Attacking Noisy Secret CRT-RSA Exponents in Binary Method

Kento Oonishi[1] and Noboru Kunihiro[1]

The University of Tokyo, Tokyo, Japan
kento_oonishi@mist.i.u-tokyo.ac.jp
kunihiro@k.u-tokyo.ac.jp

Abstract. In this paper, we perform a security evaluation on the RSA encryption scheme with the Chinese remainder theorem (CRT-RSA), against side-channel attacks. We discuss the methods for recovering the CRT-RSA secret keys by observing physical information. In the CRT-RSA scheme, we calculate the exponentiations by repeated squaring and multiplication operations during decryption. The square-and-multiply sequences of the exponentiation can be obtained by side-channel attacks. However, errors occur in the square and multiply sequences because of physical-information observation errors, due to which the secret keys cannot be recovered by using Bernstein et al.'s method, even if window size $w = 1$ in sliding window exponentiation. In this paper, we propose an algorithm for correcting the errors in the square-and-multiply sequences, and for obtaining the correct secret keys, when the square-and-multiply sequences are generated at $w = 1$, namely, the binary method. Moreover, we theoretically prove that the expected time complexity of our algorithm is in polynomial time, when the error rate is less than 5.8 %.

Keywords: CRT-RSA Encryption Scheme · Exponentiation · Error Correction · Side-Channel Attacks.

1 Introduction

1.1 Background

RSA encryption scheme [25] is an extensively used public-key cryptosystem. It is composed of public keys (N, e), and a secret key d. The public key e, is a sufficiently small parameter, such as $2^{16} + 1 = 65537$, which is used in many systems; whereas, N is the product of two distinct $n/2$ bit primes, p and q. The value-pair $(e, d) \in \mathbb{Z}^*_{(p-1)(q-1)} \times \mathbb{Z}^*_{(p-1)(q-1)}$, satisfies $ed \equiv 1 \mod (p-1)(q-1)$. In this paper, we consider the PKCS#1 standard [26], RSA with the Chinese remainder theorem (CRT-RSA). In the CRT-RSA scheme, the public keys are the same as those of the standard RSA scheme, while secret keys are (p, q, d, d_p, d_q, q_p). Parameters $d_p \in \mathbb{Z}^*_{p-1}, d_q \in \mathbb{Z}^*_{q-1}$, and $q_p \in \mathbb{Z}_p$ are defined as $d_p := d \mod p-1, d_q := d \mod q-1$, and $q_p := q^{-1} \mod p$, respectively. These additional secret keys enable faster decryption using the CRT.

The security of the RSA scheme is based on the difficulty of the factorization problem. However, even if the factorization problem is difficult, the implemented RSA is not always secure against physical attacks, such as cold boot attacks [7] and side-channel attacks [14]. As the secret keys can be leaked by physical attacks, the RSA scheme must be secure against such attacks.

In previous research, security analysis of the RSA scheme against physical attacks, such as cold boot and side-channel attacks, were conducted. In cold boot attacks, attackers observe the DRAM data remanence and read the secret-key bits. In side-channel attacks, they observe physical data based on secret keys, such as the power consumption [11, 15], implementation time [14], electromagnetic [5], sound [6], and cache access [2, 12, 23, 30, 31], during RSA decryption. We review these physical attacks on the RSA scheme.

Cold Boot Attacks on the RSA Scheme. Before cold boot attacks were introduced, there were several studies on the consecutive partial leakage of the RSA secret-key bits. The first key recovery method was proposed by Rivest and Shamir [24]. They proved that the RSA scheme can be broken, when 2/3 of the most or least significant bits of p are known. Further, Coppersmith [4] proved that the RSA scheme can be broken, when half of the most significant bits of p are known. In the Coppersmith method, attackers recover the RSA secret keys by solving equations on the small unknown values of the secret keys using LLL reduction [19]. Recent studies on key recovery from known consecutive bits are based on the Coppersmith method [29].

After the emergence of cold boot attacks, the recovery of the RSA secret keys, based on cold boot attacks, were extensively researched. In cold boot attacks, the RSA secret keys are given as bits with non-consecutive erasure and error; therefore, the setting of the given consecutive bits in previous research is an unnatural assumption. Hence, several models for the partial RSA secret keys have been proposed, and a key recovery algorithm was constructed, based on Heninger and Shacham's work [9] in CRYPTO 2009, in which RSA secret keys with erasure bits were considered, and the search for the RSA secret keys was thorough a binary tree. Based on this, studies on the various settings of noisy bits were conducted. For example, Henecka et al. [8] considered bits with random errors, and Kunihiro et al. [17] considered bits with both erasure and error. Similarly, RSA-secret-key recovery has been considered for various settings in discrete [22] and analog data [16, 18]. The above mentioned methods recover the RSA secret keys from the least significant bits. In addition to these methods, methods for recovering the RSA secret keys from the most significant bits by changing the method of constructing the tree have been proposed [13, 20, 27, 28].

Side-Channel Attacks on the RSA Scheme. Since the proposal of the timing attack [14] in 1996, there have been many side-channel attacks on the RSA scheme. These attacks monitor the exponentiations during decryption and extract the RSA secret keys as square-and-multiply sequences. Obtaining these

sequences is easier than obtaining the bits because cold boot attacks require the device itself of a regular user, whereas side-channel attacks do not.

Exponentiation is implemented efficiently using the binary, fixed window, and sliding window methods. Recent studies focus on attacking the fixed window [5, 11, 31] and sliding window methods [2, 5, 11, 12]. In particular, Bernstein et al. [2] have proposed a method for recovering CRT-RSA secret keys from square-and-multiply sequences without knowing the multiplier.

1.2 Motivation

Previous studies on side-channel attacks on the RSA scheme have mainly focused on observing the physical data precisely; they assumed that there were no errors in the square-and-multiply sequences. Bernstein et al. attacked the exponentiation using the sliding window method. They demonstrated that, if the square-and-multiply sequences can be obtained correctly, the CRT-RSA secret keys can be recovered, when the window size is $w \leq 4$.

However, there are errors in the square-and-multiply sequences obtained from physical data [1, 2]. Bernstein et al. [2], in particular, report that there is an average of 14 errors in the square-and-multiply sequences of 1024-bit CRT-RSA decryption, under the sliding window method at $w = 4$. This corresponds to a 1.1 % error rate in the square-and-multiply sequences. If there are errors in these sequences, Bernstein's method fails even if $w = 1$. In order to correct the errors in the square-and-multiply sequences, they proposed a majority vote by observing the physical data repeatedly, and declared that their method succeeds when the physical data is observed 20 times. However, there is no guarantee that attackers can observe the physical data repeatedly because decryption depends on the regular user. A similar problem arises in other methods, such as the differential power analysis [15].

