n) = |\mathbb{Z}_n^*|$. The Euler's theorem states that $a^{\phi(n)} = 1 \bmod n$ for all $a \in \mathbb{Z}_n^*$. Let $N = pq$, where p and q are two large prime numbers. Then, $\phi(N) = (p-1)(q-1)$. In the RSA cryptosystem, the public key is represented by the pair (e, N), where $gcd(e, \phi(N)) = 1$, and the d represents the private key, where $e \cdot d = 1 \bmod \phi(N)$. This means that $e \cdot d = k \cdot \phi(N) + 1$ for an integer k. The public-private key pair is usually denoted by (e, d), for simplicity. In the RSA cryptosystem, the ciphertext for message m is defined as $C = m^e \pmod N$, the decryption is simply performed as $C^d = m^{ed} = m^{1+k\cdot\phi(N)} = m \cdot (m^k)^{\phi(N)} = m \cdot 1 = m \pmod N$. It is evident that modular exponentiation, a computationally expensive operation, is required for both encryption and decryption processes. Recently, [4] proposed a technique for efficient computation of modular exponentiation. To increase efficiency in encryption, the value of e is selected as small as possible, for instance, $e = 3$. However, the most common choice for public key is the value of $e = 2^{16} + 1$ [5].

It should be noted that when the public key is selected, the value of the secret key may become a large number that is out of control. Section 1.3 explains how to reduce the computational complexity of decryption.

As of contribution, we propose a common modulus attack that surpasses existing methods in efficiency and broad applicability across various RSA variants, such as the Carmichael function, black-box CRT mode, and Euler's ϕ function. Unlike traditional attacks, it does not require specific access to many elements, making it more versatile and dangerous by leveraging any available cryptosystem data. Additionally, we present a factoring algorithm based on this attack, which achieves a worst-case time complexity similar to cyclic attacks and operates independently of private or public keys, offering a potentially more efficient solution in certain cases.

1.1. Real World Applications of RSA

Adopting RSA encryption within the perspective of cloud systems is of great interest since large amounts of sensitive information are vulnerable due to big data security risks. There exist many real-world applications still using traditional public-key cryptosystems like RSA [1]; these include but are not limited to Google Workspace Single Sign-On [6], Telegram [7], OpenVPN [8], Secure Shell (SSH) [9], and OpenPGP [10]. However, the fast pace of big data implementation within cloud environments puts organizations at risk of various cyber-attacks, including denial-of-service and injection attacks. In [11], the use of RSA algorithms to enhance security features for large volumes, high velocity, and veracity of big data has been suggested to help mitigate all risks. They conclude from their results that RSA enhances the security of computing services on the cloud, e.g. AWS [12, 13, 14]. Also, regarding wireless sensor networks, RSA still has a high potential. Devi and Sampradeepraj in [15] presented a hybrid encryption protocol based on RSA with an integrated symmetric-asymmetric methodology. This two-phase approach bolsters efficiency in both the encryption process and key management and underlines the fact that strong RSA encryption plays a serious role in enabling

secure data transit on cloud services. Moreover, even when it comes to fog computing, in [16], researchers refer to secure authentication and load balancing-related problems as an effective solution using RSA. The integration of those methodologies also shows the potential of RSA in enhancing cloud computing frameworks upon newly identified threats to improve the security management of the data by better and more effective practices.

1.2. On the Structure of RSA

Factoring a large number into its prime factors is a hard problem when the factors p and q are sufficiently large. The size of p and q directly influences the security level of the encryption. Recently, a technique based on the smooth integers has been developed to address this issue [17].

The RSA cryptosystem also works for the Carmichael function instead of the Euler function. Carmichael function of a positive integer n which is shown with $\lambda(n)$ is the smallest positive integer r such that for any integer a coprime to n the relation of $a^r = 1 \mod (n)$ is satisfied. This is clear that $\lambda(N)|\phi(N)$. In the RSA case $\lambda(N) = lcm(p-1, q-1)$ where $lcm(a, b)$ denotes the least common multiplier of a and b [18]. This means that the public-private key pair (e, d) is selected as $e \cdot d = 1 \bmod \lambda(N)$. So, we will have $C^d = m \cdot (m^k)^{\lambda(N)} = m$. In the rest of this paper, we will be using the Euler function. However, it is worth noting that all of our arguments remain valid for the Carmichael function if we substitute $\phi(N)$ with $\lambda(N)$ in our analysis.

