Recovering RSA Secret Keys from Noisy Key Bits with Erasures and Errors

Noboru Kunihiro[1], Naoyuki Shinohara[2], and Tetsuya Izu[3]

[1] The University of Tokyo, Japan
kunihiro@k.u-tokyo.ac.jp
[2] NICT, Japan
[3] Fujitsu Labs, Japan

Abstract. We discuss how to recover RSA secret keys from noisy key bits with erasures and errors. There are two known algorithms recovering original secret keys from noisy keys. At Crypto 2009, Heninger and Shacham proposed a method for the case where an erroneous version of secret keys contains only erasures. Subsequently, Henecka et al. proposed a method for an erroneous version containing only errors at Crypto 2010. For physical attacks such as side-channel and cold boot attacks, we need to study key recovery from a noisy secret key containing both erasures and errors. In this paper, we propose a method to recover a secret key from such an erroneous version and analyze the condition for error and erasure rates so that our algorithm succeeds in finding the correct secret key in polynomial time. We also evaluate a theoretical bound to recover the secret key and discuss to what extent our algorithm achieves this bound.

Keywords: RSA, Key-recovery, Cold Boot Attack, Side-channel Attack, Maximal Likelihood

1 Introduction

1.1 Background

RSA [12] is a widely used cryptosystem. In RSA a public modulus N is chosen to be a product of two distinct primes p and q. The key-pair $e, d \in \mathbb{Z}$ satisfies $ed \equiv 1 \pmod{(p-1)(q-1)}$. The encryption keys are (N, e) and the decryption keys are (N, d). The PKCS#1 standard [10] specifies that the RSA secret key includes the following information: $(p, q, d, d_p, d_q, q^{-1} \bmod p)$ in addition to d, which allows a fast decryption process using the Chinese Remainder Theorem.

Secret keys must be kept secret. Nevertheless, some fractional amounts of the secret information can be leaked by physical attacks such as side-channel and cold boot attacks [4]. If the amount of leaked bits for secret keys is quite small, it is impossible to recover the secret keys from the leaked information. Conversely, it might be possible to recover them by using their redundancy if a certain amount of bits are leaked. Note that all bits are not necessarily leaked.

For example, Coppersmith [2] showed that RSA can be broken if the upper half of the secret key p is revealed. Herrmann and May [7] showed that RSA can be broken (in exponential time) if at least 70% of the bits for a prime factor p of N are leaked. Their methods are based on the lattice reduction technique. Note that the Herrmann-May method does not require that the leaked bits are consecutive.

At Crypto 2009, Heninger and Shacham [6] proposed an algorithm that efficiently recovers secret keys (p, q, d, d_p, d_q) given a random fraction of their bits. Concretely, they showed that if at least 27% of the secret key bits are leaked at random, the full secret keys can be recovered. Conversely, we can say that even if 73% of original secret bits are erased, the key recovery succeeds.

As opposed to the Heninger-Shacham algorithm correcting erasures, Henecka et al. [5] proposed an algorithm correcting error bits of secret keys at Crypto 2010. They showed that the secret key (p, q, d, d_p, d_q) can be fully recovered if the error probability is less than 0.237. They also showed the bound for the error probability is given by 0.084 if the involved secret key is (p, q).

Independently of our work, Paterson et al. proposed an algorithm correcting error bits which asymmetrically occurs at Asiacrypt 2012 [9]. Their algorithm works in a true cold boot setting. They took a coding theoretic approach for designing a new algorithm and analyzing its performance.

1.2 Motivation: Attack Scenario

All existing works concerning key recovery from noisy secret keys have discussed the erasure-only (error-free) case or error-only (erasure-free) case. This paper deals with the key recovery for a noisy secret key with both erasures and errors. We call the erroneous version of the secret key with both erasures and errors *noisy secret keys*. We denote the correct secret key by $\mathbf{sk}$, and the noisy secret key corresponding to $\mathbf{sk}$ by $\overline{\mathbf{sk}}$. Before discussing the details, we address the motivations of this study.

Cold Boot Attack Scenario: Under the cold boot attack scenario [4], (the degraded version of) secret keys are observed with (almost) unidirectional bit flipping. Assume that the flip of each bit occurs as completely unidirectional. For simplicity, we assume that only the bit flipping of $1 \to 0$ occurs. If the observed bit is 1, the corresponding bit of the correct secret key is definitely 1. In contract, if the observed bit is 0, we cannot determine whether the corresponding bit is 0 or 1. Therefore, the observed bit 0 can be considered *erasure*. Heninger and Shacham [6] proposed an efficient algorithm that recovers the secret key from the degraded version of the secret key with erasure. However, as Heninger and Shacham [6] pointed out, the bit flip with an opposite direction occurs with small but non-zero probability. If the observed bit sequence contains errors, Heninger-Shacham's algorithm can never recover the correct secret key. This algorithm is then no longer applicable for the noisy secret key containing both erasures and errors.

Side-channel Attack Scenario: Henecka et al. [5] proposed an efficient algorithm given a noisy secret key only with errors. The noisy keys are often

provided through a side-channel attack. Under some attack situations, each bit is provided with additional information: so-called *reliability*. Consider the following situation: some bits of secret keys are 0 (or 1) with very high reliability and others are 0 (or 1) with not so high reliability. One reasonable strategy is to set a bit value as the observed bit if its reliability is sufficiently high. How should we set a bit value with low reliability? We have two potential strategies. The first is to set a bit value as the observed bit, which will cause a high number of bit errors. The second strategy is to regard the bit as an *erasure* bit, which will involve the observed secret key with (fewer) errors and erasures. So then which of strategies is good for attackers? As Henecka et al. pointed out, the correction of errors seems to be a much more difficult problem than the correction problem. We therefore expect that the second strategy leads to a better algorithm. However, their algorithm is not applicable to a noisy secret key containing both erasures and errors.

For both cases, studies for the key recovery for noisy secret keys with both errors and erasures are important to maximize and evaluate the potential threat of physical attacks and to consider the possible countermeasures against them.

1.3 Our Contributions

This paper discusses secret key recovery from noisy secret key sequences with both errors and erasures. First, we present a polynomial time algorithm for recovering secret keys and show an explicit success condition for recovering the keys. We denote the erasure probability by δ and error probability by ϵ. We also denote by m the number of involved secret keys. For example, $m = 5$ if $\mathbf{sk} = (p, q, d, d_p, d_q)$ is involved. Our algorithm can asymptotically recover secret keys in polynomial time with high probability provided that

$$1 - \delta - 2\epsilon \geq \sqrt{\frac{2(1-\delta)\ln 2}{m}},$$

where we denote the natural logarithm of n to the base e by $\ln n$. In special case, our algorithm also includes previous methods. In fact, our algorithm achieves the upper bound of Heninger-Shacham [6] and that of Henecka et al. [5] for the error-free case ($\epsilon = 0$) and erasure-free case ($\delta = 0$), respectively. We ran experiments to verify our analysis. We achieved to the error rates of up to 0.6 and the erasure rate $\epsilon = 0.01$ for 1024-bit RSA with high success probability.