In view of the above, we focus on correcting the errors in the square-and-multiply sequences, based on one observation alone. In cold boot attacks, various bits models are studied. In these researches, if key recovery succeeds, the correct CRT-RSA secret keys can be obtained. Therefore, the errors in the bits can be corrected. In side-channel attacks, the errors in the square-and-multiply sequences are considered in [12]; they proposed a key recovery algorithm, using square-and-multiply sequences with error. However, this study is ambiguous regarding the construction of the algorithm. Moreover, it does not reveal the error that can be recovered by their algorithm.

Thus, we require a key recovery algorithm using square-and-multiply sequences with small errors, extracted from one physical data alone. In addition, the error, in the square-and-multiply sequences, that can be recovered must be revealed.

1.3 Our Contribution

In this paper, we present a key recovery algorithm using the square-and-multiply sequences on exponentiations, c^{d_p} and c^{d_q}, with small errors at $w = 1$, namely,

the binary method. In order to recover the CRT-RSA secret keys from square-and-multiply sequences with error, we focus on the similarity between our attacking situation and that of Henecka et al.'s [8], whose algorithm recovers CRT-RSA secret key bits with error using a binary tree. We focus on the relationship between the bits and the square-and-multiply sequences, and propose a key recovery algorithm based on the square-and-multiply sequences.

In addition, we analyze our proposed algorithm theoretically. We prove that we can recover with an error rate less than 5.8 % in the square-and-multiply sequences in the expected polynomial time, if square-and-multiply sequences with error are provided from one physical data, by observing c^{d_p} and c^{d_q}. Our algorithm works well on the errors in the square-and-multiply sequences, as demonstrated experimentally in [2]; the average error rate is 1.1 %.

2 Preliminaries

In this section, we introduce the encryption and decryption of the CRT-RSA scheme and the binary method. In addition, we present the leakage model of a square-and-multiply sequence, and define the problem dealt with in this study. Finally, we introduce the notations used in our proposed algorithm, and the analysis of our algorithm.

2.1 Encryption and Decryption of the CRT-RSA Scheme

First, we introduce the encryption and decryption of the standard RSA scheme. In standard RSA scheme, the public keys are (N, e) and the secret key is d. The encryption of a message m, is performed as $C = m^e \mod N$, and the decryption as $m = C^d \mod N$.

Next, we present the encryption and decryption of the CRT-RSA scheme. In the CRT-RSA scheme, the public keys are (N, e) and the secret keys are (p, q, d, d_p, d_q, q_p). The mathematical relationship between the public and secret keys can be expressed using $k, k_p, k_q \in \mathbb{Z}$ as follows:

$$\begin{aligned} N &= pq, \\ ed &= 1 + k(p-1)(q-1), \\ ed_p &= 1 + k_p(p-1), \\ ed_q &= 1 + k_q(q-1). \end{aligned}$$

The encryption is the same as that of the standard RSA scheme. The decryption algorithm is given by Algorithm 1.

While the standard RSA decryption calculates under modulus N , the CRT-RSA decryption calculates under moduli p and q. Therefore, the exponentiation deals with half bits compared to the standard RSA scheme; decryption in the CRT-RSA scheme is approximately four times faster than that in the standard RSA scheme.

Algorithm 1 CRT-RSA Decryption Algorithm [26]

Input: Ciphertext c, and secret keys (p, q, d, d_p, d_q, q_p).
Output: Message m
Compute $m_1 = c^{d_p} \mod p, m_2 = c^{d_q} \mod q$
Compute $h = (m_1 - m_2)q_p \mod p$
Compute $m = m_2 + qh$
return m

2.2 Exponentiation (Binary Method)

The binary method is performed as Algorithm 2. With this Algorithm 2, we read d from the most-significant-bit side. Exponentiation includes two operations: squaring and multiplication. When the bit is zero, we perform squaring; when the bit is one, we perform squaring followed by multiplication. In this paper, **S** denotes squaring and **M** denotes multiplication. We write the implementation record using **S** and **M**. Using Algorithm 2, when we convert c^d as a square-and-multiply sequence; we convert a zero bit of d into **S**, and a one bit of d into **SM**. For example, when we calculate c^5, the binary representation of 5 is $5 = 101_2$; therefore, the square-and-multiply sequence is **SMSSM**. In a square-and-multiply sequence, the first operation is always **S**, and the **M**s are not consecutive. Moreover, the number of **S**s is the same as the number of bits of d, and the number of **M**s is the same as the number of one bits in d.

Algorithm 2 Exponentiation (binary method) [21]

Input: c and $d = (d_t \dots d_0)_2$
Output: c^d
Set $x = 1$
for $i = t$ **down to** 0
  Compute $x = x^2$ (Squaring)
  if $d_i = 1$
    Compute $x = cx$ (Multiplication)
end for
return x

2.3 Leakage Model

In CRT-RSA decryption, we execute modular exponentiations, c^{d_p} and c^{d_q}. From these operation, we obtain the square-and-multiply sequence by physical attacks. If these sequences can be obtained correctly, we can obtain the secret keys easily because the bits and the square-and-multiply sequence have a one-to-one correspondence. However, we do not always obtain the square-and-multiply sequence correctly.

In this paper, we consider that attackers obtain a square-and-multiply sequence, in which each operation, **S** (resp. **M**), is incorrectly judged as another operation, **M** (resp. **S**), with probability $\delta > 0$, independently. When $\delta = 0$, the above attacking situation is the same as that in Bernstein et al. However, in our model, the attacker knows the public keys and the square-and-multiply sequence of c^{d_p} and c^{d_q} with error. We propose an algorithm that can solve our problem, and analyze the upper bound of the error rate δ, that can be recovered in the expected polynomial time.

2.4 Notations

We use notations similar to those in [9]. First, we introduce the notations used in our algorithm. We rewrite each bit of integer $x \geq 0$, as $x = x_{n-1}x_{n-2}\dots x_0$ and $x[i] = x_i (0 \leq i \leq n-1)$. Next, we define $\tau(x) = \max_{m \in \mathbb{Z}} 2^m | x$. Moreover, we define Slice(i) as $(p[i], q[i], d_p[i + \tau(k_p)], d_q[i + \tau(k_q)])$. Slice$(i)$ denotes the corresponding bits of secret keys p, q, d_p, d_q.

Next, we introduce the notations used in the analysis of our algorithm. In the analysis, we define the entropy function $H(x)$, as $H(x) = -x \log x - (1-x)\log(1-x)$. The base of log is two. In addition, we define the inverse function of the entropy function. We define $y = H^{-1}(x) (0 < x \leq 1)$ as y such that $H(y) = x$ and $0 < y \leq 1/2$. In addition, we define α_t and β_t as following.

$$\alpha_t = \frac{1}{2} - \frac{\log(2t+1)+1}{2t}, \ \beta_t = H^{-1}(\alpha_t).$$

3 Previous CRT-RSA Secret Key Recovery Algorithm [8]

In this section, we review Henecka et al.'s key recovery algorithm. Their method is based on the RSA-key-candidate tree [9]. We review the construction of the binary tree and Henecka et al.'s work.