The RSA cryptosystem is a multiplicative homomorphic scheme [4]. This means that if you have the ciphertexts of two messages, you can compute the encryption of their multiplication without knowing the plaintexts. Assume that we have two messages, m_1 and m_2, and their respective RSA encryptions under the same key, C_1 and C_2. This means that $C_1 = m_1^e$ and $C_2 = m_2^e$. To obtain the ciphertext of their product, $m_1 \cdot m_2$, we simply multiply the two ciphertexts: $C_1 \cdot C_2 = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e$.

1.3. CRT-decryption

In [19], the Chinese Reminder Theorem (CRT) is used to increase decryption efficiency. This technique involves performing computations in $\mathbb{Z}_p$ and $\mathbb{Z}_q$, instead of $\mathbb{Z}_N$, which reduces complexity in two ways. Firstly, computing in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ is more efficient than computing in $\mathbb{Z}_N$. Secondly, the Lagrange theorem allows for the replacement of a large secret key d with two smaller secret keys, $d_p = d \mod (p-1)$ and $d_q = d \mod (q-1)$, for the computation in $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively. To use the CRT-decryption Algorithm 1, the secret key is (d, p, q) instead of just d.

CRT decryption, in theory, is cheaper than normal decryption by a factor of about 25 percent, and as it handles data with half the RSA modulus size, RSA with CRT is theoretically about four times faster and is therefore better suited to embedded devices [20]. Additionally, CRT decryption can be parallelized for faster decryption. It's important to note that Algorithm 1 can still be used even when $d_p = d_q = d$; however, we cannot use CRT decryption in such a scenario.

Algorithm 1 CRT-decryption

1: **Input:** d_p, d_q, and $C \in \mathbb{Z}_N$
2: **Output:** m
3: $C_p \leftarrow C^{d_p} \mod p$
4: $C_q \leftarrow C^{d_q} \mod q$
5: $m' \leftarrow (C_p - C_q) \cdot q^{-1} \mod p$
6: $m \leftarrow m' \cdot q + C_q$
7: **return** m

In the rest of this work, we assume that the user does not know the factors of N in the CRT model. This implies we treat Algorithm 1 as a black box. The user will only import the secret keys and ciphertext of the RSA encryption scheme into this algorithm and receive the message m without knowing the intermediate values such as p and q. Also, inspired by [21], the following lemma holds.

Lemma 1.1. *Let (e_1, d_1) and (e_2, d_2) be two distinct RSA public-private key pairs sharing a common modulus $N = pq$, with e_1 and e_2 chosen such that $g = \gcd(e_1 \cdot d_1 - 1, e_2 \cdot d_2 - 1)$. Suppose that there exist values $\alpha, \beta \in \mathbb{Z}$ such that $\alpha \equiv d_1 \mod \phi(N)$ and $\beta \equiv d_2 \mod \phi(N)$. Then the equation $\alpha \cdot e_2 + \beta \cdot e_1 \equiv 1 \pmod{\phi(N)}$ holds.*

Proof. Assume that $e_1 \cdot d_1 = 1 + k \cdot \phi(N)$ and $e_2 \cdot d_2 = 1 + j \cdot \phi(N)$ for some integers k, j. Expanding α and β using their definitions in terms of d_1 and d_2 for integers m, n:

$$\alpha = d_1 + m \cdot \phi(N) \text{ and } \beta = d_2 + n \cdot \phi(N),$$
$$\alpha \cdot e_2 + \beta \cdot e_1 = (d_1 + m \cdot \phi(N)) \cdot e_2 + (d_2 + n \cdot \phi(N)) \cdot e_1.$$

Since $e_1 \cdot d_1 \equiv 1 \pmod{\phi(N)}$ and $e_2 \cdot d_2 \equiv 1 \pmod{\phi(N)}$:

$$\alpha \cdot e_2 + \beta \cdot e_1 \equiv 1 \pmod{\phi(N)}.$$

□

1.4. Common Modulus Attacks

The original common modulus attack is a type of attack that assumes the existence of two RSA public keys, denoted by e and e', that share the same modulus N, where $\gcd(e, e') = 1$. With this assumption, an adversary can decrypt any encrypted message using both of these public keys [22].

The Common Modulus Attack is not applicable turns into a trivial attack in the CRT variant of RSA, since the key owner needs to know the values of p and q, in Algorithm 1. However, suppose that Algorithm 1 is executed as a black box, equivalently, the key owner does not know the factors of N. It is not possible to apply any kind of common modulus attacks to the RSA system that operates in this black box CRT decryption, since it is impossible to compute d only with d_p and d_q, and without knowing p and q, as $\gcd(p-1, q-1) \neq 1$. The attacker can only apply the above attacks by listing all possible p and q that satisfy $e.d_p - 1 = k(p-1)$ and $e.d_q - 1 = k'(q-1)$ and then checking with $N = pq$ to recover the correct solution. This technique requires factoring of at least one of $k(p-1)$ or $k'(q-1)$, which can be inefficient for large numbers, as the goal is to avoid factoring.

