mathematics

MDPI

Article

Revisiting the Polynomial-Time Equivalence of Computing the CRT-RSA Secret Key and Factoring

Mengce Zheng [1,2]

[1] College of Information and Intelligence Engineering, Zhejiang Wanli University, Ningbo 315100, China; mczheng@zwu.edu.cn
[2] School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China

Abstract: The Rivest–Shamir–Adleman (RSA) cryptosystem is currently the most influential and commonly used algorithm in public-key cryptography. Whether the security of RSA is equivalent to the intractability of the integer factorization problem is an interesting issue in mathematics and cryptography. Coron and May solved the above most fundamental problem and proved the polynomial-time equivalence of computing the RSA secret key and factoring. They demonstrated that the RSA modulus $N = pq$ can be factored in polynomial time when given RSA key information (N, e, d). The CRT-RSA variant is a fast technical implementation of RSA using the Chinese Remainder Theorem (CRT), which aims to speed up the decryption process. We focus on the polynomial-time equivalence of computing the CRT-RSA secret key and factoring in this paper. With the help of the latest partial key exposure attack on CRT-RSA, we demonstrate that there exists a polynomial-time algorithm outputting the factorization of $N = pq$ for $ed_p, ed_q < N^{3/2}$ when given the CRT-RSA key information (N, e, d_p, d_q). We apply Coppersmith's lattice-based method as a basic mathematical tool for finding the small root solutions of modular polynomial equations. Furthermore, we provide validation experiments to illustrate the correctness of the CRT-RSA modulus factorization algorithm, and show that computing the CRT-RSA secret key and factoring its modulus is polynomial-time equivalent by using concrete numerical examples.

Keywords: CRT-RSA; cryptanalysis; factorization; lattice-based method; polynomial-time equivalence

MSC: 94A60

Citation: Zheng, M. Revisiting the Polynomial-Time Equivalence of Computing the CRT-RSA Secret Key and Factoring. *Mathematics* **2022**, *10*, 2238. https://doi.org/10.3390/math10132238

Academic Editor: Angel Martín-del-Rey

Received: 21 May 2022
Accepted: 24 June 2022
Published: 26 June 2022

1. Introduction

Rivest, Shamir, and Adleman proposed the first practical public-key cryptosystem named the RSA algorithm [1]. This algorithm just involves the elementary number-theoretic operations and is easy to compute and implement. Thus, RSA is widely studied in the field of information security and serves as an influential public-key cryptographic algorithm utilized by international standardization organizations. The RSA cryptosystem consists of a key generation algorithm, an encryption algorithm, and a decryption algorithm, which are described as follows.

Key Generation. Randomly select two prime numbers p, q of the same bit-size and compute the modulus $N = pq$ along with its Euler's totient function $\varphi(N) = (p-1)(q-1)$. Randomly select $0 < e < \varphi(N)$ satisfying $\gcd(e, \varphi(N)) = 1$ as the public key and compute $d \equiv e^{-1} \bmod \varphi(N)$ as the private key.

Encryption. Alice transforms a plaintext message into $m \in \mathbb{Z}_N$ and computes $c \equiv m^e \bmod N$ as the corresponding ciphertext.

Decryption. Bob receives a ciphertext c and calculates $c^d \bmod N$. Bob extracts the transformed m and recovers the original plaintext message.

The correctness of the encryption and decryption algorithms is ensured by Euler's Totient Theorem. Since both the public key e and the private key d are calculated in the exponent position, they are mentioned as the public and private exponents, respectively.

The hardness assumption of the RSA algorithm is the intractability of the integer factorization problem [2]. There is still no known polynomial-time algorithm that can factor sufficiently large integers using current classical computers. So there is no known effective attack against the RSA algorithm. However, on the other hand, RSA with small private exponents have been proven to be insecure when using Wiener's continued fraction-based attack [3] and Coppersmith's lattice-based attack [4]. A series of cryptanalytic researches on RSA and its variants were subsequently developed based on the lattice-based method, see the literature [5–12]. One of these works related to the hardness assumption of RSA is the polynomial-time equivalence proof of computing the RSA secret key and factoring [8,13]. The main contribution is applying the lattice-based method to prove that there exists an algorithm that can output the factorization of the modulus $N = pq$ in polynomial time when given (N, e, d).

In fact, in the public key cryptography standard PKCS #1 [14], the CRT-RSA algorithm is a fast technical implementation of RSA. Here CRT-RSA refers to the RSA variant using the Chinese Remainder Theorem, which was proposed in [15]. The Chinese Remainder Theorem formulates the solution to simultaneous linear congruences and is specified as follows.

Theorem 1. *Let $m_1, m_2, \ldots, m_k$ be mutually coprime positive integers, and let $a_1, \ldots, a_k$ be integers. Then the following system of linear congruences*

$$x \equiv a_i \bmod m_i$$

has a unique solution in the sense of modulo $M = \prod_{i=1}^{k} m_i$. Let $M_i = M/m_i$ for $i = 1, \ldots, k$, then the unique solution is

$$x = \sum_{i=1}^{k} a_i M_i (M_i^{-1} \bmod m_i) \bmod M.$$

Since the RSA modulus $N = pq$ is the product of two prime numbers, we consider the case of $k = 2$, i.e., the system of linear congruences is $x \equiv a_1 \bmod p$ and $x \equiv a_2 \bmod q$. In this case, the solution expression is more concise.

$$x = (a_1 + ((a_2 - a_1) \times (p^{-1} \bmod q) \bmod q) \times p) \bmod pq. \tag{1}$$

For a ciphertext $c = m^e \bmod N$, the decryption algorithm works in the following strategy. One first does a partial decryption operation to recover $m_p = c^d \bmod p$, $m_q = c^d \bmod q$, and then use the Chinese Remainder Theorem to combine the above two parts to recover m. Since p and q are coprime, one applies the Formula (1) that yields

$$m = (m_p + ((m_q - m_p) \times (p^{-1} \bmod q) \bmod q) \times p) \bmod pq.$$

It can be further simplified as

$$m = (m_p + p((m_q - m_p)p^{-1} \bmod q)) \bmod N. \tag{2}$$

Let $d_p \equiv e^{-1} \bmod (p-1)$ and $d_q \equiv e^{-1} \bmod (q-1)$, by Fermat's Little Theorem we have

$$m_p = c^{d_p} \equiv c^d \bmod p, \; m_q = c^{d_q} \equiv c^d \bmod q.$$

Thus, d_p, d_q are called CRT exponents and used as private exponents instead of d. The above Formula (2) is known as Garner's algorithm [16]. Theoretically speaking, the decryption efficiency of the modular operations can be accelerated by a factor of four due to the

reduction of the modulus from N to p or q. We summarize the CRT-RSA cryptosystem as follows.

Key Generation. Randomly select two prime numbers p, q of the same bit-size and compute the modulus $N = pq$ and its Euler's totient function $\varphi(N) = (p-1)(q-1)$. Randomly select $0 < e < \varphi(N)$ satisfying $\gcd(e, \varphi(N)) = 1$ as the public key and compute $d_p \equiv e^{-1} \bmod (p-1)$, $d_q \equiv e^{-1} \bmod (q-1)$ as the private key.

Encryption. Alice transforms a plaintext message into $m \in \mathbb{Z}_N$ and computes $c \equiv m^e \bmod N$ as the corresponding ciphertext.

Decryption. Bob receives a ciphertext c and calculates $m_p = c^{d_p} \bmod p$, $m_q = c^{d_q} \bmod q$. Bob extracts $m = (m_p + p((m_q - m_p)p^{-1} \bmod q)) \bmod N$ and recovers the original plaintext message.

Cryptanalysis of the CRT-RSA algorithm also attracts many researchers. The small CRT-exponent attack was originally presented as an open problem in Wiener's attack [3]. The first cryptanalytic result was given in [6], which was effective for unbalanced primes p and q. Subsequently, Jochemsz and May [17] proposed the small CRT-exponent attack for balanced primes when the attack range is d_p, $d_q < N^{0.073}$. It was further improved to d_p, $d_q < N^{0.091}$ in [12] and shortly afterwards to d_p, $d_q < N^{0.122}$ [18]. In addition, the partial-key-exposure attacks such as [19–23] were studied due to the consideration of partial leakage of the CRT-RSA private key. From the implementation aspect, side-channel attacks such as [24–28] were proposed by exploiting the side-channel information leakage during the running process of the CRT-RSA algorithm.

Unlike the above attacks, we focus on the intrinsic security of CRT-RSA, i.e., its mathematical hardness assumption. From the theoretical research aspect, it is interesting to investigate the polynomial-time equivalence of computing the CRT-RSA secret key and factoring its modulus. Therefore, the research problem is stated as follows.

> If the CRT-RSA key information (N, e, d_p, d_q) is given as input, does there exist a polynomial-time algorithm that can output the factorization of N? Furthermore, is it possible that the CRT-exponents reach the natural constraint $d_p, d_q < N^{1/2}$ for $e \approx N$?

It is well known that d_p and d_q can be efficiently inferred from the factorization of $N = pq$ and the public exponent e. Hence, it suffices to show that the above problem is solvable to conclude that computing the CRT-RSA secret key and factoring the modulus are polynomial-time equivalent. Our main goal is to demonstrate and design a polynomial-time algorithm outputting the factorization of N for given CRT-RSA key information (N, e, d_p, d_q). The specific goal is to further enlarge the attack range of CRT exponents d_p, d_q. We study how to use improved analysis techniques to carry out attacks on CRT-RSA from the perspective of its intrinsic security, namely the polynomial-time equivalence of computing the CRT-RSA secret key and factoring its modulus.

Inspired by the latest partial key exposure attack on CRT-RSA [23], we show an optimized polynomial-time equivalence of computing the CRT-RSA secret key and factoring. To be specific, we give a positive answer to the above research problem based on the partial key exposure attack on CRT-RSA. A polynomial-time factoring algorithm is presented for the given CRT-RSA key information (N, e, d_p, d_q), which outputs the factorization result of $N = pq$. Technically speaking, the factoring algorithm involves an error term $\gcd(N-1, 2^t e)$ for $t := \lceil \log_2(\max(d_p, d_q)) \rceil$. This term may equal to $O(1)$, which is negligible compared to e, d_p, d_q in the most favorable case, and asymptotically leads to the natural constraint $d_p, d_q < N^{1/2}$ for $e \approx N$. Moreover, we provide validation experiments in the form of numerical examples.