First, we review the key candidate tree proposed in [9]. When the public keys (N, e), and parameters (k_p, k_q), are given, we can construct the key candidate tree. The tree depth corresponds to the number of recovered bits of p and q.

In order to construct the tree, we calculate the root of tree, initially, as Slice$(0) = (1, 1, d_p[\tau(k_p)], d_q[\tau(k_q)])$, and the lower bits of d_p, d_q, namely, $d_p[i] \ (0 \leq i \leq \tau(k_p) - 1)$ and $d_q[i] \ (0 \leq i \leq \tau(k_q) - 1)$. We can calculate these using known data, such as the public keys (N, e), and parameters (k_p, k_q).

From the root, we can calculate the candidates for the CRT-RSA secret keys. If bits below Slice$(i-1)$ are given in each variable, we can calculate Slice(i) as follows: We define $p' = \sum_{j=0}^{i-1} p[j]2^j, q' = \sum_{j=0}^{i-1} q[j]2^j, d'_p = \sum_{j=0}^{i+\tau(k_p)-1} d_p[j]2^j, d'_q = \sum_{j=0}^{i+\tau(k_q)-1} d_q[j]2^j$. We calculate these values using known information. Then, we

solve

$$p[i] + q[i] \equiv (N - p'q')\,[i] \mod 2, \tag{1}$$

$$d_p[i + \tau\,(k_p)] + p[i] \equiv \left(k_p\,(p' - 1) + 1 - ed'_p\right)[i + \tau\,(k_p)] \mod 2, \tag{2}$$

$$d_q[i + \tau\,(k_q)] + q[i] \equiv \left(k_q\,(q' - 1) + 1 - ed'_q\right)[i + \tau\,(k_q)] \mod 2. \tag{3}$$

We can calculate the right-hand sides of (1)–(3) from known information. These simultaneous equations have two solutions; therefore, if the bits below Slice$(i-1)$ are given in each variable, we can calculate two candidates for Slice(i). Thereby, we can calculate the candidates for the CRT-RSA secret keys through the binary tree.

By repeating this operation, until we calculate Slice$(n/2-1)$, we can calculate the candidates for the CRT-RSA secret keys; the CRT-RSA secret keys always have $2^{n/2}$ candidates. However, finding the correct secret keys consumes tremendous time. Therefore, we adopt the branch and bound algorithm to searching for the secret key candidates. Previous studies prune leaves that do not resemble the observed data, in the binary tree.

Next, we review Henecka et al.'s work. Attackers are given the public keys (N, e), and the secret keys bits with errors. Each bit is judged incorrectly as another bit with probability $\delta > 0$, independently. Their algorithm calculates t slices using (1)–(3), and compares the Hamming distance between the t new bits and observed data, pruning leaves, whose Hamming distance is greater than C. Using this algorithm, when we are given two secret keys d_p and d_q, we can recover with an error rate of 11 % in the expected polynomial time.

4 Proposed CRT-RSA Secret Key Recovering Algorithm

4.1 Proposed Algorithm

First, we calculate the candidates for (k_p, k_q), as described in [9]. The number of candidates for (k_p, k_q) is $2(e-1)$; therefore, we construct $2(e-1)$ trees.

Next, we calculate the key candidate in each tree, using the branch and bound strategy. We calculate t new unknown slices using simultaneous equations (1)–(3). We set parameter t, to obtain the expected time complexity of our algorithm's polynomial time (see Sect. 4.2). After calculating the t new unknown slices, we convert t bits of d_p and d_q into square-and-multiply sequences, 0 to **S** and 1 to **SM**.

We then prune the leaves that do not match the observed sequence. We further calculate the disagreement rate between the calculated and observed sequences. We pay attention to the t new bits of d_p and d_q, and when the disagreement rate is strictly more than Y, we prune the leaf. We set parameter Y, to obtain the expected time complexity of our algorithm's polynomial time (see Sect. 4.2).

An example of pruning is illustrated in Fig. 1. We compare candidates and observed data from the rightmost operation. For candidate 1, we consider 10

Fig. 1. Example of pruning, when $Y = 0.15$

	d_p	d_q	Disagreement Rate	Result
Observed Data	**SMSSSM**	**MSSMSS**		
Candidate 1	**SMSS**	**SMSMSM**	5/10=0.5	Prune
Candidate 2	**SSSM**	**SSMSM**	1/9=0.111	Remain
Candidate 3	**SSS**	**SSSM**	3/7=0.429	Prune

operations: four in d_p and six in d_q. For d_p, we compare **SMSS** in candidate 1 and **SSSM** in the observed data; therefore, the number of disagreements is two. Similarly, for d_q, we compare **SMSMSM** of candidate 1 and **MSSMSS** in the observed data; therefore, the number of disagreements is three. The number of disagreements between candidate 1 and the observed data is five; thus, the disagreement rate is $5/10 = 0.5$. Therefore, candidate 1 is pruned because $0.5 > 0.15$. Candidates 2 and 3 are similar.

We repeat this step, until Slice$(n/2 - 1)$ are calculated, and search for the leaf that satisfies $N = pq$. The proposed algorithm is given by Algorithm 3.

Algorithm 3 Proposed CRT-RSA Secret Key Recovery Algorithm

Input: Public keys (N, e), square-and-multiply sequences of c^{d_p}, c^{d_q} with error, number of expansions t, threshold Y
Output: Secret keys (p, q, d_p, d_q)
Each k_p, k_q
  Calculate the root of the tree.
  for $i = 1$ **to** $\lceil (n/2 - 1)/t \rceil$
    Compute Slice$((i-1)t + 1), \cdots,$ Slice(it).
    Transform the new bits of d_p, d_q into square-and-multiply sequences.
    Prune the leaves, whose disagreement rates are strictly more than Y.
  end for
  Search for the leaf that satisfies $N = pq$.

4.2 Analysis of the Proposed Algorithm

In this subsection, we analyze Algorithm 3. We introduce a heuristic assumption [9] where, if the values of k_p and k_q are correct, the slice calculated from the incorrect leaf is random. Moreover, we assume that, when the values of k_p and k_q are incorrect, the slice calculated is random.

The goal of our analysis is to prove the following Theorem 1:

Theorem 1. Let $\varepsilon > 0$, $t = \left\lceil \ln(n) / \left(4\varepsilon^2\right) \right\rceil$. Moreover, let Y satisfy $Y \leq \beta_t / (2 - \beta_t)$. Then, when $\delta < Y - \varepsilon$, the expected time complexity of Algorithm 3 is $O\left(n^{2+\frac{\ln 2}{2\varepsilon^2}}\right)$, and the success rate of Algorithm 3 is at least $1 - \left(\frac{2\varepsilon^2}{\ln n} + \frac{1}{n}\right)$.