1.5. Cyclic Attack

Suppose we have an RSA ciphertext c for message m, where $c = m^e \mod (N)$. An attacker may try to re-encrypt the ciphertext c by computing $c^e \mod (N)$ and then repeating this process by computing $c^{e^i} \mod (N)$ in each iteration until an integer k is found such that $c^{e^k} = c \mod (N)$. Once k is found, the attacker can retrieve the original message m by computing $c^{e^{k-1}} \mod (N)$. In this case, the goal is to find k such that $\gcd(c^{e^k} - c, N) \neq 1$. To protect against cyclic attacks, the safest choice of system parameters is discussed in [23] and [24]. This attack was first presented in [25] and was later generalized in [26] and [24] to factor N. Note that this attack does not require factoring N and has shown that it can be applied to any Abelian finite group [27]. In [28], an algorithm has been proposed to compute $\phi(N)$ after finding k and e^k.

2. Related Works

In 2019, a survey of the attacks on RSA in the forty years of its life has been published [22]. In this survey, the attacks on RSA are classified into four groups: elementary attacks, weak public exponent attacks, weak private exponent attacks, and large private exponent attacks. Except for elementary attacks on RSA, all of the attacks originate from wrong choices in the public or private key. There are two elementary attacks on RSA: the common modulus attack and the blind signature attack [29]. The conventional common modulus attack assumes two coprime RSA public keys e_a and e_b in the same modulus N, i.e. $\gcd(e_a, e_b) = 1$. Then, the adversary can decrypt any message m encrypted by public keys e_a and e_b [22]. One can classify the attacks based on factorizing N given a public-private key pair [30, 31] as the common modulus attack, where the condition of $\gcd(e_a, e_b) = 1$ is not required anymore.

In the original paper of RSA, [1], it is clearly mentioned that given the public-private key pair (e, d), one can factorize N. This works off [32], which showed that N could be factored, given any multiple of $\phi(N)$ with the complexity of $O(log^3 N)$. Miller shows [32] how knowing the public-private key pair (e, d) is probabilistically equivalent to the factorization of N. Also, [31] presented the first deterministic polynomial time algorithm that factorizes N using the public-private key pair (e, d) with the complexities of $O(log^9 N)$ and $O(log^2 N)$, when $ed < N^2$ and $ed < N^{3/2}$, respectively. It is obvious that if the same modulus is used for Alice and Bob, then Bob can use his own exponents (e_b, d_b) to factorize the modulus N [30] and recover Alice's private key d_a given her public key e_a and the factorization of N.

The public key of RSA is usually set to a small number to provide fast encryption. Hastad presents [33] a small public exponent attack for a broadcast use case based on Coppersmith's method in [34]. In [35], a general version of the small public exponent was proposed. Although choosing a small exponent for the RSA private key provides a fast RSA signature, it makes the scheme vulnerable to small private exponent attacks. Wiener in [36] shows that the RSA system is insecure when $d < N^{0.25}$. However, this attack is useless if $e > N^{1.5}$. Boneh and Durfee in [37] improves Wiener's attack for a higher bound $d < N^{0.292}$. Their attack, which uses the LLL (Lenstra–Lenstra–Lovász) algorithm [38], is effective for $e < N^{\frac{15}{8}} = N^{1.875}$. Although they can not prove that it always succeeded, they have not found evidence of their attack's ineffectiveness [39]. Recently, a special case of Boneh and Durfee's attack is investigated by Mumtaz and Ping in [40], where the running time of the attack is improved. Mumtaz and Ping propose a large RSA decryption exponent attack in [41]. A cryptanalysis of the RSA cryptosystem with smooth prime sum is proposed in [42]. Moreover, [43] tries to inject backdoors into RSA and other cryptographic primitives based on the integer factoring problem.

3. Proposed Attack

Suppose two RSA public-private key pairs are generated using a common modulus for two different users. Then the following theorem shows that each user can compute during decryption a number equivalent to the private key of the other user, and hence can decrypt any ciphertext that was encrypted for the other user.

