Improved Key Recovery Algorithms from Noisy RSA Secret Keys with Analog Noise

Noboru Kunihiro[1] and Yuki Takahashi[2]

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

Abstract. From the proposal of key-recovery algorithms for RSA secret key from its noisy version at Crypto2009, there have been considerable researches on RSA key recovery from discrete noise. At CHES2014, two efficient algorithms for recovering secret keys are proposed from noisy analog data obtained through physical attacks such as side channel attacks. One of the algorithms works even if the noise distributions are unknown. However, the algorithm is not optimal especially if the noise distribution is imbalanced. To overcome this problem, we propose new algorithms to recover from such an imbalanced analog noise. We first present a generalized algorithm and show its success condition. We then construct the algorithm suitable for imbalanced noise under the condition that the variances of noise distributions are a priori known. Our algorithm succeeds in recovering the secret key from much more noise. We present the success condition in the explicit form and verify that our algorithm is superior to the previous results. We then show its optimality. Note that the proposed algorithm has the same performance as the previous one in the balanced noise. We next propose a key recovery algorithm that does not use the values of the variances. The algorithm first estimates the variance of noise distributions from the observed data with help of the EM algorithm and then recover the secret key by the first algorithm with their estimated variances. The whole algorithm works well even if the values of the variance is unknown in advance. We examine that our proposed algorithm succeeds in recovering the secret key from much more noise than the previous algorithm. This is the full version of [12].

1 Introduction

1.1 Background and Motivation

RSA [15] is the most widely used cryptosystem and its security is based on the difficulty of factoring a large composite. Furthermore, the side-channel attacks are a real threat to RSA scheme. This kind of attack can be executed by physically observing cryptographic devices and recovering internal information. Side channel attacks are important concerns for

security analysis in the both of public key cryptography and symmetric cryptography. In the typical scenario of the side channel attacks, an attacker tries to recover the full secret key when he can measure some leaked information from cryptographic devices.

In this paper, we focus on the side channel attacks on RSA cryptosystem. In the RSA cryptosystem [15], a public modulus N is the product of two distinct primes p and q. The public and secret exponents are (e, d), which satisfy $ed \equiv 1 \pmod{(p-1)(q-1)}$. In the textbook RSA, the secret key is only d. However, the PKCS#1 standard [14] specifies that the RSA secret key includes $(p, q, d, d_p, d_q, q^{-1} \bmod p)$ in addition to d, which allows for fast decryption using the Chinese Remainder Theorem (CRT). It is important to analyze the security of CRT-RSA in addition to that of the original RSA.

Halderman et al. [4] presented the cold boot attack at USENIX Security 2008, which is classified as a practical side channel attack. They demonstrated that DRAM remanence effects make possible practical, nondestructive attacks that recover a noisy version of secret keys stored in a computer's memory. They showed how to reconstruct the full of the secret key from the noisy variants for some encryption schemes including RSA scheme. How to recover the correct secret key from a noisy version of the secret key is an important question concerning the cold boot attack situation.

Inspired by cold boot attacks [4], there have been considerable researches on RSA secret key recovery from discrete noise [5, 6, 9, 13]. In contrast, Kunihiro and Honda introduced an analog leakage model and proposed two efficient key recovery algorithm (ML-based algorithm and DPA-like algorithm) from the observed analog data [10].

Observing Analog Data and Motivation Consider the simple power analysis [8] for CRT-RSA, which is conducted by observing power consumption while executing decryption process. Power consumption trace depends on the bit value of d_p (and d_q). We can obtain analog data from the observed trace through some adequate functions. Further, the distributions of such analog data for the bit 0 and 1 differ from each other due to the difference of power consumption trace. In the same manner, we can obtain the analog data for the bit of d. Note that we cannot obtain those of p and q in the scenario.

In another attack scenario, analog data may be obtained from a (discrete) measurement value of bit value with an analog value of *confidence*. This will be done by some side-channel attacks such as cold boot attack

where different pieces of RAM have different preference to flip towards 0 or to flip towards 1.

Our main research aim is to propose efficient algorithms when such analog data, especially imbalanced analog data, are obtained.

1.2 Our Contributions

This paper discusses secret key recovery algorithms from noisy analog data. In our noise model, the observed value is output according to some fixed probability distribution depending on the corresponding correct secret key bit. Unlike [10], we do not assume that the probability density functions are known. Our strategy for constructing the algorithms is summarized as follows: (i) estimate the probability density functions and (ii) run the key-recovery algorithm with the score function designed by the estimated one. We present the success condition (Theorem 3) in adapting the strategy, which shows that we can recover the secret key from more noisy keys if we could succeed to obtain a closer estimation of the probability density functions.