Our contribution mainly includes a new factoring algorithm and an improved attack result, which are summarized as follows.

- We study the security of CRT-RSA concerning its mathematical hardness assumption and propose an improved polynomial-time algorithm of factoring the CRT-RSA modulus.

- We verify the correctness and validity of the proposed CRT-RSA modulus factorization algorithm with numerical computer experiments and always successfully obtain the factorization results.
- We further discuss the advantages and disadvantages of our approach with several comparisons to properly assess the proposed factoring attack.

The rest of this paper is organized as follows. Section 2 introduces the lattice-based method, which is used as a basic mathematical tool to solve the research problem. Section 3 first reviews the latest partial key exposure attack on CRT-RSA and gives a refined theorem with more flexible parameters. Then we propose an improved polynomial-time equivalence of computing the CRT-RSA secret key and factoring. Both the CRT-RSA key computation algorithm and the CRT-RSA modulus factorization algorithm are presented. Finally, the validation experiments are provided in the form of numerical examples. We compare our results with previous studies and discuss the advantages and disadvantages of the proposed algorithms in Section 4. Section 5 concludes this paper.

2. Materials and Methods

We introduce the lattice-based method, which consists of the lattice reduction algorithm [29] and Coppersmith's techniques [4]. Generally speaking, a lattice is a discrete additive subgroup in $\mathbb{R}^n$ that is defined as follows.

Definition 1. *Let $1 \leq \omega \leq n$ and $\vec{b}_1, \ldots, \vec{b}_\omega \in \mathbb{R}^n$ be a set of linearly independent vectors. The lattice $\mathcal{L}$ generated by $\vec{b}_1, \ldots, \vec{b}_\omega$ is the set containing all linear combinations of the vectors with integer coefficients.*

$$\mathcal{L} := \mathbb{Z}\vec{b}_1 + \cdots + \mathbb{Z}\vec{b}_\omega = \left\{ \sum_{i=1}^{\omega} z_i \vec{b}_i : z_i \in \mathbb{Z} \right\}.$$

The vector set $\vec{b}_1, \ldots, \vec{b}_\omega$ is called a basis of the lattice $\mathcal{L}$. Further, n is the lattice dimension, ω is the lattice rank, and $\mathcal{L}$ is full-rank if $\omega = n$. For $i = 1, \ldots, \omega$, each basis vector can be written as $\vec{b}_i = (b_{i1}, \ldots, b_{in})$ in the form of row vectors, thus forming a basis matrix $\mathbf{B} = (b_{ij})_{\omega \times n}$.

Definition 2. *The fundamental parallelepiped of $\mathcal{L}$ is defined as $\mathcal{P}(\mathcal{L}) := \{\mathbf{B}\vec{x} : \vec{x} \in [0,1)^n\}$, where $\mathbf{B}$ is any basis matrix of the lattice $\mathcal{L}$. The lattice determinant is defined as the volume of the fundamental parallelepiped.*

$$\det(\mathcal{L}) := \sqrt{\det(\mathbf{B}\mathbf{B}^\mathsf{T})},$$

where $\mathbf{B}^\mathsf{T}$ is the transpose matrix of $\mathbf{B}$. Thus, it follows that the determinant of a full-rank lattice $\mathcal{L}$ is

$$\det(\mathcal{L}) = \sqrt{\det(\mathbf{B}\mathbf{B}^\mathsf{T})} = \sqrt{\det(\mathbf{B})\det(\mathbf{B}^\mathsf{T})} = |\det(\mathbf{B})|,$$

where $\det(\mathbf{B})$ is the determinant of the basis matrix $\mathbf{B}$.

The lattice determinant does not depend on a particular basis matrix since it is an invariant of the lattice itself. A lattice can usually be generated by many different basis matrices. More concretely, any two basis matrices of the same lattice can be converted to each other using a unimodular matrix. Unless otherwise specified, the lattices in this paper are full-rank.

The famous Lenstra–Lenstra–Lovász (LLL) lattice reduction algorithm was proposed in [29]. Though this algorithm cannot directly output the shortest lattice vector, it outputs the approximately shortest basis vectors in polynomial time. In other words, the length of the output vectors does not exceed a certain multiple of the shortest vector, and the LLL algorithm usually works better in practice. The works [29,30] show the following lemma for the upper bound estimation of the approximately shortest vectors output by the LLL algorithm.

Lemma 1 (LLL). *Let $\vec{v}_1, \ldots, \vec{v}_\omega$ be a set of basis vectors output by the LLL algorithm running on the lattice $\mathcal{L}$. We have*

$$|\vec{v}_1|, \ldots, |\vec{v}_i| \leq 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} (\det(\mathcal{L}))^{\frac{1}{\omega+1-i}},\ i = 1, \ldots, \omega.$$

Its time complexity is $O(\omega^6(\log B)^3)$ that is polynomial in the lattice dimension ω and the length of the input vectors B.

Coppersmith presented how to solve modular or integer polynomial equations based on the LLL algorithm [31,32]. The core idea is to construct solvable equations over the integers, which are represented as short basis vectors in the lattice. Meanwhile, the short basis vectors can be efficiently found by the LLL algorithm.

We describe the specific problem of solving modular polynomial equations as follows. Let R be a known positive integer, and let $v(x_1, \ldots, x_k)$ be a multivariate polynomial. The goal is to find small root solutions of $v(x_1, \ldots, x_k) \bmod R$, i.e., to solve $v(x'_1, \ldots, x'_k) \equiv 0 \bmod R$ under $|x'_i| \leq X_i$, where X_i is the upper bound on the absolute value of the variable x'_i related to the small root solution. It is required is to maximize the upper bound X_i on the unknown variables, while keeping the time complexity polynomial in the input parameters.

Before describing the strategy of solving modular polynomial equations, we define the polynomial norm. For a certain k-variable polynomial $v(x_1, \ldots, x_k) = \sum a_{i_1,\ldots,i_k} x_1^{i_1} \cdots x_k^{i_k}$, the non-zero coefficients $a_{i_1,\ldots,i_k}$ are related to the corresponding monomials $x_1^{i_1} \cdots x_k^{i_k}$. The polynomial norm is defined as follows.

Definition 3. *Let $v(x_1, \ldots, x_k) = \sum a_{i_1,\ldots,i_k} x_1^{i_1} \cdots x_k^{i_k}$ with $a_{i_1,\ldots,i_k} \neq 0$ be a k-variable polynomial. Its norm is defined as*

$$\|v(x_1, \ldots, x_k)\| := \left(\sum a_{i_1,\ldots,i_k}^2\right)^{1/2}.$$

Howgrave-Graham [33] further improved on Coppersmith's original techniques. The following lemma is used for determining whether the small root solution of a modular polynomial equation also holds on the integers, i.e., whether the modular condition can be eliminated.

Lemma 2 (Howgrave-Graham). *Let $v(x_1, \ldots, x_k) \in \mathbb{Z}[x_1, \ldots, x_k]$ be a k-variable integer polynomial containing ω monomials. When the following conditions are satisfied, $v(x'_1, \ldots, x'_k) = 0$ also holds over the integers.*

1. $v(x'_1, \ldots, x'_k) \equiv 0 \bmod R$, $|x'_1| \leq X_1, \ldots, |x'_k| \leq X_k$,
2. $\|v(X_1 x_1, \ldots, X_k x_k)\| < R/\sqrt{\omega}$.

The basic idea of solving k-variable polynomials is to construct $\ell \geq k$ algebraically with independent polynomials sharing the same small root over the integers. Hence, one can extract the final solution by the Gröbner basis computation [34]. By applying Lemma 2, the problem of solving modular polynomial equations can be turned to solving integer equations. Further, by combining Lemma 1, one can solve a given modular polynomial equation under certain conditions. The lattice-based method constructs a set of shift polynomials sharing the same root modulo R and then transforms the derived polynomials into integer equations in a specific way. Technically speaking, a lattice-basis matrix is generated from the coefficient vectors of the shift polynomials, which spans an ω-dimensional lattice. The LLL algorithm is applied to obtain the approximately shortest basis vectors, which are later transformed into polynomial equations. If the corresponding polynomial norm is small enough, then the derived equation also holds over the integers. We summarize the process of solving k-variable polynomial equations using the lattice-based method as follows.

1. The first step is to construct the set of shift polynomials $\mathcal{P}$ for given modular polynomials $f_1(x_1, \ldots, x_k), \ldots, f_n(x_1, \ldots, x_k)$ containing k unknown variables and a given modulus M. For $0 \leq i_1, \ldots, i_n \leq m \in \mathbb{N}$ and $j_1, \ldots, j_k \in \mathbb{N}$, the basic form of a shift polynomial is defined as
$$p_i(x_1, \ldots, x_k) := f_1^{i_1} \cdots f_n^{i_n} \cdot x_1^{j_1} \cdots x_k^{j_k} \cdot M^{m-(i_1+\cdots+i_n)}, \ 1 \leq i \leq \omega.$$
Thus, all polynomials in $\mathcal{P}$ have the same root $(x'_1, \ldots, x'_k)$ mod M^m, where the modulus is $R = M^m$.
2. The second step is to use $\vec{b}_i$ denoting the row vector transformed from the coefficient vectors of the shift polynomial $p_i(X_1x_1, \ldots, X_kx_k)$ when substituting X_ix_i for x_i. The lattice $\mathcal{L} = \left\{\sum_{i=1}^{\omega} z_i\vec{b}_i : z_i \in \mathbb{Z}\right\}$ is constructed by the lattice basis matrix $\mathbf{B} = (b_{ij})_{\omega\times\omega}$. Then apply the LLL algorithm to the lattice $\mathcal{L}$ and extract the first ℓ many approximately shortest vectors $\vec{v}_1, \ldots, \vec{v}_\ell$ from the output. The output vectors are transformed into a system $\mathcal{V}$ of integer equations $v_1(x_1, \ldots, x_k), \ldots, v_\ell(x_1, \ldots, x_k)$, whose roots also hold over the integers.
3. The third step is extract the desired small root solution. If the derived polynomials $v_i(x_1, \ldots, x_k)$ are mutually algebraically independent, the system of integer equations $\mathcal{V}$ can be solved by applying the Gröbner basis computation. At this point, the small root solution $(x'_1, \ldots, x'_k)$ of the original modular polynomials is obtained.