Theorem 3.1. *Let (e_1, d_1) and (e_2, d_2) be two public-private key pairs in a common modulus $N = pq$, and s_2 is the second Bézout coefficient for $gcd(\frac{e_1 d_1 - 1}{g}, e_2)$, i.e. $\frac{e_1 d_1 - 1}{g} \cdot s_1 + e_2 \cdot s_2 = 1$, where $g = \gcd(e_1 d_1 - 1, e_2)$. Then, $s_2 = d_2 \mod \phi(N)$, i.e. s_2 is equivalent to the private key d_2 in the decryption of RSA.*

Proof. Since $e_1 d_1 = 1 \mod \phi(N)$, so

$$e_1 d_1 - 1 = k \cdot \phi(N); \quad k \neq 0 \tag{1}$$

Based on (1), set $g = \gcd(e_1 d_1 - 1, e_2) = \gcd(k \cdot \phi(N), e_2)$. Since $\gcd(e_2, \phi(N)) = 1$, we have $g = \gcd(k, e_2)$, which means that $g|k$, and hence $k' = \frac{k}{g}$ is an integer. On the other hand, since $g = \gcd(e_1 d_1 - 1, e_2)$, it can be concluded that $\gcd(\frac{e_1 d_1 - 1}{g}, e_2) = 1$. Therefore, we have:

$$\begin{aligned} \gcd\left(\frac{e_1 d_1 - 1}{g}, e_2\right) &= \gcd\left(\frac{k \cdot \phi(N)}{g}, e_2\right) \\ &= \gcd\left(\frac{k}{g} \cdot \phi(N), e_2\right) \\ &= \gcd(k' \cdot \phi(N), e_2) = 1 \end{aligned} \tag{2}$$

Using the extended Euclidean algorithm, we can find Bézout coefficients s_1 and $s_2 \in \mathbb{Z}$ such that $k'\phi(N) \cdot s_1 + e_2 \cdot s_2 = 1$. This equation implies that,

$$e_2 \cdot s_2 = k'' \cdot \phi(N) + 1 \tag{3}$$

where $k'' = -k' \cdot s_1$ is an integer. Equation (3) shows that s_2 is the modular inverse of e_2 and is equivalent to d_2 in the decryption of ciphertexts encrypted by e_2. To verify the claim, assume we have the ciphertext C of a message m encrypted by e_2, written as $C = m^{e_2} \pmod N$. To decrypt C, all we need to do the following calculation:

$$C^{s_2} = (m^{e_2})^{s_2} = m^{k'' \phi(N) + 1} = m \mod N \tag{4}$$

So, we can use s_2 for decryption instead of d_2. □

Algorithm 2 presents a summary of our attack. According to Theorem 3.1, this algorithm is guaranteed to succeed. It is evident that the algorithm in Algorithm 2 is fast, as it only involves one multiplication, one gcd computation, one division, and finally, one Bézout algorithm.

Algorithm 2 The Proposed Common Modulus Attack

1: **Input:** e_1, d_1, and e_2
2: **Output:** s_2
3: $a \leftarrow e_1 \cdot d_1 - 1$
4: $g \leftarrow \gcd(a, e_2)$
5: $a \leftarrow a/g$
6: Find s_1 and s_2 such that $a \cdot s_1 + e_2 \cdot s_2 = 1$ ▷ Since a and e_2 are co-prime
7: **return** s_2

Remark 1. *The complexity of the attack based on Theorem (3.1) is $O(logN)$ since it requires two (extended) Euclidean Algorithm runs. It has no extra condition on the key pairs, the common modulus, or the ciphertext to be decrypted.*

Remark 2. *The proposed attack is valid also when (e_1, d_1) are in Euler RSA system and (e_2, d_2) are Carmichael or Euler. If both have Euler secret keys, this is already proved in Theorems 3 and 3.1. To prove the second type, suppose that $\phi(N) = t.\lambda(N)$. Set $g = \gcd(e_1 d_1 - 1, e_2) = \gcd(k.\phi(N), e_2) = \gcd(k.t.\lambda(N), e_2)$. Note that $\gcd(e_2, \lambda(N)) = 1$. The rest of the proof is similar to Theorem 3.1 by replacing k with $k.t$. To find s_2 in these types, Algorithm 2 is run, too.*

Remark 3. *We prove that the proposed attack is valid when (e_1, d_1) are in Carmichael's RSA system and (e_2, d_2) are either Carmichael or Euler. The following theorems prove this:*

Theorem 3.2. *The attack is valid if (e_1, d_1) are in Carmichael RSA system and (e_2, d_2) are either in Carmichael or Euler.*

Proof. If both parties have Carmichael's secret keys, it can be demonstrated easily by substituting $\lambda(N)$ for $\phi(N)$ in equations (1), (2), and (3). To prove the second type, it should be noted that $\gcd(e_2, \lambda(N)) = 1$ because $\gcd(e_2, \phi(N)) = 1$ and $\lambda(N)$ divides $\phi(N)$. The remainder of the proof is similar to the previous one. To determine s_2 in these instances, Algorithm 2 is applied. □

Corollary 3.2.1. *The attack proposed in Theorem 3.2 can be applied to the RSA system without considering Euler or Carmichael using Theorems of 3.2 and 3.3.*