Remark 1. When $n \to \infty$, β_t converges to $H^{-1}(1/2) = 0.11$. Therefore, theorem 1 implies that $0.11/(2 - 0.11) = 0.058$ is the upper bound of the error rate that can be recovered in expected polynomial time. Moreover, the success rate converges to 1 in $n \to \infty$.

Analysis strategy. Before proving Theorem 1, we review the analysis in [8], wherein the parameters satisfy the condition that the expected number of new leaves generated from incorrect partial keys is $1/2$, after calculating t slices and pruning. This guarantees that the bound of the expected number of leaves after pruning is constant. Then, because the expected number of all leaves in Henecka et al.'s algorithm is polynomial in n, the expected time complexity of their algorithm is polynomial in n.

We evaluate the number of remaining leaves, when we calculate t slices and prune the candidates for parameter Y. When t slices are calculated, 2^t leaves are generated from one leaf. As heuristic assumption in [9], the new $2t$ bits of d_p, d_q calculated from the incorrect partial keys are independent of the 2^{2t} elements. Thus, if these elements are reduced to less than 2^{t-1} by pruning, the expected number of new leaves generated from the incorrect partial keys is less than $1/2$.

In [8], the analytical result is obtained directly from the Hoeffding inequality [10]. However, in square-and-multiply sequences, there are restrictions between each operation. Moreover, same-length key bits have different lengths in a square-and-multiply sequence. These restrictions render analysis hard, when square-and-multiply sequences are given. In order to address this problem, we analyze each length of a square-and-multiply sequence.

When we calculate t slices, we convert t bits of d_p and d_q into square-and-multiply sequences. We deal these $2t$ bits together. In these bits, the number of $\mathbf{M}$ in the square-and-multiply sequence is equal to the number of "one" bits. When we define the number of one bits in $2t$ bits as $t_{\mathbf{M}}$ $(0 \leq t_{\mathbf{M}} \leq 2t)$, the length of the sequence is $2t + t_{\mathbf{M}}$. If there are less than $2^{t-1}/(2t+1)$ leaves in each $t_{\mathbf{M}}$ after pruning, the number of leaves is reduced to 2^{t-1} because $\sum_{t_{\mathbf{M}}=0}^{2t} 2^{t-1}/(2t+1) = 2^{t-1}$. We now analyze the condition, Y, where less than $2^{t-1}/(2t+1)$ leaves remain in each $t_{\mathbf{M}}$ after pruning. Y indicates the condition for the error rate that can be corrected in the expected polynomial time. We focus on the analysis of Y and the details of the proof of Theorem 1 with respect to the setting of t, the time complexity, and the success rate, shown in full version.

Analysis of the upper bound of Y. We now prove the following Lemma 1 that shows the condition, Y, where less than $2^{t-1}/(2t+1)$ leaves remain in each $t_{\mathbf{M}}$, after pruning.

Lemma 1. When $0 \leq t_{\mathbf{M}} \leq 2t\beta_t$ or $2t(1-\beta_t) \leq t_{\mathbf{M}} \leq 2t$, the number of leaves is always less than $2^{t-1}/(2t+1)$. When $2t\beta_t < t_{\mathbf{M}} < 2t(1-\beta_t)$, less than $2^{t-1}/(2t+1)$ leaves remain after pruning, under $Y \leq 2t\beta_t/(2t+t_{\mathbf{M}})$.

In order to prove Lemma 1, we use the following the Lemma 2 proved in Appendix, and the Lemma 3 from [3].

Lemma 2. Let $L \in \mathbb{N}$, $t_{\mathbf{M}}$ be an integer that satisfies $0 \leq t_{\mathbf{M}} \leq \lfloor L/2 \rfloor$, and C be an integer that satisfies $0 \leq C \leq L$. Then, the number of sequences with length L, including $t_{\mathbf{M}}$ **M**s, generated from the bits, and whose number of disagreements is less than C compared to the observed data, is less than

$$\begin{cases} \binom{L-t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq \min\left(t_{\mathbf{M}}, L-2t_{\mathbf{M}}\right) \\ \binom{L-t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ \min\left(t_{\mathbf{M}}, L-2t_{\mathbf{M}}\right) \leq C \leq L \end{cases}$$

.

Lemma 3. [3] Let a, b be nonnegative integers. When $b \leq a$, it holds that $\binom{a}{b} \leq 2^{aH(b/a)}$.

Remark 2. We explain the relationship between Lemma 1 and the error rate that can be recovered. We now consider the average and the worst cases, when $t \to \infty$ that β_t converges to $H^{-1}(1/2) = 0.11$. First, we consider the average case. In practice, as there are random 0 and 1 values in the $2t$ bits, $t_{\mathbf{M}} = t$ with a high probability. Therefore, $Y \leq 2/3 \times 0.11 = 0.073$, and we can correct a 7.3 % error rate in the square-and-multiply sequences practically. From [8], we can correct an 11 % error rate in the bits. This difference occurs because one bit corresponds to 1.5 operations on an average, and the information in a character in an operation is 2/3 times as much as that in a bit. We next consider the worst case. In Lemma 1, the strictest upper bound of Y is $\beta_t/\left(2-\beta_t\right)$. Therefore, $0.11/(2-0.11) = 0.058$ is the worst case of the error rate that can be recovered in the expected polynomial time. We consider the latter worst case in Theorem 1.

Proof. We consider a sequence with length, $L = 2t + t_{\mathbf{M}}$, that is generated from $2t$ bits and includes $t_{\mathbf{M}}$ **M**s. When $L = 2t + t_{\mathbf{M}}$, the condition, where the leaf is not pruned is $C = (2t + t_{\mathbf{M}})\, Y$. Therefore, from Lemma 2, the number of leaves after pruning is less than

$$\begin{cases} \binom{2t}{(2t+t_{\mathbf{M}})\,Y} & \text{if } \ 0 \leq (2t+t_{\mathbf{M}})\, Y \leq \min\left(t_{\mathbf{M}}, 2t-t_{\mathbf{M}}\right) \\ \binom{2t}{t_{\mathbf{M}}} & \text{if } \ \min\left(t_{\mathbf{M}}, 2t-t_{\mathbf{M}}\right) \leq (2t+t_{\mathbf{M}})\, Y \leq 2t+t_{\mathbf{M}}. \end{cases}$$