Second, we derive a theoretical bound for recovering the secret keys from the noisy secret keys. We first introduce a natural abstract algorithm (`meta-algorithm`) and derive a condition for δ and ϵ such that it needs exponential time for recovering keys. The binary Entropy function $H(x)$ [3] is defined by by $-x \log x - (1-x)\log(1-x)$, where $\log n$ is the binary logarithm of n to the base 2. Then, we prove that we cannot recover the secret keys in polynomial time under our `meta-algorithm` if it holds that

$$(1-\delta)\left(1 - H\left(\frac{\epsilon}{1-\delta}\right)\right) < \frac{1}{m}.$$

Finally, we discuss the relation between the condition where our algorithm can recover secret keys and the theoretical bound. We first see that there exists a small gap between the success condition and the theoretical bound. Then, we show that the proposed algorithm achieves the second order expansion of the theoretical bound.

2 Preliminaries

This section presents an overview of methods using binary trees to recover the secret key of the RSA cryptosystem [12]. In particular, we briefly explain two known methods: Heninger-Shacham method [6] and the method of Henecka et al. (abbreviated to HMM method) [5].

We use similar notations as [5]. For an n-bit sequence $\mathbf{x} = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$, we denote the i-th bit of $\mathbf{x}$ by $x[i] = x_i$, where $x[0]$ is the least significant bit of $\mathbf{x}$. Let $\tau(M)$ denote the largest exponent such that $2^{\tau(M)} | M$. As well as [5], Hoeffding's bound [8] is the main tool in our analysis.

Theorem 1 (Hoeffding's Bound). *Let $X_1, \ldots, X_k$ be a sequence of independent Bernoulli trials with identical success probability $\Pr[X_i = 1] = p$ for all i. Define $X := \sum_{i=1}^{k} X_i$. Then, for every $0 < \gamma < 1$ we have $\Pr[X \geq k(p+\gamma)] \leq \exp(-2k\gamma^2)$ and $\Pr[X \leq k(p-\gamma)] \leq \exp(-2k\gamma^2)$.*

2.1 Noise Models

We formalize (three) noise models discussed in this paper. Let ϵ and δ be real numbers satisfying $0 \leq \epsilon < 1/2$, $0 \leq \delta < 1$ and $0 \leq \epsilon + \delta < 1$. In our noise models, each bit in a secret bit sequence is either erased with probability δ or flipped with probability ϵ, or remains unchanged with probability $1-\delta-\epsilon$. Then, only the transformed sequence is observed. Nevertheless, the original sequence is not directly obtained. We refer to this noise model as the Binary Erasure-Error model (BEE model). The error-free model, that is $\epsilon = 0$ (but, $\delta > 0$), is referred to as the Binary Erasure model (BE model) and erasure-free model, that is $\delta = 0$ (but, $\epsilon > 0$), is referred to as the Binary Symmetric model (BS model).

Our target in this paper is to recover the original secret key from the observed noisy sequence. We can say that Heninger and Shacham [6] have studied key recovery from the noisy keys in the BE model and Henecka et al. [5] have studied it in the BS model.

2.2 Recovering RSA Secret Key by Using Binary Trees

First, we review the key setting of the RSA cryptosystem [12], especially of the PKCS #1 standard [10]. The public key is (N, e) and the secret key is $\mathbf{sk} = (p, q, d, d_p, d_q, q^{-1} \bmod p)$. As in the previous works, we also ignore the last component $q^{-1} \bmod p$ in the secret key. The public and secret keys have the following relations:

$$N = pq,\ ed \equiv 1 \pmod{(p-1)(q-1)},\ ed_p \equiv 1 \pmod{p-1},\ ed_q \equiv 1 \pmod{q-1}.$$

From the key setting, there exist some integers k, k_p, k_q such that

$$N = pq, ed = 1 + k(p-1)(q-1), ed_p = 1 + k_p(p-1), ed_q = 1 + k_q(q-1). \quad (1)$$

Suppose that we know the exact values of k, k_p, and k_q. There exist five unknowns (p, q, d, d_p, d_q) in Eq. (1). Then, if we know just one of the exact values of unknowns, we can easily obtain the others.

The small public exponent e is usually used in practical applications [13], so we suppose that e is small enough such that $e = 2^{16} + 1$ in the same manner as [5, 6]. We need to find k, k_p, and k_q for small e. See the full version for how to compute k, k_p and k_q in our method.

In the Heninger-Shacham method [6], HMM method [5] and our new method, a secret key $\mathbf{sk}$ is recovered by using a binary-tree-based technique. Here we explain how to recover secret keys, taking $\mathbf{sk} = (p, q, d, d_p, d_q)$ as an example.

First we mention generating the tree. Since p and q are $n/2$ bit prime numbers and half of the most significant bit (MSB) of d is efficiently computable in the Heninger-Shacham or HMM methods [5, 6], there exist at most $2^{n/2}$ candidates for each secret key in $\{p, q, d, d_p, d_q\}$.

Heninger and Shacham [6] define the i-th bit slice for each bit index i and we denote by

$$\mathbf{slice}(i) := (p[i], q[i], d[i + \tau(k)], d_p[i + \tau(k_p)], d_q[i + \tau(k_q)]).$$

Assume that we have computed a partial solution $\mathbf{sk}' = (p', q', d', d'_p, d'_q)$ up to $\mathbf{slice}(i-1)$. Heninger and Shacham [6] applied Hensel's lemma to Eq. (1) and presented the following equations

$$p[i] + q[i] = (N - p'q')[i] \bmod 2, \quad (2)$$

$$d[i + \tau(k)] + p[i] + q[i] = (k(N+1) + 1 - k(p' + q') - ed')[i + \tau(k)] \bmod 2, \quad (3)$$

$$d_p[i + \tau(k_p)] + p[i] = (k_p(p' - 1) + 1 - ed'_p)[i + \tau(k_p)] \bmod 2, \quad (4)$$

$$d_q[i + \tau(k_q)] + q[i] = ((k_q(q' - 1) + 1 - ed'_q)[i + \tau(k_q)] \bmod 2. \quad (5)$$

We can easily see that $p[i], q[i], d[i+\tau(k)], d_p[i+\tau(k_p)]$, and $d_q[i+\tau(k_q)]$ are not independent and the degree of freedom is 1. Therefore, each Hensel lift yields exactly two candidate solutions. Then, the number of all candidates is given by $2^{n/2}$. The root node is given by $\mathbf{slice}(0) = (1, 1, d[\tau(k)], d_p[\tau(k_p)], d_q[\tau(k_q)])$.

Next we explain a pruning step, in which we count the number of matching bits between a bit sequence given by a node sequence and the corresponding bit sequence of a noisy secret key. We then discard or leave each node according to given criteria.

Section 2.3 briefly overviews the Heninger-Shacham method, which is for the case of an erroneous version $\overline{\mathbf{sk}}$ with an erasure rate δ of $\mathbf{sk}$. And in section 2.4 we mention the HMM method for an erroneous version $\overline{\mathbf{sk}}$ with error rate ϵ.