When obtaining the first ℓ number of basis vectors using the LLL algorithm, to ensure that the polynomial equations related to the basis vectors satisfy the solvable conditions, we have $2^{\frac{\omega(\omega-1)}{4(\omega+1-\ell)}} \det(\mathcal{L})^{\frac{1}{\omega+1-\ell}} < R/\sqrt{\omega}$. Since we always have $\ell \ll \omega \ll R$, the following simplified condition is obtained by omitting the lower error terms,

$$\det(\mathcal{L}) < R^{\omega}. \tag{3}$$

To make the lattice determinant $\det(\mathcal{L})$ easy to compute, it is generally required that the basis matrix $\mathbf{B}$ is of lower or upper triangular form. Therefore, during the construction of shift polynomials, it is necessary to ensure that each new shift polynomial p_i introduces exactly one new monomial λ_i. Hence, the lattice determinant is calculated by accumulating the monomials on the diagonal of the lattice basis matrix. We have $\det(\mathcal{L}) = |\det(\mathbf{B})| = \prod_{i=1}^{\omega} |\lambda_i(X_1, \ldots, X_k)|$.

The solution of multivariate polynomials using the lattice-based method is heuristic, so we note that its feasibility relies on the following heuristic assumption. Although the LLL algorithm guarantees that the reduced basis vectors are linearly independent, it cannot guarantee the algebraic independence of the transformed polynomials. On the other hand, similar to other cryptanalytic researches on RSA and its variants using the lattice-based method, the following assumption always holds in the validation experiments.

Assumption 1. *The system of integer equations related to the approximately shortest basis vectors output by the LLL algorithm can be efficiently solved. The small root solution can be extracted in polynomial time by the Gröbner basis computation.*

3. Results

We aim to propose a polynomial-time algorithm outputting the factorization of N for given CRT-RSA key information (N, e, d_p, d_q) by using the latest partial key exposure attack. To do so, we modify solving the factoring problem into conducting the partial key exposure attack. In other words, we extract two primes p and q for known LSB components of the CRT-exponents (i.e., the LSB components are exactly d_p, d_q themselves if let the MSB components be 0). We follow the lattice-based method and briefly review the partial key exposure attack on CRT-RSA.

The partial key exposure attack on CRT-RSA [23] implicitly shows the existence of a factoring algorithm, which takes given CRT-RSA key information (N, e, d_p, d_q) as

input and outputs the factorization of $N = pq$ for $d_p,\ d_q < N^{1/2}$ and $e \approx N$. Therefore, we explicitly design a polynomial-time algorithm that outputs the factorization of N for $d_p,\ d_q < N^{1/2}$ and $e \approx N$. From a mathematical standpoint, we simplify the complicated and redundant attack analysis in [23] and further present a refined theorem in order to fit the factoring problem on CRT-RSA. To be specific, we concentrate on the error term that appears in the mathematical derivation process. We carefully analyze and reduce the influence of this error term and hence obtain attack results that are better than previous studies [23,35].

3.1. Partial Key Exposure Attack on CRT-RSA

Recently, May, Nowakowski, and Sarkar [23] proposed an improved partial key exposure attack on CRT-RSA, which is based on the small CRT exponent attack working for $d_p,\ d_q < N^{0.122}$ of [18]. This attack applies to the range $N^{0.122} < d_p,\ d_q < N^{0.5}$, which smoothly covers the full interval of the CRT exponents. Let $N = pq$ be the CRT-RSA modulus, where p and q are prime numbers of the same bit-size. Assume that the public exponent is $e \approx N^{\alpha}$, the CRT-exponents are $d_p,\ d_q \approx N^{\beta}$. The CRT-exponents can be written as $d_p = d_p^* 2^t + \widetilde{d}_p,\ d_q = d_q^* 2^t + \widetilde{d}_q$, where $t \in \mathbb{N}$ and the LSB components $\widetilde{d}_p,\ \widetilde{d}_q \approx N^{\beta-\delta}$ are known, whereas the MSB components $d_p^*,\ d_q^* \approx N^{\delta}$ are unknown. Then, there exists a polynomial-time algorithm that outputs the secret information $p,\ q$ using the given partial key exposure.

According to $d_p \equiv e^{-1} \bmod (p-1)$ and $d_q \equiv e^{-1} \bmod (q-1)$, we have

$$ed_p = k(p-1) + 1 \tag{4}$$
$$ed_q = \ell(q-1) + 1 \tag{5}$$

for two positive integers $k,\ \ell \in \mathbb{N}$. Since the sizes of e, d_p, d_q, p, and q are known, the upper bounds on k and ℓ are estimated as

$$k, \ell < \max\left(\frac{ed_p}{p-1}, \frac{ed_q}{q-1}\right) = \Theta\left(\frac{N^{\alpha}N^{\beta}}{N^{1/2}}\right) = \Theta\left(N^{\alpha+\beta-1/2}\right). \tag{6}$$

Applying the Formula (4), we have $f(x_p, y_p) := x_p(y_p - 1) + 1 = x_p y_p - x_p + 1$, whose root is $(k,\ p) \bmod e$. In addition, multiplying the Formula (5) by p and rearranging it gives

$$ped_q = p\ell(q-1) + p = N\ell - p\ell + p = N(\ell - 1) + N - p(\ell - 1).$$

Similarly, we have $g(y_p, z_p) := y_p z_p - N z_p - N$, whose root is $(p,\ \ell - 1) \bmod e$. Moreover, the Equations (4) and (5) can be rewritten as $kp = k - 1 + ed_p$ and $\ell q = \ell - 1 + ed_q$. Multiplying them together gives $k\ell N = (k-1)(\ell-1) + (k-1)ed_q + ed_p(\ell-1) + e^2 d_p d_q$, and we obtain

$$(N-1)k(\ell-1) + Nk + (\ell-1) = e\big(d_q(k-1) + d_p(\ell-1) + ed_p d_q\big).$$

Therefore, we have $h(x_p, z_p) := (N-1)x_p z_p + N x_p + z_p$, whose root is $(k,\ \ell - 1) \bmod e$. Combining the above modular polynomials, we obtain the following system of polynomial equations,

$$\begin{cases} f(x_p, y_p, z_p) = x_p y_p - x_p + 1 = 0, \\ g(x_p, y_p, z_p) = y_p z_p - N z_p - N = 0, \\ h(x_p, y_p, z_p) = (N-1)x_p z_p + N x_p + z_p = 0. \end{cases}$$

Their common root is $(k,\ p,\ \ell - 1) \bmod e$.

Considering the special algebraic relationship between the unknown variables, we use six variables, namely $x_p,\ x_q,\ y_p,\ y_q,\ z_p,\ z_q$ instead of three variables. In this case, the

relevant elements in certain monomials of the original polynomials need to be converted algebraically as follows.

$$y_p y_q \longleftrightarrow N,\ x_p - 1 \longleftrightarrow x_q,\ x_q + 1 \longleftrightarrow x_p,\ z_p + 1 \longleftrightarrow z_q,\ z_q - 1 \longleftrightarrow z_p. \tag{7}$$

Thus, we obtain linear polynomials after the conversion,

$$\begin{cases} f(x_p, x_q, y_p, y_q, z_p, z_q) := x_p y_p - x_q, \\ g(x_p, x_q, y_p, y_q, z_p, z_q) := y_p z_p - N z_q, \\ h(x_p, x_q, y_p, y_q, z_p, z_q) := N x_p z_q - x_q z_p. \end{cases}$$

Combining known values $\widetilde{d}_p$ and $\widetilde{d}_q$, the polynomials involved in the partial key exposure attack are as follows.

$$\begin{cases} \widetilde{f}(x_p, x_q, y_p, y_q, z_p, z_q) := x_p y_p - x_q - e\widetilde{d}_p \\ \widetilde{g}(x_p, x_q, y_p, y_q, z_p, z_q) := y_p z_p - N z_q + e\widetilde{d}_q y_p \\ \widetilde{h}(x_p, x_q, y_p, y_q, z_p, z_q) := N x_p z_q - x_q z_p - e^2 \widetilde{d}_p \widetilde{d}_q - e\widetilde{d}_p z_p - e\widetilde{d}_q z_q. \end{cases} \tag{8}$$

The root is $(x'_p,\ x'_q,\ y'_p,\ y'_q,\ z'_p,\ z'_q) = (k,\ k-1,\ p,\ q,\ \ell - 1,\ \ell) \bmod 2^t e$, and the corresponding upper bounds are $x'_p,\ x'_q,\ z'_p,\ z'_q \le X = N^{\alpha+\beta-1/2}$, and $y'_p,\ y'_q \le Y = N^{1/2}$.

We apply the lattice-based method to extract $(x'_p,\ x'_q,\ y'_p,\ y'_q,\ z'_p,\ z'_q)$. The first step is to construct the shift polynomials. Before that, we need to define the monomial set, whose elements are contained in the shift polynomials. The monomial set is closely related to the columns of the lattice basis matrix in the second step, and directly affects the alignment of the matrix rows.

Definition 4. *Let $m \in \mathbb{Z}_+$. The monomial set $\widetilde{\mathcal{M}}$ is defined as*

$$\widetilde{\mathcal{M}} = \left\{ x_p^a y_p^b z_p^c \mid 0 \le a \le m,\ 0 \le c \le m,\ 0 \le b \le 2m \right\}.$$

To facilitate the exposition of the lattice construction, it is additional to define $\mathcal{M} \subseteq \widetilde{\mathcal{M}}$ as

$$\mathcal{M} = \left\{ x_p^a y_p^b z_p^c \mid 0 \le a \le m,\ 0 \le c \le m,\ 0 \le b \le a + c \right\}.$$

Furthermore, $\widetilde{\mathcal{M}} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3 \cup \mathcal{M}_4 \cup \mathcal{M}_5$ is partitioned using five disjoint subsets as follows.

$$\begin{aligned} \mathcal{M}_1 &:= \left\{ x_p^a y_p^b z_p^c \in \mathcal{M} \mid a \le c,\ b \le c - a \right\}, \\ \mathcal{M}_2 &:= \left\{ x_p^a y_p^b z_p^c \in \mathcal{M} \mid a > c,\ b < a - c \right\}, \\ \mathcal{M}_3 &:= \left\{ x_p^a y_p^b z_p^c \in \mathcal{M} \mid x_p^a y_p^b z_p^c \notin (\mathcal{M}_1 \cup \mathcal{M}_2),\ a + b + c \equiv 0 \bmod 2 \right\}, \\ \mathcal{M}_4 &:= \left\{ x_p^a y_p^b z_p^c \in \mathcal{M} \mid x_p^a y_p^b z_p^c \notin (\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3) \right\}, \\ \mathcal{M}_5 &:= \left\{ x_p^a y_p^b z_p^c \in \widetilde{\mathcal{M}} \mid x_p^a y_p^b z_p^c \notin \mathcal{M} \right\}. \end{aligned}$$

The shift functions and the corresponding shift polynomials are defined based on the monomial sets $\widetilde{\mathcal{M}}$ and $\mathcal{M}$.