Thus, from Lemma 3, the number of leaves after pruning is less than

$$\begin{cases} 2^{2tH((2t+t_{\mathbf{M}})Y/2t)} & \text{if } \ 0 \leq (2t+t_{\mathbf{M}})\, Y \leq \min\left(t_{\mathbf{M}}, 2t-t_{\mathbf{M}}\right) \\ 2^{2tH(t_{\mathbf{M}}/2t)} & \text{if } \ \min\left(t_{\mathbf{M}}, 2t-t_{\mathbf{M}}\right) \leq (2t+t_{\mathbf{M}})\, Y \leq 2t+t_{\mathbf{M}}. \end{cases} \tag{4}$$

We must prove that, when we set Y as Lemma 1, the number of (4) is less than $2^{t-1}/2t+1$. From the definiton of α_t, $2t\alpha_t = (t-1) - \log(2t+1)$. Therefore, $2^{2t\alpha_t} = 2^{t-1}/(2t+1)$. Thus, we must show the following, under Y, in Lemma 1:

$$\begin{cases} H\left(\dfrac{(2t+t_{\mathbf{M}})\,Y}{2t}\right) \leq \alpha_t & \text{if } \ 0 \leq (2t+t_{\mathbf{M}})\, Y \leq \min\left(t_{\mathbf{M}}, 2t-t_{\mathbf{M}}\right) \\ H\left(\dfrac{t_{\mathbf{M}}}{2t}\right) \leq \alpha_t & \text{if } \ \min\left(t_{\mathbf{M}}, 2t-t_{\mathbf{M}}\right) \leq (2t+t_{\mathbf{M}})\, Y \leq 2t+t_{\mathbf{M}}. \end{cases} \tag{5}$$

Further, we show this is satisfied for all $0 \leq t_{\mathbf{M}} \leq 2t$, under Y, in Lemma 1.

First, when $0 \leq t_{\mathbf{M}} \leq 2t\beta_t$ or $2t\,(1-\beta_t) \leq t_{\mathbf{M}} \leq 2t$, then $0 \leq H(t_{\mathbf{M}}/2t) \leq \alpha_t$. Thus, in $0 \leq (2t+t_{\mathbf{M}})\,Y \leq \min\,(t_{\mathbf{M}}, 2t-t_{\mathbf{M}})$,

$$H\left(\frac{(2t+t_{\mathbf{M}})\,Y}{2t}\right) \leq H\left(\frac{t_{\mathbf{M}}}{2t}\right) \leq \alpha_t.$$

Therefore, the first inequality in (5) is satisfied. The second inequality in (5) is also satisfied obviously.

Next, we consider in the case, where $2t\beta_t < t_{\mathbf{M}} < 2t\,(1-\beta_t)$. In Lemma 1, when $Y \leq 2t\beta_t/\,(2t+t_{\mathbf{M}})$, we insist that the number of leaves after pruning, is less than $2^{t-1}/(2t+1)$. Now, we show that, when $Y \leq 2t\beta_t/\,(2t+t_{\mathbf{M}})$, (5) is satisfied.

When $Y \leq 2t\beta_t/\,(2t+t_{\mathbf{M}})$, for all $0 \leq x \leq Y$,

$$H\left(\frac{(2t+t_{\mathbf{M}})\,x}{2t}\right) \leq H\,(\beta_t) = \alpha_t.$$

However, if there are Y that satisfy $(2t+t_{\mathbf{M}})\,Y > \min\,(t_{\mathbf{M}}, 2t-t_{\mathbf{M}})$, there are x in $[0, Y]$ such that

$$H\left(\frac{(2t+t_{\mathbf{M}})\,x}{2t}\right) > H\left(\frac{t_{\mathbf{M}}}{2t}\right) > H\,(\beta_t) = \alpha_t.$$

Therefore, if there are Y that satisfy $(2t+t_{\mathbf{M}})\,Y > \min\,(t_{\mathbf{M}}, 2t-t_{\mathbf{M}})$, contradiction occurs. Therefore, Y satisfies $(2t+t_{\mathbf{M}})\,Y \leq \min\,(t_{\mathbf{M}}, 2t-t_{\mathbf{M}})$. Thus, we consider only the first inequality in (5), which is satisfied obviously. Therefore, we show (5) in all $t_{\mathbf{M}}$, we prove Lemma 1. □

5 Numerical Experiments with the Proposed Algorithm

We implemented our proposed Algorithm 3 in C++ using NTL library version 10.3.0. Our tests were run on an Intel Core i7, at 2.40 GHz with 16-GB memory. To render the proposed Algorithm 3 more efficient, we used following techniques. We implemented the proposed Algorithm 3 for the first depth search. We pruned the leaves containing $4tY$ errors, before calculating t slices completely. This is because the errors monotonically increase, during the calculation of the slices.

In the experiment, we tested the proposed Algorithm 3 on a 1024-bit CRT-RSA. We generated 100 random keys. In each key, we added noise in the square-and-multiply sequences as $\delta = 0.011$ observed in [2]. We then set parameters t, Y, in Algorithm 3. In this experiment, we set $t = 40, 60, 80, 100$ for $Y = 0.03$, $t = 40, 60$ for $Y = 0.04$. We executed our proposed Algorithm 3 for each square-and-multiply sequence with error. When the correct secret keys were output, the proposed Algorithm 3 was considered successful. We measured the success rate, average time for all the trials, and average time for successful trials for 100 pairs of square-and-multiply sequences with noise, when given the correct k_p and k_q. The experimental results are listed in Tables 1 and 2.

Table 1. Experimental data under $\delta = 0.011, Y = 0.03$ for correct k_p and k_q

	t			
	40	60	80	100
Average time for all the trials (ms)	22.2	237	1530	17973
Average time for successful trials (ms)	25.2	234	1537	17188
Success Rate (%)	67	93	98	97

Table 2. Experimental data under $\delta = 0.011, Y = 0.04$ for correct k_p and k_q

	t	
	40	60
Average time for all the trials (ms)	124	2287
Average time for successful trials (ms)	130	2287
Success Rate (%)	88	100

From Tables 1 and 2, it can be established that the secret keys were recovered with a high success rate. In Table 1, when we set $t = 40$, we recovered 67 % of the secret keys in 25.2 ms; thus, we recovered 2/3 of the secret keys in 30 ms. Moreover, when we set a larger value of t, the running time was more and the success rate was higher. For example, when we set $t = 60$, we recovered 93 % of the secret keys in 0.24 s. For $t = 80, 100$, almost all the secret keys were recovered; however, the success rate was not 100 %.

Thus, we set the larger value of Y as $Y = 0.04$, for achieving a success rate of 100 %. In Table 2, when we set $t = 40$, we recovered 88 % of the secret keys in 0.13 s. When we set $t = 60$, we recovered all the secret keys in 2.3 s.

In addition to these experiments, we executed Algorithm 3 for $\delta = 0.011, t = 40, Y = 0.05$. In this experiment, we recovered 99 % of the secret keys in 1.1 s. Thus, for $\delta = 0.011$, almost all the secret keys were recovered in approximately 1–2 s using our proposed Algorithm 3.