2.3 Heninger-Shacham Method [6]

In the Heninger-Shacham method, a binary tree is constructed by iterating an *expansion phase* and a *pruning phase*. At the pruning step, we compare a bit sequence given by one node with the corresponding bit sequence given by $\overline{\mathbf{sk}}$. Then we discard a node containing a bit not matching with the corresponding bit of $\overline{\mathbf{sk}}$, skipping the bit corresponding to an erasure bit of $\overline{\mathbf{sk}}$.

In the Heninger-Shacham method, discarded nodes are exactly wrong nodes, so the node corresponding to the correct solution consistently remains. Therefore, the success probability of the Heninger-Shacham method is 1. The computational cost of Heninger-Shacham method is evaluated with the number of remaining nodes of the binary tree, and depends on erasure rate δ. Therefore, Heninger and Shacham estimated the upper bound of δ such that the expected number of remaining nodes yielded from one wrong node is less than 1 under the following assumption:

Assumption 1 *The bit slice corresponding to a wrong node consists of random bits.*

This assumption is also used in the analysis of [5] and our new method.

Heninger and Shacham showed that their method recovers the secret keys provided that $\delta \leq 0.73$ if the noisy secret key is of the form (p, q, d, d_p, d_q), namely $m = 5$. If we use the noisy secret information (p, q), the secret key (p, q) can be obtained provided that $\delta \leq 0.43$. For general m of the involved secret information, the secret key can be recovered provided that $\delta \leq 2^{\frac{m-1}{m}} - 1$. We can see that the right-hand side of the above inequality is approximated by $1 - \frac{2 \ln 2}{m}$ for large m.

The Heninger-Shacham algorithm requires that the non-erasure bit is correct. However, this requirement is too idealistic in the physical attacks such as a cold boot attack, as described in Section 1.2. If the observed secret key contains an error, the Heninger-Shacham algorithm never finds the correct secret keys. We provide a simple example. Assuming that $\epsilon = 0.001$, we can regard the error rate as extremely small. Nevertheless, the number of errors in secret keys is expected to be $512 \times 5 \times 0.001 = 2.56 (> 2)$. Due to there being only two errors, the Heninger-Shacham algorithm does not work.

2.4 Henecka-May-Meurer Method [5]

We briefly explain the HMM method. For an erroneous version $\overline{\mathbf{sk}}$ with error rate ϵ, if we discard every node having a bit not matching the corresponding bit in $\overline{\mathbf{sk}}$, $\mathbf{sk}$ is never recovered since the leaf node corresponding to the correct solution does not remain. Therefore, the binary tree is separated into partial trees whose depth is t, and then the pruning step is performed for each partial tree. Actually, mt bits of the node sequence from the root node of the partial tree to the leaf node of the partial tree are compared with the corresponding bit of $\overline{\mathbf{sk}}$. If the number of matches is less than $C \in [0, mt]$, the leaf node is discarded. Since the remaining nodes of the binary tree decrease if the threshold value C increases, the

computational cost decreases and the success probability decreases. Especially the pruning step is not practically performed when $C = 0$, and we never obtain $\mathbf{sk}$ when $C = mt$. Henecka et al. considered the two following restrictions, which help to decide how to choose parameters (t, C). Note that $\mathbb{E}[X]$ is the mean of a random variable X.

Restriction 1 *Let $Z_{b,i}$ be the number of bad candidates generated from one bad partial solution at the i-th pruning step. Then, we choose parameters (t, C) so that $\mathbb{E}[Z_{b,i}] \leq 1/2$ holds.*

Restriction 2 *For each pruning step, we choose parameters (t, C) so that the probability that the correct node is discarded is less than $1/n$.*

The HMM method recovers the secret keys (p, q, d, d_p, d_q) if the error rate ϵ of the noisy keys is not larger than 0.237. If we use the noisy secret information or (p, q), the secret key (p, q) can be obtained, provided that $\epsilon \leq 0.084$. For general m of the involved secret information, the secret key can be recovered provided that $\epsilon \leq 1/2 - \sqrt{\ln 2/2m}$.

2.5 Naive Method Based on HMM Method

As mentioned, our main purpose is to recover secret keys from the noisy keys with both erasures and errors (that is, obtained through the BEE model). The following naive algorithm, which is not described in the literature, is sufficient for merely achieving this purpose.

Naive Method
Input: Public key (N, e), observed secret key $\overline{\mathbf{sk}}$, erasure probability δ and error probability ϵ
Output: Correct secret key $\mathbf{sk}$
Step 1: Transform $\overline{\mathbf{sk}}$ to $\overline{\mathbf{sk}}'$ by substituting random bits into erasure positions of $\overline{\mathbf{sk}}$.
Step 2: Perform the HMM method with the sequence $\overline{\mathbf{sk}}'$ and the error probability $\epsilon + \frac{\delta}{2}$ as inputs.

We evaluate the success condition of the algorithm. Each erasure bit will change a correct bit with probability 1/2 and a wrong bit with 1/2. The secret key sequence transformed in Step 1 can be considered a sequence with erasure probability 0 and error probability $\epsilon + \frac{\delta}{2}$. By applying the success condition for the BS model, we have the following condition for the naive method:

$$\epsilon + \delta/2 \leq 1/2 - \sqrt{\ln 2/(2m)}. \tag{6}$$

Although the algorithm does work for the noisy secret key for the BEE model, the above algorithm is not better than expected. There are some drawbacks to the naive method. Assuming that $\epsilon = 0$, the condition is described as $\delta \leq 1 - \sqrt{2 \ln 2/m}$. This condition is clearly worse than that of Heninger-Shacham:

$\delta \leq 1 - \frac{2\ln 2}{m}$. Next, we discuss the case where the error probability ϵ is very small but not zero, which is a natural situation in the cold boot attack scenario. For example, we assume that $m = 5$, $\delta = 0.6$ and $\epsilon = 0.001$. Considering that the Heninger-Shacham algorithm works well if $\delta = 0.73$ and $\epsilon = 0$, it is natural that we expect that the key recovery succeeds if $\delta = 0.6, \epsilon = 0.001$. However, the condition that $\delta = 0.6$ and $\epsilon = 0.001$ does not satisfy Eq. (6), and the naive method then cannot recover the secret key if $\delta = 0.6$ and $\epsilon = 0.001$. Our main goal in this paper is to propose a method that works in that case.

3 Recovering Secret Key from Noisy Secret Keys in BEE Model

Let $\overline{\mathbf{sk}}$ be an erroneous version of a secret key $\mathbf{sk}$ with erasure rate δ and error rate ϵ. The main purpose of our algorithm is to recover the original secret key from the observed $\overline{\mathbf{sk}}$ with the help of redundancy. We propose an algorithm to recover $\mathbf{sk}$ from $\overline{\mathbf{sk}}$ by using the binary-tree-based technique as in the Heninger-Shacham method [6] and HMM method [5]. Our algorithm is a combination of the two methods.

In our algorithm, the binary tree is separated into partial trees, and the pruning step is executed for every partial tree with threshold values as with the HMM method. Analysis of our algorithms then requires Assumption 1, Restrictions 1 and 2 in the same manner as with the HMM method.