Theorem 3.3. *Let (e_1, d_{1p}, d_{1q}) and (e_2, d_{2p}, d_{2q}) be two public-private key pairs for CRT decryption in a common modulus $N = pq$. Then, Algorithm 2 is valid to compute $s_{2p} = d_{2p} \bmod (p-1)$ and $s_{2q} = d_{2q} \bmod (q-1)$, i.e. s_{2p} and s_{2q} are equivalent to the private keys d_{2p} and d_{2q} respectively in the CRT decryption of RSA.*

Proof. In this case, the Algorithm 2 is run twice with inputs of the first (e_1, d_{1p}, d_{1q}) and then (e_2, d_{2p}, d_{2q}). In the case of (e_1, d_{1p}, e_2), we have $e_1 d_{1p} - 1 = k \cdot (p-1)$. Set $g = \gcd(e_1.d_{1p} - 1, e_2) = \gcd(k \cdot (p-1), e_2)$. Since $\gcd(e_2, \phi(N)) = 1$ and $(p-1) | \phi(N)$ then $\gcd(e_2, p-1) = 1$. Similar to the discussion in the proof of Theorem 3.1, we can write

$$\begin{aligned} \gcd\left(\frac{e_1 d_{1p} - 1}{g}, e_2\right) &= \gcd\left(\frac{k \cdot (p-1)}{g}, e_2\right) \\ &= \gcd\left(\frac{k}{g} \cdot (p-1), e_2\right) \\ &= \gcd(k' \cdot (p-1), e_2) = 1 \end{aligned} \quad (5)$$

So, there are s_{1p} and s_{2p} such that

$$\begin{aligned} & k'(p-1) \cdot s_{1p} + e_2 \cdot s_{2p} = 1 \\ \implies & e_2 \cdot s_{2p} = k'' \cdot (p-1) + 1 \\ \implies & s_{2p} = d_{2p} \mod (p-1) \end{aligned} \quad (6)$$

The proof for d_{2q} is the same as for the previous case. □

Remark 4. *If an attacker has the secret keys of RSA in the CRT model (without knowing the factors), then they can attack other users that use the same module. This becomes from the fact that if $\gcd(\phi(N), e_2) = 1$ or $\gcd(\lambda(N), e_2) = 1$ then we can conclude that $\gcd(p-1, e_2) = \gcd(q-1, e_2) = 1$. On the other side, we know that $\gcd(d_{p1}, p-1) = \gcd(d_{q1}, q-1) = 1$ is held. Therefore, we can write $\gcd(e_2.d_{p1}, p-1) = \gcd(e_2.d_{q1}, q-1) = 1$, $e_2.d_{p1} = 1 \mod (p-1)$ and $e_2.d_{q1} = 1 \mod (q-1)$. So, the attacker can use Algorithm 1 to recover message m.*

Corollary 3.3.1. *Based on Corollary 3.2.1 and Remark 4, if an attacker has a public-private key pair in any of the Euler, Carmichael, and black-boxed CRT RSA models, they can attack the ciphertexts of other users in a common modulus in any of the Euler, Carmichael, and black-boxed RSA models.*

4. Generalization and Factoring Algorithm

In this section, we present a generalized version of the presented attack. Unlike before, the attacker no longer needs to be the owner of the (e_1, d_1) key pair in modulus N. Instead, the attacker only needs to have leaked information of the form (α, β), where $f(\alpha) = \beta \mod (\phi(N))$ holds under the condition $k \neq 0$ in $f(\alpha) = \beta + k.\phi(N)$. Here, f can be any function composed of multiplication, division, summation, extraction, or exponentiation.

In the our attack case, assume that $\alpha = d_1$, $\beta = 1$, and $f(x) = e_1 \cdot x \mod (\phi(N))$. Then we have $f(\alpha) = \beta = 1$. For the cyclic attack, we have $\alpha = \beta = e$ and $f(x) = x^k$ such that $f(\alpha) = \beta$. By computing $f(\alpha) - \beta = k \cdot \phi(N)$, the attacker can apply the proposed attack in Theorem 3.1. This yields $\frac{f(\alpha)-\beta}{k} = \phi(N)$ and $g = \gcd(f(\alpha) - \beta, e_2)$. The remaining steps of the attack are identical to those outlined in Theorem 3.1. To apply this generalized attack, follow the Algorithm 3. In the generalized version of Theorem 3, suppose that the attacker knows $f(\alpha) = \beta \mod (p-1)$ and $g(\alpha') = \beta' \mod (q-1)$. In this case, Algorithm 3 is run twice to return s_{2_p} and s_{2_q}.