Definition 5. *The shift functions are defined as follows.*

$$E_f(a,b,c) := \begin{cases} 0, & x_p^a y_p^b z_p^c \in \mathcal{M}_1 \\ b, & x_p^a y_p^b z_p^c \in \mathcal{M}_2 \\ (a+b-c)/2, & x_p^a y_p^b z_p^c \in \mathcal{M}_3 \\ (a+b-c+1)/2, & x_p^a y_p^b z_p^c \in \mathcal{M}_4 \\ a, & x_p^a y_p^b z_p^c \in \mathcal{M}_5 \end{cases}$$

$$E_g(a,b,c) := \begin{cases} b, & x_p^a y_p^b z_p^c \in \mathcal{M}_1 \\ 0, & x_p^a y_p^b z_p^c \in \mathcal{M}_2 \\ (-a+b+c)/2, & x_p^a y_p^b z_p^c \in \mathcal{M}_3 \\ (-a+b+c-1)/2, & x_p^a y_p^b z_p^c \in \mathcal{M}_4 \\ c, & x_p^a y_p^b z_p^c \in \mathcal{M}_5 \end{cases}$$

$$E_h(a,b,c) := \begin{cases} a, & x_p^a y_p^b z_p^c \in \mathcal{M}_1 \\ c, & x_p^a y_p^b z_p^c \in \mathcal{M}_2 \\ (a-b+c)/2, & x_p^a y_p^b z_p^c \in \mathcal{M}_3 \\ (a-b+c-1)/2, & x_p^a y_p^b z_p^c \in \mathcal{M}_4 \\ 0, & x_p^a y_p^b z_p^c \in \mathcal{M}_5 \end{cases}$$

$$E_x(a,b,c) := \begin{cases} a-b-c, & x_p^a y_p^b z_p^c \in \mathcal{M}_2 \\ 0, & x_p^a y_p^b z_p^c \in \mathcal{M}_1 \cup \mathcal{M}_3 \cup \mathcal{M}_4 \cup \mathcal{M}_5 \end{cases}$$

$$E_z(a,b,c) := \begin{cases} -a-b+c, & x_p^a y_p^b z_p^c \in \mathcal{M}_1 \\ 0, & x_p^a y_p^b z_p^c \in \mathcal{M}_2 \cup \mathcal{M}_3 \cup \mathcal{M}_5. \\ 1, & x_p^a y_p^b z_p^c \in \mathcal{M}_4 \end{cases}$$

For a given monomial $x_p^a y_p^b z_p^c \in \widetilde{\mathcal{M}}$, the corresponding shift polynomial is determined as

$$\widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q) := \widetilde{f}^{E_f} \widetilde{g}^{E_g} \widetilde{h}^{E_h} x_p^{E_x} z_p^{E_z} (2^t e)^{2m-(E_f+E_g+E_h)}. \tag{9}$$

It is known that all the shift polynomials $\widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q)$ have the common root $(k,\ k-1,\ p,\ q,\ \ell-1,\ \ell) \bmod (2^t e)^{2m}$. The shift polynomials are further adapted by using the algebraic relation Formula (7). We construct the updated shift polynomials under the following induced transformation rules.

Definition 6. *Let u be a polynomial defined over $x_p,\ x_q,\ y_p,\ y_q,\ z_p,\ z_q$. $\mathrm{tr}(u)$ is defined as the updated polynomial after the following transformation rules.*

1. *For all monomials, $y_p y_q$ are converted to N,*
2. *For monomials without y_p, x_p are converted to $x_q + 1$ and z_p are converted to $z_q - 1$,*
3. *For monomials with y_p, x_q are converted to $x_p - 1$ and z_q are converted to $z_p + 1$.*

Thus, $\mathrm{tr}(u)$ contains only monomials of the form $x_p^a y_p^b z_p^c$ and $x_q^a y_q^b z_q^c$, i.e., variables with subscripts p and q do not appear in a monomial at the same time.

The second step is to construct the lattice-basis matrix **B**. To do this, we need to define the monomial order and the polynomial order, which are used to arrange the order of columns and rows in the lattice-basis matrix. When we generate the lattice-basis matrix, a polynomial $\widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q)$ is first multiplied by some extra variables and then transformed by Definition 6. Finally, the upper bounds on the unknown variables are substituted and the polynomials are arranged in a one-to-one correspondence with the monomial order.

Definition 7. *The monomial order is defined as follows.*

$$x_p^{a_1} y_p^{b_1} z_p^{c_1} \prec x_p^{a_2} y_p^{b_2} z_p^{c_2} \Longleftrightarrow \begin{cases} c_1 < c_2 \\ c_1 = c_2,\ a_1 < a_2 \\ c_1 = c_2,\ a_1 = a_2,\ b_1 < b_2 \end{cases}$$

The lattice basis matrix $\mathbf{B}$ *and the polynomial order (namely the matrix row) are defined in the following way. The i-th column is related to the monomial* $\lambda_{[a,b,c]}$*, which is* $x_q^a y_q^{b/2} z_q^c$ *for even* b *and* $x_p^a y_p^{\lceil b/2 \rceil} z_p^c$ *for odd* b*. The parameters* a*,* b*,* c *are taken from the i-th smallest monomial* $x_p^a y_p^b z_p^c$ *in* $\widetilde{\mathcal{M}}$*. For* $x_p^a y_p^b z_p^c \in \mathcal{M}$*, the i-th row is associated with the coefficient vector of polynomial* $\mathrm{tr}\left(\widetilde{p}_{[a,b,c]} \cdot y_q^{\lfloor b/2 \rfloor}\right)\left(Xx_p, Xx_q, Yy_p, Yy_q, Xz_p, Xz_q\right)$*. For* $x_p^a y_p^b z_p^c \in \mathcal{M}_5$ *with even* b*, the i-th row takes* $\mathrm{tr}\left(\widetilde{p}_{[a,b,c]} \cdot y_q^{\lfloor (a+c)/2 \rfloor} \cdot y_q^{\lceil (b-a-c)/2 \rceil}\right)\left(Xx_p, Xx_q, Yy_p, Yy_q, Xz_p, Xz_q\right)$*. For* $x_p^a y_p^b z_p^c \in \mathcal{M}_5$ *with odd* b*, it takes* $\mathrm{tr}\left(\widetilde{p}_{[a,b,c]} \cdot y_q^{\lfloor (a+c)/2 \rfloor} \cdot y_p^{\lceil (b-a-c)/2 \rceil}\right)\left(Xx_p, Xx_q, Yy_p, Yy_q, Xz_p, Xz_q\right)$*. Then the lattice basis matrix* $\mathbf{B}$ *is lower triangular.*

We then apply the lattice reduction algorithm to $\mathcal{L}$ generated by the basis matrix $\mathbf{B}$. The first few approximately shortest basis vectors $\vec{v}_i$ of the output are converted to polynomial form v_i, which make up the system of integer equations $\mathcal{V}$. The third step is to solve the above system of integer equations. We apply the Gröbner basis computation to extract $(k,\ k-1,\ p,\ q,\ \ell-1,\ \ell)$. At this point, we finally obtain the secret information p and q. Under the above lattice-based solution process, we present the following theorem.

Theorem 2. *Let* $N = pq$ *be a sufficiently large CRT-RSA modulus, where* p *and* q *are prime numbers of the same bit-size. Assume that the public exponent is* $e \approx N^{\alpha}$*, the CRT exponents are* $d_p,\ d_q \approx N^{\beta}$*. The CRT exponents are written as* $d_p = d_p^* 2^t + \widetilde{d}_p$ *and* $d_q = d_q^* 2^t + \widetilde{d}_q$*, where* $t \in \mathbb{N}$ *and the LSB components* $\widetilde{d}_p,\ \widetilde{d}_q \approx N^{\beta-\delta}$ *are known, whereas the MSB components* $d_p^*,\ d_q^* \approx N^{\delta}$ *are unknown. Then given* $(N,\ e,\ \widetilde{d}_p,\ \widetilde{d}_q)$ *and ensuring* $\gcd(N-1,\ 2^t e) = O(1)$*, if the condition*

$$\delta < \frac{3 - 2\alpha - 2\beta}{10} \tag{10}$$

holds then p *and* q *can be computed in polynomial time in* $\log N$*.*

Proof. We construct the lattice-basis matrix $\mathbf{B}$ and generate the ω-dimensional full-rank lattice $\mathcal{L}$. Hence, each diagonal element $b_{i,i}$ of $\mathbf{B}$ contains the powers of $2^t e$, X, Y, and N, $(N-1)$, which are expressed as

$$b_{i,i} = (2^t e)^{E_{1,i}} X^{E_{2,i}} Y^{E_{3,i}} N^{E_{4,i}} (N-1)^{E_{5,i}}.$$

To simplify the computation of the lattice determinant, we should multiply the polynomials of the rows by the following appropriate multiplicative inverse

$$\left(N^{E_{4,i}} \left(\frac{N-1}{\gcd(2^t e, N-1)} \right)^{E_{5,i}} \right)^{-1} \bmod (2^t e)^{2m} \tag{11}$$

to eliminate the effect of powers of N and $(N-1)$. The updated diagonal elements are $b_{i,i}^* = (2^t e)^{E_{1,i}} X^{E_{2,i}} Y^{E_{3,i}} \gcd(2^t e, N-1)^{E_{4,i}^*}$. Meanwhile, the root solution remains $(k,\ k-1,\ p,\ q,\ \ell-1,\ \ell) \bmod (2^t e)^{2m}$. Since it is assumed that $\gcd(2^t e,\ N-1) = O(1)$ in the most favorable case, its influence can be ignored. The lattice determinant is calculated

asymptotically as $\det(\mathcal{L}) = |\det(\mathbf{B})| = (2^t e)^{s_e} X^{s_X} Y^{s_Y}$, where the corresponding exponents are calculated as follows.