Lesson Learned from Failure of Naive Method In the naive method described in section 2.5, we transform the erasure bit to the error bit with probability 1/2. This worsens the success condition. Any erasure bit should then be handled as erasure not error.

3.1 Our Proposed Method

In the HMM method [5], the noisy secret key sequence $\overline{\mathbf{sk}}$ is divided in an mt-bit subsequence to construct a partial tree, where t is a fixed integer. On the other hand, in our new method we divide the sequence in a T-bit subsequence skipping erasure bits in $\overline{\mathbf{sk}}$. We show a small example for $m = 3$ and $T = 4$. First, we explain how to divide bits for the i-th pruning step. Let E be the error symbol in $\overline{\mathbf{sk}}$. Suppose that we have divided bits until the bit p_s in the s-th node $[p_s, q_s, E]$ at the $(i-1)$-th pruning step, and the following nodes are given: $[p_s, q_s, E]$, $[p_{s+1}, E, d_{s+1}], [p_{s+2}, q_{s+2}, d_{s+2}]$. Then, since the i-th pruning step will be performed for T bits skipping bits corresponding to E in $\overline{\mathbf{sk}}$, we check the bits corresponding to $q_s, p_{s+1}, d_{s+1}, p_{s+2}$. Here we denote by t_i the length of a node sequence that is newly generated for the i-th pruning step, and denote by Δ_i the number of E in $\overline{\mathbf{sk}}$ at the i-th pruning step. In the example, $t_i = 2$ and $\Delta_i = 2$. Since the condition $T \geq m$ practically holds, we have that

$$t_i = \lceil (T + \Delta_i)/m \rceil \text{ or } \lceil (T + \Delta_i)/m \rceil - 1. \tag{7}$$

In the HMM method, only one threshold value C is used. In contrast, we use threshold values $C_1, \ldots, C_\ell$ when $\overline{\mathbf{sk}}$ is separated into ℓ intervals. Theorem 2 in Section 3.2 provides how to set each C_i. Note that unknown values of k, k_p and k_q are efficiently computable from $\overline{\mathbf{sk}}$. We show the details of how to compute them in the full version.

New method

Input: Public key (N, e), noisy secret key $\overline{\mathbf{sk}}$, error probability ϵ and erasure probability δ

Output: Correct secret key $\mathbf{sk}$.

Step 1: Compute k, k_p, k_q and $\mathbf{slice}(0)$.

Step 2: Compute $(T, C_1, \ldots, C_\ell)$.

Step 3: From $i = 1$ to ℓ, perform the following computation. Set $t_0 = 0$: Compute t_i slices: $\mathbf{slice}(1 + \sum_{j=0}^{i-1} t_j), \mathbf{slice}(2 + \sum_{j=0}^{i-1} t_j), \ldots, \mathbf{slice}(\sum_{j=0}^{i} t_j)$ and generate a partial tree whose depth is $t_i + 1$. For T bits skipping erasure bits $\overline{\mathbf{sk}}$, count the number of matches of bits in partial solutions with the corresponding bits in $\overline{\mathbf{sk}}$. If it is not less than C_i, then set $i = i + 1$ and go to the generating of a partial tree step. Otherwise, discard the node.

Step4: For each remaining leaf node, check whether the nodes are indeed the valid secret key with the help of public information.

Remark 1. Suppose that $\epsilon = 0$. Our method for $T = C = 1$ is equivalent to the Heninger-Shacham method [6]. Suppose that $\delta = 0$. Our method with $(T, C, C, \ldots, C)$ is equivalent to the HMM method [5] with $(T/m, C)$. Our method includes both of the two methods.

3.2 Analysis of Our Proposed Method

This section provides the analysis of our proposed method. The proofs of theorem and corollary in this section are given in Appendix A.

Theorem 2. *Suppose that Assumption 1 holds. Let (N, e) be an RSA public key with n-bit N and fixed e. We choose*

$$T = \left\lceil \frac{\ln n}{2\epsilon'^2} \right\rceil, \ \gamma_i = \sqrt{\frac{t_i + 1}{T} \frac{\ln 2}{2}}, \ C_i = T\left(\frac{1}{2} + \gamma_i\right), \tag{8}$$

where t_i and Δ_i are defined in section 3.1. Furthermore, let $\overline{\mathbf{sk}} = (\overline{\mathbf{sk}}_1, \ldots, \overline{\mathbf{sk}}_m)$ be an RSA secret key with noise rate ϵ such that

$$\frac{1}{2} + \gamma_i \leq 1 - \frac{(T + \Delta_i)\epsilon}{T} - \epsilon' \tag{9}$$

for every i. Then, Restrictions 1 and 2 hold for every fixed $\epsilon' > 0$. Our method also corrects $\overline{\mathbf{sk}}$ in expected time $\mathcal{O}(n^{2+2(\frac{\ln 2}{2m\epsilon'^2} + \frac{\Delta}{m}\frac{\ln 2}{\ln n})})$ with success probability at least $1 - \left(\frac{(1-\delta)m\epsilon'^2}{\ln n} + \frac{1}{n}\right)$, where $\Delta = \max\{\Delta_i\}$ and $\delta mn/2 = \sum \Delta_i$.

By Theorem 2, we have the corollary.

Corollary 1 *Suppose that Assumption 1 holds and that the number of erasure bits is Δ for each block. We choose*

$$T = \left\lceil \frac{\ln n}{2\epsilon'^2} \right\rceil,\ t = \frac{T+\Delta}{m},\ \gamma = \sqrt{\left(1+\frac{1}{t}\right)\frac{\ln 2}{2m}},\ C = T\left(\frac{1}{2}+\gamma\right).$$

If δ and ϵ satisfy

$$\epsilon + \frac{\delta}{2} \leq \frac{1}{2} - \sqrt{\left(1+\frac{1}{t}\right)\frac{(1-\delta)\ln 2}{2m}} - (1-\delta)\epsilon', \tag{10}$$

then our method satisfies Restrictions 1 and 2 for every fixed $\epsilon' > 0$. It also corrects $\overline{\mathbf{sk}}$ in expected time $\mathcal{O}(n^{2+2(\frac{\ln 2}{2m\epsilon'^2}+\delta t\frac{\ln 2}{\ln n})})$ with success probability at least $1-\left(\frac{(1-\delta)m\epsilon'^2}{\ln n}+\frac{1}{n}\right)$.

Remark 2. For sufficiently large n, t goes to infinity and thus γ converges to $\sqrt{\frac{\ln 2}{2m}}$. This implies that our algorithm asymptotically works if

$$\epsilon + \frac{\delta}{2} \leq \frac{1}{2} - \sqrt{\frac{(1-\delta)\ln 2}{2m}} - \epsilon' \tag{11}$$

and succeeds with a probability close to 1. Hereafter, we ignore the term "$-\epsilon'$" for simplicity.

If the erasure rate δ is 0, then the new method is equivalent to the HMM method [5] by Corollary 1. Therefore, the new method naturally combines the results of the Heninger-Shacham and HMM methods. The upper bound of the new method coincides with that of Heninger-Shacham for $\epsilon = 0$ and that of the HMM method for $\delta = 0$. Finally, we confirm that our algorithm works well for $\delta = 0.6, \epsilon = 0.001$. Remember that our algorithm works provided that $\epsilon + \delta/2 \leq 1/2 - 0.263\sqrt{1-\delta}$. The left-hand side is given by 0.301 and the right-hand side is given by 0.334; the left-hand side is less than the right-hand side. Our algorithm works in that case.