Now, we propose an algorithm for breaking RSA encryption, which is inspired by Algorithm 3 and can be executed without prior knowledge of the RSA secret key. To do this, we can run the Algorithm 4 with only

TABLE 1. COMPARISON OF THE COMMON MODULUS ATTACKS ON RSA

Attack Aim	Data required	Success Rate	Conditions	Complexity	Reference
Factoring N	$((e_1, d_1), N)$	Probabilistic $(1/2)$	-	$O(log^3 N)$	[32]
Factoring N	$((e_1, d_1), N)$	Deterministic	$e_1 d_1 < N^2$	$O(log^9 N)$	[31]
Factoring N	$((e_1, d_1), N)$	Deterministic	$e_1 d_1 < N^{3/2}$	$O(log^2 N)$	[31]
Obtaining Plaintext[1]	$(e_1, e_2, m^{e_1}, m^{e_2}, N)$	Deterministic	$\gcd(e_1, e_2) = 1$	$O(logN)$	[30]
Obtaining Plaintext	$((e_1, d_1), e_2, N)$	Deterministic	-	$O(logN)$	Ours
Obtaining Plaintext	$((e_1, d_{p1}, d_{q1}), e_2, N)$	Deterministic	-	$O(logN)$	Ours
Obtaining Plaintext	$(e, N), f(\alpha)$	Deterministic	-	$O(logN)$	Ours

the knowledge of the public key N to find the function $f(x)$. Once we have found the function f, we can apply Algorithm 3's algorithm to break RSA encryption successfully. Line 11 in Algorithm 4 guarantees $k \neq 0$ in relation $\beta = a \cdot \alpha^k + k \cdot \lambda(N)$.

In the worst case, the algorithm's running time is equivalent to the cyclic attack if we set $\alpha = e$ and $f(x) = x^k$. However, unlike a cyclic attack, Algorithm 4 can be used for factoring in cases beyond RSA. This means that Algorithm 4 does not need the value of e, which can be an added advantage. Moreover, the Algorithm 4 can run be parallelized for multiple α and β, a noteworthy feature.

Algorithm 3 Generalization of the Proposed Attack

1: **Input:** α, β, $f(x)$ and e_2
2: **Output:** s_2
3: $b \leftarrow f(\beta)$
4: $a \leftarrow \alpha \cdot b - 1$
5: $g \leftarrow \gcd(a, e_2)$
6: $a \leftarrow a/g$
7: Find s_1 and s_2 such that $a \cdot s_1 + e_2 \cdot s_2 = 1$ ▷ Since a and e_2 are co-prime
8: **return** s_2

Algorithm 4 Finding Function f

1: **Input:** N
2: **Output:** α, β, and $f(x)$
3: $k \leftarrow 1, m \leftarrow 2$ ▷ Or any $m \in \mathbb{Z}_N$. The function f is independent of m
4: $\alpha, \beta, a \leftarrow$ random elements from $\mathbb{Z}_N$
5: $c_1 \leftarrow m^{a \cdot \alpha} \mod N$
6: $c_2 \leftarrow m^{\beta} \mod N$
7: **while** $c_1 \neq c_2$ **do**
8: $\quad k \leftarrow k + 1$
9: $\quad c_1 \leftarrow c_1^{\alpha} \mod N$
10: **end while**
11: **if** $\beta = a \cdot \alpha^k$ **then**
12: $\quad k \leftarrow k + 1$
13: $\quad c_1 \leftarrow c_1^{\alpha} \mod N$
14: $\quad$ **goto** line 8
15: **end if**
16: **return** $\alpha, \beta, f(x) = a \cdot x^k$

To factorize N, we can use the algorithm's function. By computing $\gcd(m^{f(\alpha)} - m^{\beta}, N)$ using this function, we can find the factors of N. Another way to factorize N using the Algorithm 4 is to compute $f(\alpha) - \beta = k\lambda(N)$, where $k \neq 0$, and then run the Miller algorithm to recover factors of N. The complexity order of Algorithm 4's algorithm is equivalent to the order of cyclic attack in the worst case, as the cyclic attack is a special case of ours. While the complexity order of the cyclic attack on the RSA cryptosystem is not explicitly stated in the provided search results, we can say that our factoring attack can achieve a complexity order of less than the cyclic attack's complexity order.