$$s_e = \sum_{i=1}^{\omega} E_{1,i} = \sum_{x_p^a y_p^b z_p^c \in \widetilde{\mathcal{M}}} \Big(2m - E_f(a,b,c) - E_g(a,b,c) - E_h(a,b,c)\Big) = \frac{7}{3}m^4 + o\Big(m^4\Big),$$

$$s_X = \sum_{i=1}^{\omega} E_{2,i} = \sum_{x_p^a y_p^b z_p^c \in \widetilde{\mathcal{M}}} (a+c) = 2m^4 + o\Big(m^4\Big),$$

$$s_Y = \sum_{i=1}^{\omega} E_{3,i} = \sum_{x_p^a y_p^b z_p^c \in \widetilde{\mathcal{M}}} \frac{b}{2} = m^4 + o\Big(m^4\Big).$$

On the other hand, the dimension of the lattice $\mathcal{L}$ is

$$\omega = \sum_{x_p^a y_p^b z_p^c \in \widetilde{\mathcal{M}}} 1 = (m+1)^2(2m+1) = 2m^3 + o\Big(m^3\Big).$$

We apply the condition (3), i.e., $\det(\mathcal{L}) < (2^t e)^{2m\omega}$. By substituting $2^t = N^{\beta-\delta}$, $e = N^{\alpha}$, $X = N^{\alpha+\beta-1/2}$ and $Y = N^{1/2}$, we have

$$(\alpha+\beta-\delta)\cdot\frac{7}{3}m^4 + \left(\alpha+\beta-\frac{1}{2}\right)\cdot 2m^4 + \frac{1}{2}\cdot m^4 < (\alpha+\beta-\delta)\cdot 4m^4 + o\Big(m^4\Big).$$

After simplification, we obtain $\delta < (3-2\alpha-2\beta)/10$. The running time mainly depends on the LLL algorithm, while the running time of the Gröbner basis computation is negligible in comparison. Thus, its time complexity is $O\big(\omega^6(\log B)^3\big) = O\big(m^{21}(\log N)^3\big)$. Because m is a fixed number for generating the lattice, the time complexity is polynomial in $\log N$. □

We elaborate on the details of the lattice-basis matrix in the partial key exposure attack on CRT-RSA and present the specific attack algorithm. Furthermore, to understand Theorem 2 and the order of columns and rows in the lattice construction more intuitively, we give a toy example of the lattice basis matrix for $m = 1$.

Example 1. *According to Definition 7, the lattice basis matrix* $\mathbf{B}$ *for* $m = 1$ *is shown in Table 1, where* $\widetilde{e} := 2^t e$ *and* $\widetilde{f}$, $\widetilde{g}$, $\widetilde{h}$ *are the derived polynomials to be solved.*

Table 1. A toy example of the lattice basis matrix for $m = 1$.

	1	y_p	y_q	x_q	$x_p y_p$	$x_q y_q$	z_q	$y_p z_p$	$y_q z_q$	$x_q z_q$	$x_p y_p z_p$	$x_q y_q z_q$
$\widetilde{e}^2$	$\widetilde{e}^2$											
$y_p\widetilde{e}^2$	–	$\widetilde{e}^2 Y$										
$y_q\widetilde{e}^2$	–	–	$\widetilde{e}^2 Y$									
$x_q\widetilde{e}^2$	–	–	–	$\widetilde{e}^2 X$								
$\widetilde{f}\widetilde{e}$	–	–	–	–	$\widetilde{e}XY$							
$\widetilde{f}y_q\widetilde{e}$	–	–	–	–	–	$\widetilde{e}XY$						
$z_p\widetilde{e}^2$	–	–	–	–	–	–	$\widetilde{e}^2 X$					
$\widetilde{g}\widetilde{e}$	–	–	–	–	–	–	–	$\widetilde{e}XY$				
$\widetilde{g}y_q\widetilde{e}$	–	–	–	–	–	–	–	–	$\widetilde{e}XY$			
$\widetilde{h}\widetilde{e}$	–	–	–	–	–	–	–	–	–	$\widetilde{e}XY$		
$\widetilde{f}z_p\widetilde{e}$	–	–	–	–	–	–	–	–	–	–	$\widetilde{e}X^2Y$	
$\widetilde{f}\widetilde{g}z_p$	–	–	–	–	–	–	–	–	–	–	–	X^2Y

It is clear that the lattice basis matrix $\mathbf{B}$ *is indeed a lower triangular matrix, and that each new shift polynomial* $\widetilde{p}_i$ *introduces only one new monomial* λ_i.

We present the partial key exposure attack on CRT-RSA in Algorithm 1, which is denoted by $\mathcal{A}(N, e, 2^t, \widetilde{d}_p, \widetilde{d}_q)$.

Algorithm 1 The partial key exposure attack algorithm $\mathcal{A}(N, e, 2^t, \widetilde{d}_p, \widetilde{d}_q)$.

Input: CRT-RSA modulus N, public exponent e, 2^t and known LSBs $\widetilde{d}_p,\ \widetilde{d}_q$

$\widetilde{f}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow x_p y_p - x_q - e\widetilde{d}_p$

$\widetilde{g}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow y_p z_p - N z_q + e\widetilde{d}_q y_p$

$\widetilde{h}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow N x_p z_q - x_q z_p - e^2 \widetilde{d}_p \widetilde{d}_q - e\widetilde{d}_p z_p - e\widetilde{d}_q z_q$

$\widetilde{\mathcal{M}} \Leftarrow m \in \mathbb{Z}_+$ {construct the monomial set}

$E_f, E_g, E_h, E_x, E_z \Leftarrow \widetilde{\mathcal{M}}$ {generate shift exponents}

$\widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow \widetilde{f}^{E_f} \widetilde{g}^{E_g} \widetilde{h}^{E_h} x_p^{E_x} z_p^{E_z} (2^t e)^{2m-(E_f+E_g+E_h)}$

$\mathbf{B} \Leftarrow \widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q)$ {generate the lattice basis matrix}

$\vec{v}_i \Leftarrow \mathbf{B}$ {compute reduced basis vectors}

$\mathcal{V} \Leftarrow \vec{v}_i$ {construct the equation system}

$(k,\ k-1,\ p,\ q,\ \ell-1,\ \ell) \Leftarrow \mathcal{V}$ {extract the desired root}

return p and q

Output: The factorization of $N = pq$

3.2. Polynomial Time Equivalence of Computing the CRT-RSA Secret Key and Factoring

We show an explicit demonstration on the polynomial-time equivalence of computing the CRT-RSA secret key and factoring by invoking the partial key exposure attack algorithm $\mathcal{A}(N, e, 2^t, \widetilde{d}_p, \widetilde{d}_q)$. It is specified in the following theorem.

Theorem 3. *Let $N = pq$ be a sufficiently large CRT-RSA modulus, where p and q are prime numbers of the same bit size. Let e be the public exponent and $d_p,\ d_q$ be the corresponding CRT-exponents. Then the following statements hold under the condition $ed_p,\ ed_q < N^{3/2}$ and ensuring $\gcd(N-1,\ 2^t e) = O(1)$.*

- *Given $(N,\ e,\ p,\ q)$, d_p and d_q can be computed in polynomial time,*
- *Given $(N,\ e,\ d_p,\ d_q)$, p and q can be computed in polynomial time.*

When $e \approx N$ is close to the CRT-RSA modulus, the CRT-exponents satisfies the natural constraint $d_p,\ d_q < N^{1/2}$ (or $d_p d_q < N$).

Note that the error term $\gcd(N-1,\ 2^t e)$ is derived from the proof of Theorem 2. To simplify the lattice determinant computation, we multiply each row polynomial by an appropriate multiplicative inverse, i.e., (11) to eliminate the effect of powers of N and $(N-1)$. However, $\gcd(N-1,\ 2^t e)$ is at least equal to 2 and hence its influence cannot be completely eliminated. We further check whether the assumption $\gcd(N-1,\ 2^t e) = O(1)$ holds in Section 4. From an algorithmic perspective, this error term $\gcd(N-1,\ 2^t e)$ affects the computation efficiency. To achieve the same attacking effect, a larger error term requires lattices with higher dimension and hence the corresponding factoring attack takes more time.

Proof. We divide the proof into two parts. One part proves that knowing $(N,\ e,\ p,\ q)$ is sufficient to compute d_p and d_q in polynomial time. The other part proves that known $(N,\ e,\ d_p,\ d_q)$ is sufficient to compute p and q in polynomial time. According to the key generation algorithm, we have $d_p \equiv e^{-1} \bmod (p-1)$, $d_q \equiv e^{-1} \bmod (q-1)$. When $(N,\ e,\ p,\ q)$ is known, the values of $e^{-1} \bmod (p-1)$ and $e^{-1} \bmod (q-1)$ can be computed by applying the extended Euclidean algorithm. It is shown in Algorithm 2 and denoted by $\mathcal{E}(e, p-1, q-1)$.