4 Implementation and Experiments

We implemented our algorithm in the Risa/Asir [11] computer algebra system and used the program on an Intel Xeon X5570 at 2.93 GHz with 72 GB memory of DDR3 at 1333 MHz. In our experiments for 1024-bit RSA, we prepared 100 different tuples of secret keys $\mathbf{sk}$, e.g. $\mathbf{sk} = (p, q, d, d_p, d_q)$. For a fixed ϵ and δ, we generated one erroneous version $\overline{\mathbf{sk}}$ for each of $\mathbf{sk}$. For a given T, the threshold value C_i is determined by using Eq. (8).

Table 1 shows the experimental results for the case of $\mathbf{sk} = (p, q, d, d_p, d_q)$, $\epsilon = 0.01$, and $T = 40$. Note that the erasure rate δ was selected to be smaller than the theoretical bound 0.684 estimated by Eq. (11). Similarly, but for $T = 75$, we

Table 1. Experiments for $\mathbf{sk} = (p, q, d, d_p, d_q)$, $n = 1024$, $\epsilon = 0.01$, and $T = 40$

δ	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65
success rate	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.91	0.42	0.02
average time (s)	0.45	0.57	0.83	0.98	0.99	1.41	1.75	1.91	2.05	2.24	2.07	1.56	0.97	0.59

Table 2. Experiments for $\mathbf{sk} = (p, q)$, $n = 1024$, $\epsilon = 0.01$, and $T = 75$

δ	0	0.05	0.10	0.15	0.20	0.25
success rate	1.00	1.00	0.97	0.91	0.42	0.04
average time (s)	14.06	5.86	3.26	1.07	0.25	0.08

also conducted the experiments for the case of $\mathbf{sk} = (p, q)$ and the results are given in Table 2.

For fixed n, ϵ, and T, if an erasure rate δ becomes large, then the average of depth t_i becomes large with the increase in δ by Eq. (7). The average of threshold values C_i also increase because of the process of determining C_i, namely, Eq. (8). We determine these C_i's to satisfy Restriction 1 for the fixed T, so the success rate of our algorithm becomes small as Tables 1 and 2 show. If we use $T = 80$ instead of $T = 40$ for the case of Table 1, the success rate for $\delta = 0.65$ increases to 0.21 from 0.02 and the average time becomes 40.14 seconds.

5 Theoretical Bound

This section derives a theoretical upper bound for key recovery from noisy secret keys with errors and erasures in polynomial time.

First, we define the Hamming distance between two l-bit sequences; the symbol of one sequence (Sequence 1) is $\{0, 1\}$ and that of the other sequence (Sequence 2) is $\{0, 1, E\}$, where E is an erasure symbol. We denote the number of positions at which the corresponding symbols are different by h. We also denote the number of symbols E in Sequence 2 by a. We define the Hamming distance b between two sequences by $b := h - a$. We also have the equivalent definition of Hamming distance as follows. First, remove the bit of the position at which the symbol in Sequence 2 is E in Sequence 1 and remove the symbol E in Sequence 2. We define the Hamming distance between Sequences 1 and 2 by the ordinary Hamming weight between resulting sequences. For example, the Hamming weight between 1111 and 1E01 is 1.

We recall some known facts about the binary Entropy function. Remember that the binary Entropy function $H(x)$ is defined by $H(x) = -x \log x - (1 - x) \log(1 - x)$. It is well known that the following inequalities hold between the number of combinations and the binary Entropy [3].

Lemma 1. *For any positive integer n and $w(\leq n)$, it holds that*

$$\frac{1}{\sqrt{8w(1 - w/n)}} 2^{nH(w/n)} \leq \sum_{i=0}^{w} \binom{n}{i} \leq 2^{nH(w/n)}. \tag{12}$$

It is known that $H(x)$ can be represented by the following sum of an infinite series:

$$H(x) = 1 - \frac{1}{\ln 2} \sum_{u=1}^{\infty} \frac{1}{2u(2u-1)} (2x-1)^{2u}. \tag{13}$$

5.1 Maximal-likelihood-based Approach

We consider the following meta-algorithm.

`Meta-Algorithm` **for Recovering Keys**
Input: Public key (N, e) and noisy secret key $\overline{\mathbf{sk}}$, (ϵ, δ)
Output: Correct secret key $\mathbf{sk}$
Step 1: Expansion Phase (Virtually) generate a candidate set $\mathcal{C}$ by using the public information and Eqs. (2)–(5). Note that the number of elements of $\mathcal{C}$ is given by $2^{n/2-1}$.
Step 2: Pruning Phase Discard the candidate that is not consistent with $\overline{\mathbf{sk}}$. We denote the obtained set by $\mathcal{C}^*$.
Step 3: Finalization Phase Test whether each candidate solution in $\mathcal{C}^*$ is indeed the correct $\mathbf{sk}$ with the help of public information

The design of Step 2 is crucial for our algorithm. It is important to adequately determine criteria in Step 2 so that the correct solution $\boldsymbol{c}$ is not discarded during Step 2 in $\mathcal{C}^*$ and $|\mathcal{C}^*|$ is as small as possible. We discuss concrete criteria for discarding a candidate solution in Step 2. In order to do so, we adopt the maximal-likelihood-based approach.

Our analysis relies on a similar heuristic assumption as that in [5] and [6].

Assumption 2 *Every candidate solution in $\mathcal{C}$ is a bit-wise sum of $n/2-1$ randomly chosen bit and the correct sequence $\boldsymbol{c}$.*

We denote a candidate solution by $\boldsymbol{c} \in \mathcal{C}$. We discuss the conditional probability that we observed $\overline{\mathbf{sk}}$ under the condition that $\boldsymbol{c}$ is the correct solution. We denote the conditional probability by $\Pr(\overline{\mathbf{sk}}; \boldsymbol{c})$ and we refer to $\Pr(\overline{\mathbf{sk}}; \boldsymbol{c})$ as *likelihood*. In the maximal likelihood-based-approach, we decide that candidate that maximizes $\Pr(\overline{\mathbf{sk}}; \boldsymbol{c})$ is the correct solution.

This probability is simply evaluated as follows:

$$\Pr(\overline{\mathbf{sk}}; \boldsymbol{c}) = \delta^a \epsilon^b (1-\epsilon-\delta)^{mn/2-a-b} = (\delta/(1-\epsilon-\delta))^a \epsilon^b (1-\epsilon-\delta)^{mn/2-b},$$

where a is the number of erasure symbols in $\overline{\mathbf{sk}}$ and b is the Hamming distance between $\overline{\mathbf{sk}}$ and $\boldsymbol{c}$.