6 Conclusion

In this paper, we presented a key recovery algorithm, using the square-and-multiply sequences on exponentiations c^{d_p} and c^{d_q} with small errors, at $w = 1$, namely, the binary method.

We theoretically proved that we can correct square-and-multiply sequences with error rates less than 5.8 % in the expected polynomial time, when c^{d_p} and c^{d_q} are observed only once. In addition, we experimentally demonstrated that our proposed Algorithm 3 recovers small errors of 1.1 %, in 1–2 s under correct (k_p, k_q).

A Appendix: Proof of Lemma 2

In this section, initially, we prepare the tools used for proving Lemma 2, after which we prove the Lemma 2.

A.1 Tools used for proving Lemma 2

In order to prove Lemma 2, we define certain sets and prove Lemma 4. First, we define the set of square-and-multiply sequences.

Definition 1. Let $L \in \mathbb{N}$. $A(L)$ is defined as the set of all square-and-multiply sequences with length L.

$A(L)$ includes elements that cannot be converted into bits. The elements in set $A(L)$, merely display $\mathbf{S}$ and $\mathbf{M}$. Therefore, the number of elements in set $A(L)$, is 2^L. In the following discussion, we refer to operations in sequence as the first operation, second operation, etc., from the left operation.

We then define the square-and-multiply sequences that can be converted into bits.

Definition 2. Let $L \in \mathbb{N}$ and $t_{\mathbf{M}}$ be non-negative integers. $T(L, \mathbf{S}, t_{\mathbf{M}})$ is defined as the set of square-and-multiply sequences that satisfy the following conditions: the length of the sequence is L, the first operation of the sequence is $\mathbf{S}$, the number of $\mathbf{M}$s is $t_{\mathbf{M}}$, and the $\mathbf{M}$s are not consecutive. Similarly, $T(L, \mathbf{M}, t_{\mathbf{M}})$ is defined as the set of square-and-multiply sequence that satisfies the following conditions: the length of the sequences is L, the first operation of the sequence is $\mathbf{M}$, the number of $\mathbf{M}$s is $t_{\mathbf{M}}$, and the $\mathbf{M}$s are not consecutive.

We now calculate the number of elements in $T(L, \mathbf{S}, t_{\mathbf{M}})$, used in our analysis. The element of $T(L, \mathbf{S}, t_{\mathbf{M}})$ that satisfies the first operation is $\mathbf{S}$, and the $\mathbf{M}$s are not consecutive. Moreover, the number of $\mathbf{S}$s is $L - t_{\mathbf{M}}$ and the number of $\mathbf{M}$s is $t_{\mathbf{M}}$. Thus, the elements of $T(L, \mathbf{S}, t_{\mathbf{M}})$ are generated from $L - t_{\mathbf{M}}$ bits, including $t_{\mathbf{M}}$ one bits. Therefore,

$$|T(L, \mathbf{S}, t_{\mathbf{M}})| = \begin{cases} \binom{L - t_{\mathbf{M}}}{t_{\mathbf{M}}} & 0 \leq t_{\mathbf{M}} \leq \lfloor L/2 \rfloor \\ 0 & \text{otherwise.} \end{cases}$$

Next, we consider a situation, where an observed square-and-multiply sequence is $\tilde{o}$. We define the set of square-and-multiply sequences at a certain distance from $\tilde{o}$.

Definition 3. Let $L \in \mathbb{N}$, $t_{\mathbf{M}}, C$ be non-negative integers, and $\tilde{o} \in A(L)$. Then, $B(L, \mathbf{S}, t_{\mathbf{M}}, C, \tilde{o})$ is defined as the set of square-and-multiply sequences in $T(L, \mathbf{S}, t_{\mathbf{M}})$ satisfying the condition that the number of disagreements with $\tilde{o}$, excluding the first operation, is less than C. In addition, $B(L, \mathbf{M}, t_{\mathbf{M}}, C, \tilde{o})$ is defined as the set of square-and-multiply sequences in $T(L, \mathbf{M}, t_{\mathbf{M}})$ satisfying the condition that the number of disagreements with $\tilde{o}$, excluding the first operation, is less than C.

By definition 3, for all $\tilde{o} \in A(L)$,

$$|B(L, \mathbf{S}, t_{\mathbf{M}}, C-1, \tilde{o})| \leq |B(L, \mathbf{S}, t_{\mathbf{M}}, C, \tilde{o})|. \tag{6}$$

Finally, we define the upper bound of the number of elements in set B.

Definition 4. Let $L \in \mathbb{N}$, k, C be non-negative integers. Then, $\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right)$ is defined as $\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right) = \max_{\tilde{o} \in A(L)} \left|B\left(L, \mathbf{S}, t_{\mathbf{M}}, C, \tilde{o}\right)\right|$.

From (6),

$$\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C-1\right) \leq \tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right). \tag{7}$$

In order to prove Lemma 2, we present the upper bound of $\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right)$ in Lemma 4.

Lemma 4. Let $L \in \mathbb{N}$, $t_{\mathbf{M}}$ be an integer satisfying $0 \leq t_{\mathbf{M}} \leq \lfloor L/2 \rfloor$, and C be an integer satisfying $0 \leq C \leq L-1$. Then,

$$\begin{cases} \tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{L - t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq \min\left(t_{\mathbf{M}}, L - 2t_{\mathbf{M}}\right) \\ \tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{L - t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ \min\left(t_{\mathbf{M}}, L - 2t_{\mathbf{M}}\right) \leq C \leq L-1 \end{cases}$$

is satisfied.

Proof. We prove Lemma 4 by mathematical induction. The recurrence formula on $\tilde{b}$ is following. When $L \geq 3$,

$$\begin{aligned} &\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right) \\ &\leq \max\Big(\tilde{b}\left(L-1, \mathbf{S}, t_{\mathbf{M}}, C\right) + \tilde{b}\left(L-2, \mathbf{S}, t_{\mathbf{M}}-1, C-1\right), \\ &\tilde{b}\left(L-1, \mathbf{S}, t_{\mathbf{M}}, C-1\right) + \tilde{b}\left(L-2, \mathbf{S}, t_{\mathbf{M}}-1, C\right)\Big). \end{aligned} \tag{8}$$

We now consider the case where $L = 1$ and $L = 2$. For $L = 1$ and $L = 2$, because $\left|T\left(1, \mathbf{S}, 0\right)\right| = 1$ and $\left|T\left(2, \mathbf{S}, 0\right)\right| = \left|T\left(2, \mathbf{S}, 1\right)\right| = 1$, $\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right)$ is always less than one. Therefore, when $L = 1$ and $L = 2$, Lemma 4 is true.