Algorithm 2 The CRT-RSA key computation algorithm $\mathcal{E}(e, p-1, q-1)$

Input: CRT-RSA public exponent e, modulus factorization p and q

$(u_1, u_2, u_3) \leftarrow (0, 1, p-1)$
$(v_1, v_2, v_3) \leftarrow (1, 0, e)$
while $v_3 > 0$ **do**
  $r \leftarrow \lfloor u_3 / v_3 \rfloor$
  $(u_1, u_2, u_3) \leftarrow (u_1, u_2, u_3) - r \cdot (v_1, v_2, v_3)$
  $(u_1, u_2, u_3) \longleftrightarrow (v_1, v_2, v_3)$
end while
$(u_1, u_2, u_3) \leftarrow (0, 1, q-1)$
$(w_1, w_2, w_3) \leftarrow (1, 0, e)$
while $v_3 > 0$ **do**
  $r \leftarrow \lfloor u_3 / w_3 \rfloor$
  $(u_1, u_2, u_3) \leftarrow (u_1, u_2, u_3) - r \cdot (w_1, w_2, w_3)$
  $(u_1, u_2, u_3) \longleftrightarrow (w_1, w_2, w_3)$
end while
return v_1 and w_1

Output: CRT-exponents $d_p = e^{-1} \bmod (p-1)$ and $d_q = e^{-1} \bmod (q-1)$

It is known that the time complexity of the modular inverse computation is $O(\log N)$, so the CRT-RSA secret key computation algorithm $\mathcal{E}(e, p-1, q-1)$ can be done in polynomial time. To prove that knowing $(N,\ e,\ d_p,\ d_q)$ is sufficient to compute p and q in polynomial time, we slightly modify the original problem into the form of the partial key exposure attack on CRT-RSA. Denoting $t := \lceil \log_2(\max(d_p, d_q)) \rceil$ and letting $d_p^* = d_q^* = 0$, we have leaked LSBs $\widetilde{d}_p := d_p,\ \widetilde{d}_q := d_q$. By calling the partial key exposure attack algorithm $\mathcal{A}(N, e, 2^t, \widetilde{d}_p, \widetilde{d}_q)$, we can extract p and q in polynomial time. The specific algorithm is shown in Algorithm 3 and denoted by $\mathcal{F}(N, e, d_p, d_q)$.

Algorithm 3 The CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$

Input: CRT-RSA modulus N, public exponent e, CRT-exponents d_p and d_q

$t \leftarrow \lceil \log_2(\max(d_p, d_q)) \rceil$
$d_p^* \leftarrow 0,\ d_q^* \leftarrow 0$
$\widetilde{d}_p \leftarrow d_p,\ \widetilde{d}_q \leftarrow d_q$
$\widetilde{f}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow x_p y_p - x_q - e\widetilde{d}_p$
$\widetilde{g}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow y_p z_p - N z_q + e\widetilde{d}_q y_p$
$\widetilde{h}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow N x_p z_q - x_q z_p - e^2 \widetilde{d}_p \widetilde{d}_q - e\widetilde{d}_p z_p - e\widetilde{d}_q z_q$
$\widetilde{\mathcal{M}} \Leftarrow m \in \mathbb{Z}_+$ {construct the monomial set}
$E_f, E_g, E_h, E_x, E_z \Leftarrow \widetilde{\mathcal{M}}$ {generate shift exponents}
$\widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q) \leftarrow \widetilde{f}^{E_f} \widetilde{g}^{E_g} \widetilde{h}^{E_h} x_p^{E_x} z_p^{E_z} (2^t e)^{2m - (E_f + E_g + E_h)}$
$\mathbf{B} \Leftarrow \widetilde{p}_{[a,b,c]}(x_p, x_q, y_p, y_q, z_p, z_q)$ {generate the lattice basis matrix}
$\vec{v}_i \Leftarrow \mathbf{B}$ {compute reduced basis vectors}
$\mathcal{V} \Leftarrow \vec{v}_i$ {construct the equation system}
$(k,\ k-1,\ p,\ q,\ \ell-1,\ \ell) \Leftarrow \mathcal{V}$ {extract the desired root}
return p and q

Output: The factorization of $N = pq$

The time complexity of the CRT-RSA modulus factorization algorithm depends mainly on the partial key exposure attack, which can be done in $O((\log N)^3)$ according to Theorem 2. Thus, $\mathcal{F}(N, e, d_p, d_q)$ runs in polynomial time in $\log N$. However, Algorithm 3 partially answers the research problem. It remains to show that the CRT exponents satisfy the natural constraint, i.e., $d_p,\ d_q < N^{1/2}$ for $e \approx N$. For this purpose, we further apply the

judgment condition (10) in Theorem 2. Assume that $e \approx N^{\alpha}$, $d_p,\ d_q \approx N^{\beta}$, $d_p^*,\ d_q^* \approx N^{\delta}$, the partial key exposure attack on CRT-RSA works if

$$\delta < \frac{3 - 2\alpha - 2\beta}{10}.$$

In Algorithm 3, we have the implication $\delta = 0$, which shows that

$$0 < \frac{3 - 2\alpha - 2\beta}{10}.$$

Hence, it follows that $\alpha + \beta < 3/2$, i.e., $ed_p,\ ed_q < N^{3/2}$. The CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$ is feasible for $e \approx N$, and the CRT-exponents are in the range of $d_p,\ d_q < N^{1/2}$. So far, our research goal is achieved. □

3.3. Validation Experiments

We implement the CRT-RSA key computation algorithm $\mathcal{E}(e, p-1, q-1)$ and the CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$ based on SageMath [36]. The experimental platform is a personal computer running Windows 10 with Intel(R) Core(TM) i5-10500 CPU 3.10 GHz and 8 GB RAM. All numbers in the validation experiments are randomly selected. We summarize the experimental results as follows. The CRT-RSA key computation algorithm $\mathcal{E}(e, p-1, q-1)$ outputs the modular inverse values $e^{-1} \bmod (p-1)$ and $e^{-1} \bmod (q-1)$ quite efficiently. The CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$ runs with enough integer equations for solving the unknown variables and finally factors the modulus N. To be specific, after running the LLL algorithm, we can obtain a sufficient number of approximately shortest basis vectors to meet the requirements in the lattice-based method. The desired root can be efficiently extracted by transforming the lattice vectors into a system of integer equations and then applying the Gröbner basis computation.

As the CRT-RSA key computation algorithm is a direct call to the extended Euclidean algorithm, we focus on the CRT-RSA modulus factorization algorithm. In the experiments, we first choose an appropriate m to control the construction of the shift polynomials. We then construct the lattice basis matrix $\mathbf{B}$ and the ω-dimensional full-rank lattice $\mathcal{L}$. The LLL algorithm is applied to $\mathcal{L}$, and we transform the first few output vectors into a system of integer equations. We finally solve the system of integer equations using the Gröbner basis computation and extract the desired root.

The detailed experimental results are given in Table 2. The $\log_2 N$ column provides the bit size of the CRT-RSA modulus of the generated instance. The $\alpha \log_2 N$ column provides the bit size of the public exponent e. The $\beta \log_2 N$ column provides the bit size of the CRT exponents in our attacks. The lattice settings for conducting the factoring algorithms are indicated by the m and ω columns. The time consumption (recorded in seconds) of the LLL algorithm and the Gröbner basis computation are given in the Time-column. Two particular numerical CRT-RSA examples are given in Examples 2 and 3.

Table 2. Experimental results of our proposed factoring attack on CRT-RSA.

$\log_2 N$	$\alpha \log_2 N$	$\beta \log_2 N$	m	ω	Time
512	40	384	2	45	3
512	512	20	2	53	4
512	512	30	3	132	424
1024	100	512	2	45	11
1024	200	768	3	112	2482
1024	1024	103	4	245	84,221
2048	250	2000	2	45	39
2048	2048	80	2	53	51
2048	2048	200	3	130	7412

Example 2. *In order to check the correctness and validity of the CRT-RSA modulus factorization algorithm, we choose the following specific parameters and generate the test example.*

1. *Randomly generate 512-bit prime numbers p, q and the modulus $N = pq$,*
2. *Randomly generate 103-bit CRT exponents d_p and d_q,*
3. *Generate the public exponent e based on above p, q and d_p, d_q,*
4. *Denote the CRT-RSA key information as $(N,\ e,\ d_p,\ d_q)$.*

The values of the numerical CRT-RSA example are as follows.

$$\begin{aligned}
N = {} & 114347036081803578673339388300699529343751209241907643180876\\
& 458206667330922848834036065664761789957008250918211860604774\\
& 566584479411936175341623770289022043707073762248864386282628\\
& 990346540436123901362681015966801988295462057535857497607361\\
& 768945381482239937653004859004894600582582900917446891506894\\
& 263052143,\\
e = {} & 223240854903959329163056785473769020659799535319489715586782\\
& 390754702103216662596028424458821582218235232577019255075762\\
& 125125183259281313726576508416053530713656742151040100413422\\
& 917401393760099325929355604572256120456696898516220623395554\\
& 309873831847027002512654955273158052172095928389224394251597\\
& 30187731,\\
d_p = {} & 679371833132313743442009708179 5,\\
d_q = {} & 568930462627864390532226882625 1.
\end{aligned}$$

In the CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$, we choose $m = 4$, which means that we need to apply the LLL algorithm to the lattice $\mathcal{L}$ with $\omega = 245$. After running for almost 84221 s, the approximately shortest basis vectors that meet the solvable condition are obtained. The system of integer equations to be solved is then derived by transforming vectors into polynomials. We finally solve it by applying the Gröbner basis computation in less than one second and recover

$$\begin{aligned}
p = {} & 108638302771434350392709121078675710123453598877100622797142\\
& 999156910491518974867800775506243747756303770936993791366549\\
& 71732967591897327085997551822606809,\\
q = {} & 105254807158005691967534136739997233691243442214073820398841\\
& 619002989698022942925837218623347529765489194410746175686245\\
& 27351967084444332562641383615278727.
\end{aligned}$$

One may check that $N = pq$ does hold, so the CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$ successfully outputs the factorization of $N = pq$.