Since a does not depend on the choice of $\boldsymbol{c}$, it is sufficient to find b that maximizes the likelihood. If b is smaller, the likelihood is obviously bigger. Then, it is sufficient to find the solution $\boldsymbol{c}$ with the smallest Hamming distance to $\overline{\mathbf{sk}}$ for finding the solution that maximizes the likelihood. The Hamming distance b_{c} between the correct solution and $\overline{\mathbf{sk}}$ is $b_{\mathrm{c}} \approx \frac{mn\epsilon}{2}$ with high probability.

Meanwhile, the Hamming distance b_{w} between the wrong solution and $\overline{\mathbf{sk}}$ is $b_{\mathrm{w}} \approx m \times \frac{n}{2} \times \frac{1-\delta}{2} (> \frac{mn\epsilon}{2})$ with high probability. Then, it is sufficient to find the solution whose Hamming distance is $mn\epsilon/2$ in order to find the solution with maximal likelihood.

Remark 3. The computation of our proposed algorithm described in section 3 corresponds to finding the solution whose Hamming distance is less than $m \times \frac{n}{2} \times (1-\delta) \times (\frac{1}{2} - \gamma)$ for small positive γ. This implies that the correct solution is not discarded and falls within $\mathcal{C}^*$ with high probability. However, the size of $\mathcal{C}^*$ increases.

Remark 4. It is obviously impossible to execute Step 2 if the computational time is limited to a polynomial of n. In practice, we need to divide the candidate sequence into several sub-sequences and execute the expansion and pruning phase as in our proposed algorithm in section 3.

5.2 Deriving Theoretical Upper Bound

We derive the condition such that the `meta-algorithm` can never recover the secret key in polynomial time. This can be done by counting up the candidate solution that is not discarded during Step 2 and deriving the condition of (ϵ, δ) when the number of candidate solutions exceeds the polynomial of n.

We note that the candidate solution $\boldsymbol{c}$ is consistent with the observed solution $\overline{\mathbf{sk}}$ in Step 2 of the `meta-algorithm` if the following criteria hold.

CRITERIA The Hamming distance between $\boldsymbol{c}$ and $\overline{\mathbf{sk}}$ is less than $mn\epsilon/2$.

Note that the expected bit length of the sequences removing erasures is given by $mn(1-\delta)/2$. The probability Pr that one candidate $\boldsymbol{c}$ is consistent with $\overline{\mathbf{sk}}$ is evaluated by

$$\Pr = \frac{\sum_{i=0}^{mn\epsilon/2} \binom{mn(1-\delta)/2}{i}}{2^{mn(1-\delta)/2}}. \tag{14}$$

From Lemma 1, Eq. (14) is lower bounded by

$$\Pr \geq 2^{-mn(1-\delta)(1-H(\epsilon/(1-\delta)))/2}. \tag{15}$$

We define $C(\epsilon, \delta)$ by $C(\epsilon, \delta) := (1-\delta)(1 - H(\epsilon/(1-\delta)))$. Then, the probability is larger than $2^{-mnC(\epsilon,\delta)/2}$. Since the number of candidate solutions is $2^{n/2}$, the expected number of candidate solutions consistent with the observed sequence $\overline{\mathbf{sk}}$ is lower bounded by $2^{n/2} 2^{-mnC(\epsilon,\delta)/2} = 2^{n(1-mC(\epsilon,\delta))/2}$.

Suppose that ϵ and δ satisfy the condition: $C(\epsilon, \delta) < 1/m$. This implies that $1 - mC(\epsilon, \delta) > 0$. Then, the expected number of candidate solutions consistent with $\overline{\mathbf{sk}}$ is an exponential function of n. Step 3 then requires the exponential testing of whether the candidate is indeed the secret key. Hence, the total computational time of the whole algorithm is actually exponential.

Conversely, suppose that $C(\epsilon, \delta) \geq 1/m$. Then, the number of candidate solutions is at most a polynomial of n and the total computational time dominates Step2. This means that it depends on $C(\epsilon, \delta)$ and $1/m$ whether there exists an algorithm that recovers in polynomial time of n. We show an information-theoretic view of our theoretical bound in the full version.

5.3 Discussion

Fig.1 shows achievable regions for the naive method and our proposed method in addition to the theoretical bound with $m = 5$. Note that all the values that lie below the respective line concerning the naive method and proposed method are vulnerable to each of the attacks and all the values that lie above the line about theoretical limitation are not solvable in polynomial time. We can see that the bound for our method nearly achieves the theoretical bound, but there is still a small gap.

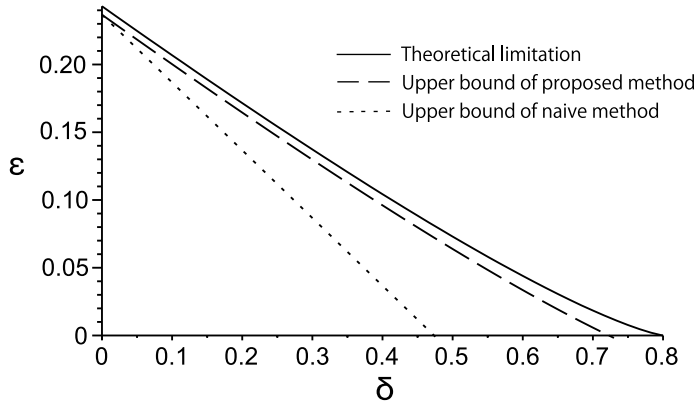

Fig. 1. Upper bounds of naive method and new method, and theoretical limitation

Table 3 shows the success conditions for three noise models; the upper is the bound the best-known algorithm achieves and the lower is the theoretical bound.

Table 3. Success Conditions of Heninger-Shacham, HMM, and our Proposed Methods

	model	BE model ($\epsilon = 0$)	BS model ($\delta = 0$)	BEE model
	best known algorithm	Heninger-Shacham [6]	HMM [5]	Proposed Method in Sec. 3
2	algorithm	$\delta \leq 0.43$	$\epsilon \leq 0.084$	$\epsilon + \delta/2 \leq \frac{1}{2} - 0.416\sqrt{1-\delta}$
2	theoretical bound	$\delta \leq 0.5$	$\epsilon \leq 0.110$	(ϵ, δ) s.t. $C(\epsilon, \delta) \geq 1/2$
5	algorithm	$\delta \leq 0.73$	$\epsilon \leq 0.237$	$\epsilon + \delta/2 \leq \frac{1}{2} - 0.263\sqrt{1-\delta}$
5	theoretical bound	$\delta \leq 0.8$	$\epsilon \leq 0.243$	(ϵ, δ) s.t. $C(\epsilon, \delta) \geq 1/5$
m	algorithm	$\delta \leq 1 - \frac{2\ln 2}{m}$	$\epsilon \leq \frac{1}{2} - \sqrt{\frac{\ln 2}{2m}}$	$\epsilon + \frac{\delta}{2} \leq \frac{1}{2} - \sqrt{\frac{(1-\delta)\ln 2}{2m}}$
m	theoretical bound	$\delta \leq 1 - \frac{1}{m}$	ϵ s. t. $H(\epsilon) \leq 1 - \frac{1}{m}$	(ϵ, δ) s.t. $C(\epsilon, \delta) \geq 1/m$