5. Conclusion

We introduced a new method of attack on the RSA cryptosystem, which is a type of common modulus attack. This attack involves using one of the public-private key pairs to obtain the other private key in the decryption process. Our proposed attack can work with both variants of RSA, namely $\phi(N)$ and $\lambda(N)$, and has a complexity of $O(\log N)$. Two types of common modulus attacks have been previously identified. The first type assumes that the two public keys are coprime, enabling the attacker to decrypt a message encrypted by both public keys. The second type, proposed in our attack, is when the attacker possesses one of the public-private key pairs and can factorize the modulus under specific conditions outlined in Table 1. Our proposed attack has several advantages, including the ability to be applied to the black-boxed CRT mode of the RSA cryptosystem. Additionally, the attack can be generalized to the known function in $\phi(N)$ or $\lambda(N)$, which can be a subset of a side-channel attack. Finally, we outline an algorithm for factoring inspired by the proposed attack. In the worst case, the order of complexity of this algorithm is equal to the cyclic attack.

References

[1] Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[2] Élie Gouzien and Nicolas Sangouard. "Factoring 2048-bit RSA integers in 177 days with 13 436 qubits and a multimode memory". In: *Physical review letters* 127.14 (2021), p. 140503.

[3] Yansong Feng, Abderrahmane Nitaj, and Yanbin Pan. "Small Public Exponent Brings More: Improved Partial Key Exposure Attacks against RSA". In: *IACR Communications in Cryptology* 1.3 (Oct. 7, 2024). ISSN: 3006-5496. DOI: 10.62056/ahjbhey6b.

[4] Sathi Sarveswara Reddy, Sharad Sinha, and Wei Zhang. "Design and Analysis of RSA and Paillier Homomorphic Cryptosystems Using PSO-Based Evolutionary Computation". In: *IEEE Transactions on Computers* (2023), pp. 1–14.

[5] Hung-Min Sun et al. "Dual RSA and its security analysis". In: *IEEE Transactions on Information Theory* 53.8 (2007), pp. 2922–2933.

[6] *Generate Keys and Certificates for SSO - Google Workspace Admin Help — support.google.com*. https://support.google.com/a/answer/6342198?hl=en. [Accessed 21-05-2024].

[7] *FAQ for the Technically Inclined — core.telegram.org*. https://core.telegram.org/techfaq. [Accessed 21-05-2024].

[8] *Hardening OpenVPN Security | OpenVPN — openvpn.net*. https://openvpn.net/community-resources/hardening-openvpn-security. [Accessed 21-05-2024].

[9] Ben Harris. *RFC 4432: RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc4432. [Accessed 25-10-2024].

[10] William Stallings. *RFC 1991: PGP Message Exchange Formats — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc1991\#section-6.2.3. [Accessed 25-10-2024].

[11] Abel Yeboah-Ofori, Iman Darvishi, and Azeez Sakirudeen Opeyemi. "Enhancement of Big Data Security in Cloud Computing Using RSA Algorithm". In: *2023 10th International Conference on Future Internet of Things and Cloud (FiCloud)* (2023), pp. 312–319. URL: https://api.semanticscholar.org/CorpusID:267328168.

[12] *Cryptographic algorithms - AWS cryptography services — docs.aws.amazon.com*. https://docs.aws.amazon.com/en_us/crypto/latest/userguide/concepts-algorithms.html. [Accessed 25-10-2024].

[13] *AWS History and Timeline regarding AWS Key Management Service - Overview, Functions, Features, Summary of Updates, and Introduction to KMS | hidekazu-konishi.com — hidekazu-konishi.com*. https://hidekazu-konishi.com/entry/aws_history_and_timeline_aws_kms.html. [Accessed 25-10-2024].

[14] *How to use AWS KMS RSA keys for offline encryption | Amazon Web Services — aws.amazon.com*. https://aws.amazon.com/blogs/security/how-to-use-aws-kms-rsa-keys-for-offline-encryption/. [Accessed 25-10-2024].

[15] V. Anusuya Devi and T. Sampradeepraj. "End-to-End Self-organizing Intelligent Security Model for Wireless Sensor Network based on a Hybrid (AES–RSA) Cryptography". In: *Wireless Personal Communications* 136.3 (June 2024), pp. 1675–1703. ISSN: 1572-834X. DOI: 10.1007/s11277-024-11353-3. URL: http://dx.doi.org/10.1007/s11277-024-11353-3.

[16] N. Premkumar and R. Santhosh. "Pelican optimization algorithm with blockchain for secure load balancing in fog computing". In: *Multimedia Tools and Applications* 83.18 (Nov. 2023), pp. 53417–53439. ISSN: 1573-7721. DOI: 10.1007/s11042-023-17632-8. URL: http://dx.doi.org/10.1007/s11042-023-17632-8.

[17] Vassil Dimitrov, Luigi Vigneri, and Vidal Attias. "Fast Generation of RSA Keys using Smooth Integers". In: *IEEE Transactions on Computers* (2021).

[18] John Friedlander, Carl Pomerance, and Igor Shparlinski. "Period of the power generator and small values of Carmichael's function". In: *Mathematics of computation* 70.236 (2001), pp. 1591–1605.