To show that the CRT-RSA key computation algorithm $\mathcal{E}(e, p-1, q-1)$ also runs in polynomial time, we use the above CRT-RSA example and apply extended Euclidean algorithm. It runs in less than one second and outputs

$$\begin{aligned}
d_p = {} & 679371833132313743442009708179 5,\\
d_q = {} & 568930462627864390532226882625 1.
\end{aligned}$$

Example 3. *In order to further check the correctness and validity of the CRT-RSA modulus factorization algorithm for smaller e and larger CRT exponents, we choose the following specific parameters and generate the test example.*

1. *Randomly generate 512-bit prime numbers p, q, and the modulus $N = pq$,*

2. *Randomly generate 200-bit public exponent e,*
3. *Generate 768-bit CRT-exponents d_p and d_q based on above p, q, and e,*
4. *Denote the CRT-RSA key information as $(N,\ e,\ d_p,\ d_q)$.*

The values of the numerical CRT-RSA example are as follows.

$$\begin{aligned}
N = {} & 156512694533516563511957875673047248424661607738276291458198\\
& 561587305961477716057755389974955796966826998822273532598444\\
& 073354133765045009506842533224118977797362970342863277332916\\
& 054340774252180351940497697856282577719012074116639472745869\\
& 837430174148381260146968287423827682202356316536216812133505\\
& 242483721,\\
e = {} & 870234913689148331430538521742157308545604222372763981584709,\\
d_p = {} & 105054920889694313236643515475986229698560379691663463225512\\
& 302039400442603589508021349904320178302089471845434045075084\\
& 406806664103314275213793412703224020726768063140359644831769\\
& 220002780936634680647859678632404963938328241541603 3,\\
d_q = {} & 148571987412482932015696111740202311634967482494902349345138\\
& 137535116614527333185153973996870696864633712947488523072563\\
& 209530823364587205330955760387748864552847890724262288611841\\
& 403762809796212436492624963331668748009958976827356 5.
\end{aligned}$$

In the CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$, we choose $m = 3$, which means that we need to apply the LLL algorithm to the lattice $\mathcal{L}$ with $\omega = 112$. After running for almost 2482 s, the approximately shortest basis vectors that meet the solvable condition are obtained. The system of integer equations to be solved is then derived by transforming vectors into polynomials. We finally solve it by applying the Gröbner basis computation in less than one second and recover

$$\begin{aligned}
p = {} & 117661909776482824125451009716243933710550298018498329113532\\
& 519144170968368272845860632479094497702166147221073340778054\\
& 061682494623967397830879204879383 03,\\
q = {} & 133018998952878525725128760686837839812976256555609678813921\\
& 757867209306278453420877763154370943445528651863267874726515\\
& 627272321450958081678652450445852 07.
\end{aligned}$$

One may check that $N = pq$ does hold, so the CRT-RSA modulus factorization algorithm $\mathcal{F}(N, e, d_p, d_q)$ successfully outputs the factorization of $N = pq$.

To show that the CRT-RSA key computation algorithm $\mathcal{E}(e, p-1, q-1)$ also runs in polynomial time, we use the above CRT-RSA example and apply extended Euclidean algorithm. It runs in less than one second and outputs d_p and d_q.

4. Discussion

To solve the proposed research problem, we apply the lattice-based method and present an improved CRT-RSA modulus factorization algorithm in Section 3.2. The correctness and validity of our factoring algorithm have been verified in Section 3.3. Our validation experiments always output the desired modulus factorization for small CRT-exponents. The running time depends on the particular lattice dimension and is in the range of 1 s to 10^6 s. Hence, the accuracy and speed of our factoring algorithm have been evaluated. Next, we discuss the difference between previous studies and ours and show the superiority of our approach.

Maitra and Sarkar [35] have proposed a factorization attack using lattice-based method when $(N,\ e,\ d_p,\ d_q)$ are known. Assume that one has $e \approx N^{\alpha}$, $d_p < N^{\delta_1}$, $d_q < N^{\delta_2}$ and $g := \gcd(N-1, ed_p - 1, ed_q - 1) \approx N^{\gamma}$. They demonstrated that N can be factored in polynomial time in $\log N$ when $2\alpha + \delta_1 + \delta_2 + 2\gamma < 3$. For $e \approx N$, i.e., $\alpha \approx 1$, we have $\delta_1 + \delta_2 < 1 - 2\gamma$. It implies that the above approach cannot reach the natural constraint since γ is not a negligible term.

Relatively speaking, our factoring algorithm involves an error term $\gcd(N-1, 2^t e)$ instead of $\gcd(N-1, ed_p - 1, ed_q - 1)$ and hence our result seems superior. To illustrate the superiority, we examine and compare $\gcd(N-1, ed_p - 1, ed_q - 1)$ in [35] and $\gcd(N-1, 2^t e)$ of ours. We choose 1024-bit CRT-RSA moduli and the corresponding CRT exponents of various bit sizes. The comparison results are given in Table 3. For the bit size of each CRT exponent, we randomly take 100 trials and calculate the average value.

Table 3. The comparison of previous error term and ours.

CRT-Exponents Bit-Size	$\gcd(N-1, ed_p - 1, ed_q - 1)$	$\gcd(N-1, 2^t e)$
192	53	6.18
256	16.18	6.78
320	19.22	3.9
384	13.62	8.8
448	14.08	2.26
512	16.74	4.38
576	18.5	3.1
640	61.78	4.5
704	18.52	7
768	16.46	8.28
832	26.02	5
896	17.52	4.16
960	21.4	4.76
1024	27.94	3.22

We observe that our error term $\gcd(N-1, 2^t e)$ equals to $O(1)$, i.e, negligible compared to $e,\ d_p,\ d_q$ in most cases. Further, our error term is always less than the previous one $\gcd(N-1, ed_p - 1, ed_q - 1)$. Hence, our approach performs better than the previous work [35]. As shown in Figure 1, the attack upper bound $\log_N d_p d_q$ of ours is closer to 1. Thus, our improvement further strengthens the modulus factoring attack on CRT-RSA when its key information is known.

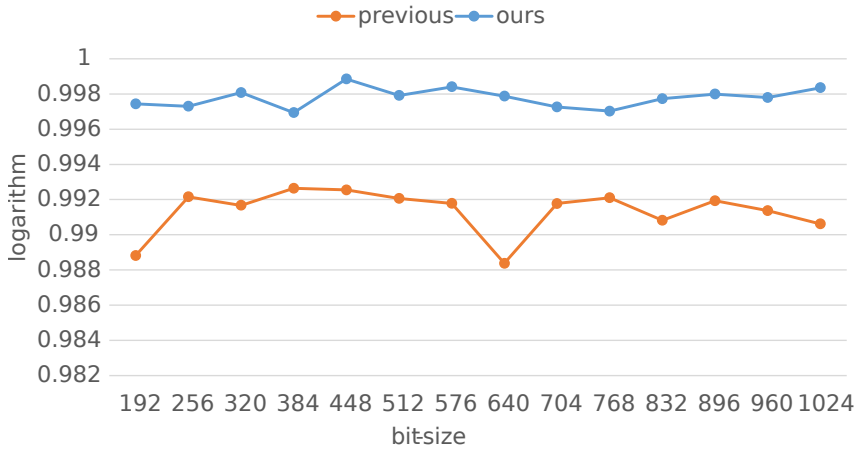

Figure 1. The comparison of previous work and ours with respect to the attack upper bound. The horizontal axis stands for the bit size of CRT exponents and the vertical one stands for the logarithm of the attack bound.

To illustrate the relation of our work to the field of attacking CRT-RSA, we compare and classify recent related works as shown in Table 4. Our work further enriches cryptanalyses of CRT-RSA. To be specific, the work not only makes further improvements on factoring attacks, but also reveals the importance and correctness of the polynomial-time equivalence of computing the CRT-RSA secret key and factoring. It can be seen that factoring attack is a more essential security threat.

Table 4. The comparison and classification of related works and ours on cryptanalysis of CRT-RSA.

Related Works	Attack Type	Used Method
ours, [35]	factoring attack	lattice-based method
[37]	factoring attack	elliptic curve method
[6,12,17,18,38]	small CRT-exponent attack	lattice-based method
[19–23]	partial key exposure attack	lattice-based method
[24–28]	side-channel attack	power-based method
[39–41]	key recovery attack	tree-based method

Notice that we experimentally verify our CRT-RSA modulus factorization algorithm for small CRT exponents. A limiting factor in achieving large CRT exponents is that we need to perform the lattice reduction algorithm with large lattice dimension in such cases. The running time increases rapidly, which leads to decreased attack efficiency, and is practically infeasible. This disadvantage can be eliminated with optimized lattice reduction algorithms and enhanced computing capability.

5. Conclusions

In this paper, we positively answer the research problem whether there exists a polynomial-time algorithm outputting the modulus factorization if given the CRT-RSA key information. Our work is summarized as follows.

- We study the security of CRT-RSA concerning its mathematical hardness assumption. We propose the improved polynomial-time algorithm of factoring the CRT-RSA modulus with the help of partial key exposure attack on CRT-RSA.
- Our asymptotic attack upper bound is superior to previous work [35]. Our factoring attack is a more essential security threat compared to others.
- We verify the correctness and validity of the proposed CRT-RSA modulus factorization algorithm with numerical experiments. Our validation experiments always successfully output the factorization results for small CRT-exponents.
- We discuss our results with several comparisons to properly assess the proposed factoring attack. Furthermore, we discuss the advantages and disadvantages of the proposed algorithm.

More concretely, under the condition that $ed_p,\ ed_q < N^{3/2}$, the factoring algorithm takes the given CRT-RSA key information $(N,\ e,\ d_p,\ d_q)$ as input and outputs the factorization of modulus $N = pq$ in polynomial time. We transform the original factorization problem into a modular polynomial solution problem. We apply Coppersmith's lattice-based method to find the root solution, which contains the prime factors of the modulus. Notice that the polynomial-time equivalence of CRT-RSA key computation and factorization is still not completely solved. One open problem is whether there exists a polynomial-time algorithm for factoring the CRT-RSA modulus in the case of $ed_p,\ ed_q > N^{3/2}$.

Future research should be devoted to the development of optimized lattice-reduction algorithm in order to enhance the efficiency of the proposed factoring attack. In addition, hybrid attacks based on the proposed CRT-RSA modulus factorization algorithm is an interesting topic for future work.