5.4 Our Algorithm Achieves Second-order Expansion of Theoretical Bound

We present a strong bridge between the theoretical bound and achieved regions. We define the whole parameter space $\mathcal{I}$ by $\mathcal{I} := \{(\epsilon, \delta) | 0 \leq \epsilon < 1/2, 0 \leq \delta < 1\}$ and define $\mathcal{H}$ by

$$\mathcal{H} := \left\{ (\epsilon, \delta) | 0 \leq \epsilon < 1/2, 0 \leq \delta < 1, (1-\delta)\left(1 - H\left(\frac{\epsilon}{1-\delta}\right)\right) \geq \frac{1}{m} \right\}.$$

The discussion in Section 5.3 shows that we cannot recover the secret keys in polynomial time if $(\epsilon, \delta) \in \mathcal{I}/\mathcal{H}$. This argument suggests that we have a chance to recover the secret key in polynomial time if $(\epsilon, \delta) \in \mathcal{H}$. However, it does not guarantee that we can recover the secret keys if $(\epsilon, \delta) \in \mathcal{H}$. As shown in Fig. 1, there exists a small gap between our theoretical bound and the achieved regions. We give a strong relation between the two regions.

From Eq. (13), $C(\epsilon, \delta) < 1/m$ can be represented as follows:

$$\sum_{u=1}^{\infty} \frac{(1-\delta)}{2u(2u-1)} \left(\frac{1-\delta-2\epsilon}{1-\delta}\right)^{2u} \leq \frac{\ln 2}{m}, \tag{16}$$

which is a representation not explicitly used by the binary Entropy $H(\cdot)$. Consider the condition truncated by $u = k$ and denote the condition by $\mathcal{H}_k$

$$\mathcal{H}_k := \left\{ (\epsilon, \delta) | 0 \leq \epsilon < 1/2, 0 \leq \delta < 1, \sum_{u=1}^{k} \frac{(1-\delta)}{2u(2u-1)} \left(\frac{1-\delta-2\epsilon}{1-\delta}\right)^{2u} \leq \frac{\ln 2}{m} \right\}.$$

Obviously, it holds that $\mathcal{H}_i \subseteq \mathcal{H}_j$ if $i \leq j$ for any $i, j \in \mathbb{Z}$ and it holds that $\lim_{k\to\infty} \mathcal{H}_k = \mathcal{H}$.

We focus on the region $\mathcal{H}_1$. By simplifying the condition corresponding to $\mathcal{H}_1$, we have the equivalent condition:

$$1 - \delta - 2\epsilon \geq \sqrt{2(1-\delta)\ln 2/m}.$$

This is equivalent to the condition obtained in section 3: Eq. (11) if we neglect the small term ϵ. This implies that our proposed algorithm can recover the secret key in polynomial time if $(\epsilon, \delta) \in \mathcal{H}_1$.

Acknowledgement

We thank PKC 2012 and Asiacrypt 2012 reviewers for revising the manuscript. The first author was supported by JSPS Grant Number KAKENHI22700006.

References

1. D. Boneh, G. Durfee, and Y. Frankel, "An Attack on RSA Given a Small Fraction of the Private Key Bits," in Proc. of Asiacrypt'98, LNCS 1514, pp. 25-34, 1998.

2. D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known," in Proc. of Eurocrypt'96, LNCS 1070, pp. 178–189, 1996.
3. C. M. Cover and J. A. Thomas, "Elements of Information Theory, 2nd Edition," Wiley-Interscience, 2006.
4. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum and, E. W. Felten, "Lest We Remember: Cold Boot Attacks on Encryption Keys," in Proc. of USENIX Security Symposium 2008, pp. 45–60, 2008.
5. W. Henecka, A. May, and A. Meurer, "Correcting Errors in RSA Private Keys," in Proc. of Crypto 2010, LNCS 6223, pp. 351–369, 2010.
6. N. Heninger and H. Shacham, "Reconstructing RSA Private Keys from Random Key Bits," in Proc. of Crypto 2009, LNCS 5677, pp. 1–17, 2009.
7. M. Herrmann and A. May, "Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits," in Proc. of Asiacrypt 2008, LNCS5350, pp. 406-424, 2008.
8. W. Hoeffding, "Probability Inequalities for Sums of Bounded Random Variables," Journal of the American Statistical Association, 58(301), pp. 13–30, 1963.
9. K. G. Paterson, A. Polychroniadou and D. L. Sibborn, "A Coding-Theoretic Approach to Recovering Noisy RSA Keys," in Proc. of Asiacrypt 2012, LNCS 7658, pp. 386–403, 2012.
10. PKCS #1 Standard for RSA. Available at `http://www.rsa.com/rsalabs/node.asp?id=2125`.
11. Risa/Asir (Kobe Distribution) Download Page, `http://www.math.kobe-u.ac.jp/Asir/asir.html`, 2011.
12. R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21(2), pp. 120–126, 1978.
13. S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage, "When Private Keys are Public: Results from the 2008 Debian OpenSSL Vulnerability," IMC 2009, ACM Press, pp. 15–27, 2009.

A Proofs of Theorem 2 and Corollary 1

A.1 Proofs of Theorem 2

First, we discuss how to determine the threshold value C_i satisfying Restriction 1 for a fixed T. Note that t_i and Δ_i are uniquely determined if T is fixed once.

In one i-th partial tree of the binary tree, there are 2^{t_i} candidates. Thus we defines 2^{t_i} variables $Z_{b,i}^j$ for $j = 1, \ldots, 2^{t_i}$ as

$$Z_{b,i}^j = \begin{cases} 1 \ (j\text{-th bad candidate passes}) \\ 0 \ (\text{otherwise.}) \end{cases}$$

Then, the number of bad candidates $Z_{b,i}$ given in Restriction 1 is described as $Z_{b,i} = \sum_{j=1}^{2^{t_i}} Z_{b,i}^j$. Since all $Z_{b,i}^j$ are identically distributed, there exists an integer j such that $\mathbb{E}[Z_{b,i}] = 2^{t_i}\mathbb{E}[Z_{b,i}^j]$.

Here we consider the number $X_{b,i}$ of matching bits between $\overline{\mathbf{sk}}$ and one bad candidate at the i-th pruning step skipping bits corresponding to erasure

positions of $\overline{\mathbf{sk}}$. Since T bits of a bad candidate are compared with the corresponding bits of $\overline{\mathbf{sk}}$, we have that $X_{b,i} \sim \mathrm{Bin}(T, 1/2)$ by Assumption 1. The condition $Z^j_{b,i} = 1$ is equivalent to that $X_{b,i} \geq C_i$, and thus $\mathbb{E}[Z^j_{b,i}] = \Pr[Z^j_{b,i} = 1] = \Pr[X_{b,i} \geq C_i]$. Supposing that

$$C_i = T\left(\frac{1}{2} + \gamma_i\right), \tag{17}$$

we have $\Pr[X_{b,i} \geq C_i] \leq \exp(-2T\gamma_i^2)$ from Theorem 1. Therefore, we obtain that $\mathbb{E}[Z_{b,i}]/2^{t_i} = \mathbb{E}[Z^j_{b,i}] \leq \exp(-2T\gamma_i^2)$. By setting

$$\gamma_i = \sqrt{\frac{t_i + 1}{T}\frac{\ln 2}{2}}, \tag{18}$$

we have $\exp(-2T\gamma_i^2) = 2^{-(t_i+1)}$. Restriction 1 holds since $\mathbb{E}[Z_{b,i}] \leq 2^{t_i}\exp(-2T\gamma_i^2) = 1/2$.