We now consider the case, where Lemma 4 is true, when $1 \leq L \leq m (m \in \{x \in \mathbb{N} | x \geq 2\})$. We prove that Lemma 4 is true, when $L = m+1$. We consider the set $T\left(m+1, \mathbf{S}, t_{\mathbf{M}}\right)$. The condition $T\left(m+1, \mathbf{S}, t_{\mathbf{M}}\right) \neq \emptyset$ is $0 \leq t_{\mathbf{M}} \leq \lfloor (m+1)/2 \rfloor$, and the possible value of C is $0 \leq C \leq m$. For these $(t_{\mathbf{M}}, C)$, we prove

$$\begin{cases} \tilde{b}\left(m+1, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{m+1 - t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq X \\ \tilde{b}\left(m+1, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{m+1 - t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ X \leq C \leq m \end{cases}$$

under $X = \min\left(t_{\mathbf{M}}, m+1-2t_{\mathbf{M}}\right)$.

First, we consider the case, where $C = 0$. $C = 0$ indicates that no mismatch exists between the sequence and the observed data; thus, the number of the sequences that satisfy $C = 0$ is no more than one. Thus, $\tilde{b}\left(m+1, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq 1$.

We then consider the case, where $1 \leq C \leq t_{\mathbf{M}} - 1$. From the assumption by mathematical induction, Lemma 4 is true, when $L = m, m-1$. When $L = m$,

$$\begin{cases} \tilde{b}\left(m, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{m - t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq \min\left(t_{\mathbf{M}}, m - 2t_{\mathbf{M}}\right) \\ \tilde{b}\left(m, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{m - t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ \min\left(t_{\mathbf{M}}, m - 2t_{\mathbf{M}}\right) \leq C \leq m-1 \end{cases}$$

is satisfied. Thus, $\binom{m-t_{\mathbf{M}}}{t_{\mathbf{M}}} \leq \binom{m-t_{\mathbf{M}}}{C}$ under $\min(t_{\mathbf{M}}, m-2t_{\mathbf{M}}) \leq C \leq t_{\mathbf{M}}$. Therefore, $\tilde{b}(m, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m-t_{\mathbf{M}}}{C}$ under $0 \leq C \leq t_{\mathbf{M}}$. Similarly, when $L = m-1$,

$$\begin{cases} \tilde{b}(m-1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m-1-t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq \min(t_{\mathbf{M}}, m-1-2t_{\mathbf{M}}) \\ \tilde{b}(m-1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m-1-t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ \min(t_{\mathbf{M}}, m-1-2t_{\mathbf{M}}) \leq C \leq m-2 \end{cases}$$

is satisfied. Thus, $\binom{m-1-t_{\mathbf{M}}}{t_{\mathbf{M}}} \leq \binom{m-1-t_{\mathbf{M}}}{C}$ under $\min(t_{\mathbf{M}}, m-1-2t_{\mathbf{M}}) \leq C \leq t_{\mathbf{M}}$. Therefore, $\tilde{b}(m-1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m-1-t_{\mathbf{M}}}{C}$ under $0 \leq C \leq t_{\mathbf{M}}$.

Thus, under $1 \leq C \leq t_{\mathbf{M}} - 1$,

$$\begin{aligned} &\tilde{b}(m, \mathbf{S}, t_{\mathbf{M}}, C) + \tilde{b}(m-1, \mathbf{S}, t_{\mathbf{M}}-1, C-1) \\ &\leq \binom{m-t_{\mathbf{M}}}{C} + \binom{(m-1)-(t_{\mathbf{M}}-1)}{C-1} = \binom{m+1-t_{\mathbf{M}}}{C} \end{aligned}$$

and

$$\begin{aligned} &\tilde{b}(m, \mathbf{S}, t_{\mathbf{M}}, C-1) + \tilde{b}(m-1, \mathbf{S}, t_{\mathbf{M}}-1, C) \\ &\leq \binom{m-t_{\mathbf{M}}}{C-1} + \binom{(m-1)-(t_{\mathbf{M}}-1)}{C} = \binom{m+1-t_{\mathbf{M}}}{C}. \end{aligned}$$

Therefore, from (8), under $1 \leq C \leq t_{\mathbf{M}} - 1$,

$$\tilde{b}(m+1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m+1-t_{\mathbf{M}}}{C}.$$

However, $|T(m+1, \mathbf{S}, t_{\mathbf{M}})| = \binom{m+1-t_{\mathbf{M}}}{t_{\mathbf{M}}}$. Thus, under $1 \leq C \leq t_{\mathbf{M}} - 1$,

$$\tilde{b}(m+1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \min\left(\binom{m+1-t_{\mathbf{M}}}{C}, \binom{m+1-t_{\mathbf{M}}}{t_{\mathbf{M}}}\right).$$

Therefore, under $X = \min(t_{\mathbf{M}}, m+1-2t_{\mathbf{M}})$,

$$\begin{cases} \tilde{b}(m+1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m+1-t_{\mathbf{M}}}{C} & \text{if } \ 1 \leq C \leq X \\ \tilde{b}(m+1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m+1-t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ X \leq C \leq t_{\mathbf{M}} - 1 \end{cases}$$

is satisfied.

Finally, we consider the case, where $t_{\mathbf{M}} \leq C \leq m$. Because $|T(m+1, \mathbf{S}, t_{\mathbf{M}})| = \binom{m+1-t_{\mathbf{M}}}{t_{\mathbf{M}}}$, $\tilde{b}(m+1, \mathbf{S}, t_{\mathbf{M}}, C) \leq \binom{m+1-t_{\mathbf{M}}}{t_{\mathbf{M}}}$ under $t_{\mathbf{M}} \leq C \leq m$.

In conclusion,

$$\begin{cases} \tilde{b}\left(m+1, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{m+1-t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq X \\ \tilde{b}\left(m+1, \mathbf{S}, t_{\mathbf{M}}, C\right) \leq \binom{m+1-t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ X \leq C \leq m \end{cases}$$

under $X = \min\left(t_{\mathbf{M}}, m+1-2t_{\mathbf{M}}\right)$. Therefore, when $L = m+1$, Lemma 4 is true. In conclusion, Lemma 4 is proved. □

A.2 Proof of Lemma 2

In Lemma 4, the upper bound of $\tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right)$ is calculated. This function does not consider the first operation of the sequence. We now consider the first operation of the sequence. Let $\tilde{o}$ be a partially observed sequence, and let its length be L. We calculate the upper bound of the number of elements $\tilde{u}(L, t_{\mathbf{M}}, C, \tilde{o})$ in $T\left(L, \mathbf{S}, t_{\mathbf{M}}\right)$, such that the number of disagreements with $\tilde{o}$ is less than C.