Funding: This work was supported by the National Natural Science Foundation of China, grant numbers 62002335 and Ningbo Natural Science Foundation, grant number 2021J174.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We thank Julian Nowakowski for sharing the experimental code of the partial key exposure attack on CRT-RSA. We thank anonymous reviewers for their helpful and constructive comments that greatly improved this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rivest, R.L.; Shamir, A.; Adleman, L.M. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
2. Boudot, F.; Gaudry, P.; Guillevic, A.; Heninger, N.; Thomé, E.; Zimmermann, P. The State of the Art in Integer Factoring and Breaking Public-Key Cryptography. *IEEE Secur. Priv.* **2022**, *20*, 80–86. [CrossRef]
3. Wiener, M.J. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inf. Theory* **1990**, *36*, 553–558. [CrossRef]
4. Coppersmith, D. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. Cryptol.* **1997**, *10*, 233–260. [CrossRef]
5. Boneh, D.; Durfee, G. Cryptanalysis of RSA with Private Key d Less than $N^{0.292}$. In *Advances in Cryptology—EUROCRYPT '99, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999*; Stern, J., Ed.; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1592, pp. 1–11. [CrossRef]
6. May, A. Cryptanalysis of Unbalanced RSA with Small CRT-Exponent. In *Advances in Cryptology—CRYPTO 2002, Proceedings of the 22nd Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2002*; Yung, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2442, pp. 242–256. [CrossRef]
7. Blömer, J.; May, A. New Partial Key Exposure Attacks on RSA. In *Advances in Cryptology—CRYPTO 2003, Proceedings of the 23rd Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2003*; Boneh, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2729, pp. 27–43. [CrossRef]
8. May, A. Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring. In *Advances in Cryptology—CRYPTO 2004, Proceedings of the 24th Annual International CryptologyConference, Santa Barbara, CA, USA, 15–19 August 2004*; Franklin, M.K., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3152, pp. 213–219. [CrossRef]
9. Ernst, M.; Jochemsz, E.; May, A.; de Weger, B. Partial Key Exposure Attacks on RSA up to Full Size Exponents. In A*dvances in Cryptology—EUROCRYPT 2005, Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005*; Cramer, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3494, pp. 371–386. [CrossRef]
10. Coron, J. Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach. In *Advances in Cryptology—CRYPTO 2007, Proceedings of the 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2007*; Menezes, A., Ed.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4622, pp. 379–394. [CrossRef]
11. Jochemsz, E.; May, A. A Polynomial Time Attack on RSA with Private CRT-Exponents Smaller Than $N^{0.073}$. In *Advances in Cryptology—CRYPTO 2007, Proceedings of the 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2007*; Menezes, A., Ed.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4622, pp. 395–411. [CrossRef]
12. Takayasu, A.; Lu, Y.; Peng, L. Small CRT-Exponent RSA Revisited. In *Part II, Advances in Cryptology—EUROCRYPT 2017—Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, 30 April 30–4 May 2017*; Coron, J., Nielsen, J.B., Eds. Springer International Publishing: Cham, Switzerland, 2017; Volume 10211, pp. 130–159. [CrossRef]
13. Coron, J.; May, A. Deterministic Polynomial-Time Equivalence of Computing the RSA Secret Key and Factoring. *J. Cryptol.* **2007**, *20*, 39–50. [CrossRef]
14. Moriarty, K.; Kaliski, B.; Jonsson, J.; Rusch, A. PKCS #1: RSA Cryptography Specifications Version 2.2. 2016. Available online: https://rfc-editor.org/rfc/rfc8017.txt (accessed on 20 May 2022).
15. Quisquater, J.J.; Couvreur, C. Fast Decipherment Algorithm for RSA Public-Key Cryptosystem. *Electron. Lett.* **1982**, *18*, 905–907. [CrossRef]
16. Garner, H.L. The Residue Number System. *IRE Trans. Electron. Comput.* **1959**, *8*, 140–147. [CrossRef]
17. Jochemsz, E.; May, A. A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In *Advances in Cryptology—ASIACRYPT 2006, Proceedings of the 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, 3–7 December 2006*; Lai, X., Chen, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4284, pp. 267–282. [CrossRef]

18. Takayasu, A.; Lu, Y.; Peng, L. Small CRT-Exponent RSA Revisited. *J. Cryptol.* **2019**, *32*, 1337–1382. [CrossRef]
19. Sarkar, S.; Maitra, S. Partial Key Exposure Attack on CRT-RSA. In *Applied Cryptography and Network Security, Proceedings of the 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, 2–5 June 2009*; Abdalla, M., Pointcheval, D., Fouque, P., Vergnaud, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5536, pp. 473–484. [CrossRef]
20. Lu, Y.; Zhang, R.; Lin, D. New Partial Key Exposure Attacks on CRT-RSA with Large Public Exponents. In *Applied Cryptography and Network Security—Proceedings of the 12th International Conference, ACNS 2014, Lausanne, Switzerland, 10–13 June 2014*; Boureanu, I., Owesarski, P., Vaudenay, S., Eds.; Springer International Publishing: Cham, Switzerland, 2014; Volume 8479, pp. 151–162. [CrossRef]
21. Takayasu, A.; Kunihiro, N. Partial Key Exposure Attacks on CRT-RSA: Better Cryptanalysis to Full Size Encryption Exponents. In *Applied Cryptography and Network Security—Proceedings of the 13th International Conference, ACNS 2015, New York, NY, USA, 2–5 June 2015*; Revised Selected Papers; Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M., Eds.; Springer International Publishing: Cham, Switzerland, 2015; Volume 9092, pp. 518–537. [CrossRef]
22. Takayasu, A.; Kunihiro, N. Partial Key Exposure Attacks on CRT-RSA: General Improvement for the Exposed Least Significant Bits. In *Information Security—Proceedings of the 19th International Conference, ISC 2016, Honolulu, HI, USA, 3–6 September 2016*; Bishop, M., Nascimento, A.C.A., Eds.; Springer International Publishing: Cham, Switzerland, 2016; Volume 9866, pp. 35–47. [CrossRef]
23. May, A.; Nowakowski, J.; Sarkar, S. Partial Key Exposure Attack on Short Secret Exponent CRT-RSA. In *Advances in Cryptology—Proceedings of the ASIACRYPT 2021—27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, 6–10 December 2021*; Part I; Tibouchi, M., Wang, H., Eds.; Springer International Publishing: Cham, Switzerland, 2021; Volume 13090, pp. 99–129. [CrossRef]
24. Schindler, W.; Wiemers, A. Generic power attacks on RSA with CRT and exponent blinding: New results. *J. Cryptogr. Eng.* **2017**, *7*, 255–272. [CrossRef]
25. Xu, S.; Lu, X.; Zhang, K.; Li, Y.; Wang, L.; Wang, W.; Gu, H.; Guo, Z.; Liu, J.; Gu, D. Similar operation template attack on RSA-CRT as a case study. *Sci. China Inf. Sci.* **2018**, *61*, 032111:1–032111:17. [CrossRef]
26. Xu, S.; Wang, W.; Lu, X.; Guo, Z.; Liu, J.; Gu, D. Side channel attack of multiplication in GF(q)-application to secure RSA-CRT. *Sci. China Inf. Sci.* **2019**, *62*, 39105:1–39105:3. [CrossRef]
27. Wan, W.; Chen, J.; Xia, J.; Zhang, J.; Zhang, S.; Chen, H. Clustering Collision Power Attack on RSA-CRT. *Comput. Syst. Sci. Eng.* **2021**, *36*, 417–434. [CrossRef]
28. Kaedi, S.; Doostari, M.; Ghaznavi-Ghoushchi, M.B.; Yusefi, H. A New Side-Channel Attack on Reduction of RSA-CRT Montgomery Method Based. *J. Circuits Syst. Comput.* **2021**, *30*, 2150038:1–2150038:18. [CrossRef]
29. Lenstra, A.K.; Lenstra, H.W.; Lovász, L. Factoring Polynomials with Rational Coefficients. *Math. Ann.* **1982**, *261*, 515–534. [CrossRef]
30. May, A. New RSA Vulnerabilities Using Lattice Reduction Methods. Ph.D. Thesis, University of Paderborn, Paderborn, Germany, 2003.
31. Coppersmith, D. Finding a Small Root of a Univariate Modular Equation. In *Advances in Cryptology—EUROCRYPT '96, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, 12–16 May 1996*; Maurer, U.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1070, pp. 155–165. [CrossRef]
32. Coppersmith, D. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In *Advances in Cryptology—EUROCRYPT '96, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, 12–16 May 1996*; Maurer, U.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1070, pp. 178–189. [CrossRef]
33. Howgrave-Graham, N. Finding Small Roots of Univariate Modular Equations Revisited. In *Cryptography and Coding, Proceedings of the 6th IMA International Conference, Cirencester, UK, 17–19 December 1997*; Darnell, M., Ed.; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1355, pp. 131–142. [CrossRef]
34. Becker, T.; Weispfenning, V.; Kredel, H. *Gröbner Bases—A Computational Approach to Commutative Algebra*; Graduate Texts in Mathematics; Springer: Berlin/Heidelberg, Germany, 1993; Volume 141.
35. Maitra, S.; Sarkar, S. On Deterministic Polynomial-Time Equivalence of Computing the CRT-RSA Secret Keys and Factoring. *Def. Sci. J.* **2012**, *62*, 122–126. [CrossRef]
36. The Sage Developers. SageMath, the Sage Mathematics Software System (Version 9.0). 2021. Available online: https://www.sagemath.org (accessed on 20 May 2022).
37. Somsuk, K. The Improvement of Elliptic Curve Factorization Method to Recover RSA's Prime Factors. *Symmetry* **2021**, *13*, 1314. [CrossRef]
38. Peng, L.; Takayasu, A. Generalized cryptanalysis of small CRT-exponent RSA. *Theor. Comput. Sci.* **2019**, *795*, 432–458. [CrossRef]
39. Oonishi, K.; Kunihiro, N. Attacking Noisy Secret CRT-RSA Exponents in Binary Method. In *Information Security and Cryptology—Proceedings of the ICISC 2018—21st International Conference, Seoul, Korea, 28–30 November 2018*; Revised Selected Papers; Lee, K., Ed.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11396, pp. 37–54. [CrossRef]

40. Oonishi, K.; Huang, X.; Kunihiro, N. Improved CRT-RSA Secret Key Recovery Method from Sliding Window Leakage. In *Information Security and Cryptology—Proceedings of the ICISC 2019—22nd International Conference, Seoul, Korea, 4–6 December 2019*; Revised Selected Papers; Seo, J.H., Ed.; Springer International Publishing: Cham, Switzerland, 2019; Volume 11975, pp. 278–296. [CrossRef]
41. Oonishi, K.; Kunihiro, N. Recovering CRT-RSA Secret Keys from Noisy Square-and-Multiply Sequences in the Sliding Window Method. In *Information Security and Privacy—Proceeding of the 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, 30 November—2 December 2020*; Liu, J.K., Cui, H., Eds.; Springer International Publishing: Cham, Switzerland, 2020; Volume 12248, pp. 642–652. [CrossRef]