Let Y_i be the number of all bad candidates passing the i-th pruning step. Then, we have the following lemma.

Lemma 2. *Suppose that γ_i and C_i satisfy Eqs. (17) and (18) for a fixed T. Then, it holds that $\mathbb{E}[Y_i] < 2^{\max\{t_j\}_{j=1}^{i}+1}$.*

Proof. At the i-th pruning step, let $Z_{g,i}$ be the number of bad candidates generated from the correct solution, and $Z_{b,i}$ the number of bad candidates generated from one bad partial solution. Then, the following holds:

$$\mathbb{E}[Y_1] = \mathbb{E}[Z_{g,1}],\ \mathbb{E}[Y_i] = \mathbb{E}[Z_{g,i}] + \mathbb{E}[Z_{b,i}]\mathbb{E}[Y_{i-1}]. \tag{19}$$

Since the number of candidates is 2^{t_i}, we have $\mathbb{E}[Z_{g,i}] \leq 2^{t_i}$. For a given T, namely a fixed t_i, we determine γ_i and C_i so that Restriction 1 holds. From (19), we have

$$\mathbb{E}[Y_i] < 2^{t_i} + \frac{\mathbb{E}[Y_{i-1}]}{2} < 2^{\max\{t_j\}_{j=1}^{i}}\frac{1-(1/2)^i}{1-1/2} < 2^{\max\{t_j\}_{j=1}^{i}+1}.$$

Then, we have the lemma. □

Next we discuss T such that Restriction 2 holds. Let $X_{c,i}$ be the number of matching bits between $\overline{\mathbf{sk}}$ and the correct solution at the i-th pruning step without the bits corresponding to erasure positions of $\overline{\mathbf{sk}}$. Since we see total $(T + \Delta_i)$ bits and the T bits of them correspond to the non-erasure position of $\overline{\mathbf{sk}}$, the probability that a bit of a correct solution matches the corresponding bit of $\overline{\mathbf{sk}}$ is $(T - (T + \Delta_i)\epsilon)/T = 1 - (T + \Delta_i)\epsilon/T$. Therefore, since $X_{c,i} \sim \mathrm{Bin}(T, 1 - \frac{(T+\Delta_i)\epsilon}{T})$, we suppose that

$$\frac{1}{2} + \gamma_i \leq 1 - \frac{(T + \Delta_i)\epsilon}{T} - \epsilon',$$

for any i. From Theorem 1, we have that

$$\Pr[X_{c,i} < C_i] \leq \Pr\left[X_{c,i} < T\left(1 - \frac{(T + \Delta_i)\epsilon}{T} - \epsilon'\right)\right] \leq \exp(-2T\epsilon'^2).$$

Since we consider T such that Restriction 2 holds, $\exp(-2T\epsilon'^2) \leq 1/n$. Therefore, we have $T \geq \ln n/2\epsilon'^2$, and so we set $T = \lceil \ln n/(2\epsilon'^2) \rceil$.

By considering the above discussion, we have Theorem 2. The proof of Theorem 2 is given in detail below.

Proof. First we show that the total expected computational cost of the new method is $\mathcal{O}(n^{2+2(\frac{\ln 2}{2m\epsilon'^2}+\frac{\Delta}{m}\frac{\ln 2}{\ln n})})$. One node is computable in time $\mathcal{O}(n)$, so the partial tree is generated in time $\mathcal{O}(n2^{t_i})$ since there are $\sum_{j=0}^{t_i-1} 2^j (< 2^{t_i})$ nodes. The pruning step can be performed in time $\mathcal{O}(t_i)$ for each of 2^{t_i} candidates, and thus the total time complexity for pruning is $\mathcal{O}(t_i 2^{t_i})$. Therefore, the time complexity for one partial tree is $\mathcal{O}((n+t_i)2^{t_i}) = \mathcal{O}(n2^{t_i})$. For a given T, we suppose that the erroneous version $\overline{\mathbf{sk}}$ is separated into ℓ parts. By Eq. (7), t_i is bounded by $t_i^* = \lceil \frac{T+\Delta_i}{m} \rceil$. Let t^* be the maximum integer of $t_1^*, \ldots, t_\ell^*$. By Lemma 2, the upper bound for the expected total number $\mathbb{E}[Y]$ of partial trees is given by $\mathbb{E}[Y] < 1 + \sum_{j=1}^{\ell-1} \mathbb{E}[Y_j] < \ell 2^{t^*+1} \leq n2^{t^*+1} = \mathcal{O}(n2^{t^*})$. Let Δ be the Δ_i corresponding to t^*. Then, the total expected computational cost is

$$\mathcal{O}(n2^{t^*} \cdot n2^{t^*}) = \mathcal{O}(n^2 n^{2t^* \frac{\ln 2}{\ln n}}) = \mathcal{O}(n^2 n^{2\frac{T+\Delta}{m}\frac{\ln 2}{\ln n}}) = \mathcal{O}(n^{2+2(\frac{\ln 2}{2m\epsilon'^2}+\frac{\Delta}{m}\frac{\ln 2}{\ln n})}).$$

Next we discuss the success probability of the new method. Note that C_i, γ_i and T are determined so that Restriction 2 holds. Hence the success probability is given by

$$\prod_{i=1}^{\ell}(1 - \Pr[X_c < C_i]) \geq \left(1 - \frac{1}{n}\right)^{\ell} \geq 1 - \frac{\ell}{n} \geq 1 - \left(\frac{(1-\delta)m\epsilon'^2}{\ln n} + \frac{1}{n}\right)$$

since $\ell \leq \frac{\frac{n}{2}(1-\delta)m}{T} + 1$. □

A.2 Proofs of Corollary 1

To give the proof of Corollary 1, we begin with the discussion of Eq. (9) in the analysis of our method. For simplicity, we consider only the case where all δ_i's are the same[4], for example, $\delta_i = \delta$. Suppose that $\overline{\mathbf{sk}}$ is separated into ℓ fractions. Then, each part consists of $mn/2\ell$ bits. By letting $t = n/2\ell$, we have $\Delta = \delta tm$ and $T = tm - \Delta = (1-\delta)tm$, so we can describe γ_i in Theorem 2 as $\sqrt{\frac{t+1}{(1-\delta)tm}\frac{\ln 2}{2}}$. Hence, in this case, the upper bound (9) implies that

$$\epsilon + \frac{\delta}{2} \leq \frac{1}{2} - \sqrt{\left(1 + \frac{1}{t}\right)\frac{(1-\delta)\ln 2}{2m}} - (1-\delta)\epsilon'.$$

[4] For a large enough T, it holds with high probability. More precisely, all of δ_i takes the value close to $\delta T/(1-\delta)$ with overwhelming probability, which can be proved by the similar analysis of [6].