[19] J-J Quisquater and Chantal Couvreur. "Fast decipherment algorithm for RSA public-key cryptosystem". In: *Electronics letters* 21.18 (1982), pp. 905–907.

[20] David Vigilant. "RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks". In: *Cryptographic Hardware and Embedded Systems – CHES 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 130–145. ISBN: 978-3-540-85053-3.

[21] Daniel Bleichenbacher. "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1". In: *Advances in Cryptology — CRYPTO '98*. Ed. by Hugo Krawczyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 1–12. ISBN: 978-3-540-68462-6.

[22] Majid Mumtaz and Luo Ping. "Forty years of attacks on the RSA cryptosystem: A brief survey". In: *Journal of Discrete Mathematical Sciences and Cryptography* 22.1 (2019), pp. 9–29.

[23] Shimshon Berkovits. "Factoring via superencyrption". In: *Cryptologia* 6.3 (1982), pp. 229–237.

[24] Peter Jamnig. "Securing the RSA-cryptosystem against cycling attacks". In: *Cryptologia* 12.3 (1988), pp. 159–164.

[25] Gustavus J Simmons and Michael J Norris. "Preliminary comments on the MIT public-key cryptosystem". In: *Cryptologia* 1.4 (1977), pp. 406–414.

[26] Marc Gysin and Jennifer Seberry. "Generalised cycling attacks on RSA and strong RSA primes". In: *Information Security and Privacy: ACISP*. Vol. 99. 1999, pp. 149–163.

[27] Dép d'Électricité. "How to choose secret parameters for RSA and its extensions to elliptic curves". In: *Designs, Codes and Cryptography* 23.3 (2001), pp. 297–316.

[28] Xiaoying Ye, Chenglian Liu, and Donald Gardner. "Weakness of RSA cryptosystem characteristic". In: *AIP Conference Proceedings*. Vol. 2040. 1. AIP Publishing LLC. 2018, p. 130005.

[29] David Chaum. "Blind signature system". In: *Advances in cryptology*. Springer. 1984, pp. 153–153.

[30] Dan Boneh et al. "Twenty years of attacks on the RSA cryptosystem". In: *Notices of the AMS* 46.2 (1999), pp. 203–213.

[31] Jean-Sébastien Coron and Alexander May. "Deterministic polynomial-time equivalence of computing the RSA secret key and factoring". In: *Journal of Cryptology* 20.1 (2007), pp. 39–50.

[32] Gary L Miller. "Riemann's hypothesis and tests for primality". In: *Journal of computer and system sciences* 13.3 (1976), pp. 300–317.

[33] Johan Hastad. "Solving simultaneous modular equations of low degree". In: *siam Journal on Computing* 17.2 (1988), pp. 336–341.

[34] Don Coppersmith. "Small solutions to polynomial equations, and low exponent RSA vulnerabilities". In: *Journal of cryptology* 10.4 (1997), pp. 233–260.

[35] Mengce Zheng, Honggang Hu, and Zilong Wang. "Generalized cryptanalysis of RSA with small public exponent". In: *Science China Information Sciences* 59.3 (2016), pp. 1–10.

[36] Michael J Wiener. "Cryptanalysis of short RSA secret exponents". In: *IEEE Transactions on Information theory* 36.3 (1990), pp. 553–558.

[37] Dan Boneh and Glenn Durfee. "Cryptanalysis of RSA with private key d less than N/sup 0.292". In: *IEEE transactions on Information Theory* 46.4 (2000), pp. 1339–1349.

[38] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische annalen* 261.ARTICLE (1982), pp. 515–534.

[39] Dan Boneh and Glenn Durfee. "New results on the cryptanalysis of low exponent RSA". In: *IEEE Transactions on Information Theory* 46.4 (2000), pp. 1339–1349.

[40] Majid Mumtaz and Luo Ping. "Cryptanalysis of a Special Case of RSA Large Decryption Exponent Using Lattice Basis Reduction Method". In: *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*. IEEE. 2021, pp. 714–720.

[41] Majid Mumtaz and Luo Ping. "An improved cryptanalysis of large RSA decryption exponent with constrained secret key". In: *International Journal of Information and Computer Security* 14.2 (2021), pp. 102–117.

[42] Meryem Cherkaoui Semmouni, Abderrahmane Nitaj, and Mostafa Belkasmi. "Cryptanalysis of RSA with smooth prime sum". In: *Journal of Discrete Mathematical Sciences and Cryptography* (2022), pp. 1–21.

[43] Marco Cesati. "A new idea for RSA backdoors". In: *arXiv preprint arXiv:2201.13153* (2022).