First, we consider the case, where $0 \leq C \leq L-1$. From Lemma 4, if the first operation of $\tilde{o}$ is $\mathbf{S}$, then $\tilde{u}(L, t_{\mathbf{M}}, C, \tilde{o}) \leq \tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right)$, and if the first operation of $\tilde{o}$ is $\mathbf{M}$, then $\tilde{u}(L, t_{\mathbf{M}}, C, \tilde{o}) \leq \tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C-1\right)$. Because of (7), $\tilde{u}\left(L, t_{\mathbf{M}}, C, \tilde{o}\right) \leq \tilde{b}\left(L, \mathbf{S}, t_{\mathbf{M}}, C\right)$. From Lemma 4, $\tilde{u}\left(L, t_{\mathbf{M}}, C, \tilde{o}\right)$ is less than

$$\begin{cases} \binom{L-t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq \min\left(t_{\mathbf{M}}, L-2t_{\mathbf{M}}\right) \\ \binom{L-t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ \min\left(t_{\mathbf{M}}, L-2t_{\mathbf{M}}\right) \leq C \leq L-1 \end{cases}$$

We next consider the case, where $C = L$. Because of $\left|T\left(L, \mathbf{S}, t_{\mathbf{M}}\right)\right| = \binom{L-t_{\mathbf{M}}}{t_{\mathbf{M}}}$, $\tilde{u}\left(L, t_{\mathbf{M}}, C, \tilde{o}\right)$ is less than $\binom{L-t_{\mathbf{M}}}{t_{\mathbf{M}}}$. Therefore, the number of sequences, whose length is L, including $t_{\mathbf{M}}$ $\mathbf{M}$s, generated from the bits, and whose number of disagreements is less than C, compared to the observed data is less than

$$\begin{cases} \binom{L-t_{\mathbf{M}}}{C} & \text{if } \ 0 \leq C \leq \min\left(t_{\mathbf{M}}, L-2t_{\mathbf{M}}\right) \\ \binom{L-t_{\mathbf{M}}}{t_{\mathbf{M}}} & \text{if } \ \min\left(t_{\mathbf{M}}, L-2t_{\mathbf{M}}\right) \leq C \leq L \end{cases}$$

□

Acknowledgements.

This research was partially supported by JST CREST Grant Number JPMJCR14D6, Japan and JSPS KAKENHI Grant Number 16H02780.

References

1. Bauer, S.: Attacking Exponent Blinding in RSA without CRT. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 82–88. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29912-4_7
2. Bernstein, D.J., Breitner, J., Genkin, D., Bruinderink, L.G., Heninger, N., Lange, T., van Vredendaal, C., Yarom, Y.: Sliding Right into Disaster: Left-to-Right Sliding Windows Leak. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 555–576. Springer, Cham (2017). doi:10.1007/978-3-319-66787-4_27
3. Bruen, A.A., Forcinito, M.A.: Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century. John Wiley & Sons, Inc., Hoboken, New Jersey (2005)
4. Coppersmith, D.: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. Journal of Cryptography. **10**, 233–260 (1997). doi:10.1007/s001459900030
5. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E.: Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 207–228. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48324-4_11
6. Genkin, D., Shamir, A., Tromer, E.: RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 444–461. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_25
7. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest We Remember: Cold-Boot Attacks on Encryption Keys. Commun. ACM. **52**, 91–98 (2009). doi:10.1145/1506409.1506429
8. Henecka, W., May, A., Meurer, A.: Correcting Errors in RSA Private Keys. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 351–369. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_19
9. Heninger, N., Shacham, H.: Reconstructing RSA Private Keys from Random Key Bits. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 1–17. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_1
10. Hoeffding, W.: Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association. **58**, 13–30 (1963). doi:10.1080/01621459.1963.10500830
11. Homma, N., Miyamoto, A., Aoki, T., Satoh, A., Shamir, A.: Comparative Power Analysis of Modular Exponentiation Algorithms. IEEE Trans. Computers. **59**, 795–807 (2010). doi:10.1109/TC.2009.176
12. İnci, M.S., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T., Suner, B.: Cache Attacks Enable Bulk Key Recovery on the Cloud. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 368–388. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53140-2_18
13. Jagnere, P., Sanket, S., Chauhan, A., Jaiswal, R.: Better Algorithms for MSB-side RSA Reconstruction. WCC2015 (2015)
14. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_9
15. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_25

16. Kunihiro, N., Honda, J.: RSA Meets DPA: Recovering RSA Secret Keys from Noisy Analog Data. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 261–278, Springer, Heidelberg (2014). doi:10.1007/978-3-662-44709-3_15
17. Kunihiro, N., Shinohara, N., Izu, T.: Recovering RSA Secret Keys from Noisy Key Bits with Erasures and Errors. IEICE Trans. Fundamentals. **E97-A**, 1273–1284 (2014). doi:10.1587/transfun.E97.A.1273
18. Kunihiro, N., Takahashi, Y.: Improved Key Recovery Algorithms from Noisy RSA Secret Keys with Analog Noise. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 328-343. Springer, Cham (2017). doi:10.1007/978-3-319-52153-4_19
19. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring Polynomials with Rational Coefficients. Mathematische Annalen. **261**, 515–534 (1982). doi:10.1007/BF01457454
20. Maitra, S., Sarkar, S., Gupta, S.S.: Factoring RSA Modulus Using Prime Reconstruction from Random Known Bits. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 82–99. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12678-9_6
21. Menezes, A.J.,van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography, CRC Press, Boca Raton, Florida (1996)
22. Paterson, K.G., Polychroniadou, A., Sibborn, D.L.: A Coding-Theoretic Approach to Recovering Noisy RSA Keys. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 386–403. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_24
23. Percival, C.: Cache Missing for Fun and Profit. `http://www.daemonology.net/papers/htt.pdf` (2005)
24. Rivest, R. L., Shamir, A.: Efficient Factoring Based on Partial Information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986). doi:10.1007/3-540-39805-8_3
25. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM. **21**, pp. 120–126 (1978). doi:10.1145/359340.359342
26. RSA Laboratories: PKCS#1 v2.2: RSA Cryptography Standard. `https://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf` (2012)
27. Sarkar, S., Gupta, S.S., Maitra, S.: Reconstruction and Error Correction of RSA Secret Parameters from the MSB Side. WCC2011 (2011)
28. Sarkar, S., Gupta, S.S., Maitra, S.: Error Correction of Partially Exposed RSA Private Keys from MSB Side. In: Bagchi, A., Ray, I. (eds.) ICISS 2013. LNCS, vol. 8303, pp. 345–359. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45204-8_26
29. Takayasu, A., Kunihiro, N.: A Tool Kit for Partial Key Exposure Attacks on RSA. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 58–73. Springer, Cham (2017). doi:10.1007/978-3-319-52153-4_4
30. Yarom, Y., Falkner, K.: FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In: USENIX 2014, pp. 719–732 (2014).
31. Yarom, Y., Genkin, D., Heninger, N.: CacheBleed: A Timing Attack on OpenSSL Constant Time RSA. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 346–367. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53140-2_17