Next, we propose an efficient algorithm (V-based algorithm) to improve the success condition from that of [10]. We propose a new score function (Variance-based Score) by modifying the DPA-like score function introduced in [10] to suit for imbalanced noise. Concretely, we incorporate the variances of the probability distributions into the DPA-like score. By this modification, we succeed in improving the success condition compared to the DPA-like algorithm in [10]. We then present the success condition in the explicit form (Theorem 4), which significantly improves the previously shown bounds. We then prove that Variance-based score is optimal in the weighted variant of DPA-like score. Moreover, we then verify that our algorithm is superior to the previous results by both of theoretical analysis and numerical experiments for various noise distributions. Note it has the same performance as the DPA-like algorithm in the balanced noise.

Although our first algorithm improves the bound, it requires the values of the variances as additional inputs, which is a significant disadvantage to the DPA-like algorithm. To overcome this problem, we use the help of the Expectation-Maximization (EM) algorithm [1, 3], which is a well-known algorithm in the area of machine learning, to estimate the variances from the observed data. The second algorithm (KRP algorithm) is constructed by combining V-based algorithm and the EM algorithm. In our combined algorithm, we first run the EM algorithm for the estimation of the variances and run the V-based algorithm with the estimated variances as

additional inputs. The KRP algorithm works under the same condition as the DPA-like algorithm, that is, that we can use only the observed data. The numerical results show that our KRP algorithm is superior to the DPA-like algorithm. For example, when the standard deviation of noises (a precise noise model is discussed in Section 2.2) is given 0.4 and 2.2, DPA-like algorithm succeeds with probability 0.16, but KRP algorithm succeeds with probability 0.65 (see Table 2). We also verify the effectiveness of our algorithms by numerical experiments on several noise distributions: Gaussian, Laplace, and Uniform distributions.

2 Preliminaries

This section presents an overview of the methods [5, 6, 10, 13] using binary trees to recover the secret key of the RSA cryptosystem. We use similar notations to those in [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 $2^{\tau(M)} | M$. We denote by $\ln n$ the natural logarithm of n to the base e and by $\log n$ the logarithm of n to the base 2. We denote the expectation of random variable X by $\mathrm{E}[X]$. We remind readers the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. The probability density function of this distribution is $f_{\mathrm{N}}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, where μ and σ^2 are the mean and variance of the distribution, respectively.

2.1 Recovering the RSA Secret Key Using a Binary Tree

An explanation of this subsection is almost the same as previous works [5, 6, 10, 13]. We first explain how to set the keys of the RSA cryptosystem [15], especially of the PKCS #1 standard [14]. Let (N, e) be the RSA public key and $\mathbf{sk} = (p, q, d, d_p, d_q, q^{-1} \bmod p)$ be the RSA secret key. We denote the bit length of N by n. As in the previous works, we ignore the last component $q^{-1} \bmod p$ in the secret key. The public and secret keys follow four equations: $N = pq$, $ed \equiv 1 \pmod{(p-1)(q-1)}$, $ed_p \equiv 1 \pmod{p-1}$, $ed_q \equiv 1 \pmod{q-1}$. Then, there exist integers k, k_p and k_q such that

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

A small public exponent e is usually used in practical applications [16], so we suppose that e is small enough such that $e = 2^{16} + 1$ as is the case

in [5, 6, 9, 10, 13]. See [5] for how to compute k, k_p and k_q. Then there are five unknowns (p, q, d, d_p, d_q) in the four equations in Eq. (1).

In the same manner as previous methods, our new methods recover secret key **sk** by using a binary tree based technique. We explain how to recover secret keys, considering $\mathbf{sk} = (p, q, d, d_p, d_q)$ as an example.

First we discuss the generation of the tree. Since p and q are $n/2$ bit prime numbers, there exist at most $2^{n/2}$ candidates for each secret key in (p, q, d, d_p, d_q). Heninger and Shacham [6] introduced the concept of **slice**. We define the i-th bit slice for each bit index i as $\mathbf{slice}(i) := (p[i], q[i], d[i + \tau(k)], d_p[i + \tau(k_p)], d_q[i + \tau(k_q)])$. 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 obtained the following identities

$$\begin{aligned}
p[i] + q[i] &= (N - p'q')[i] \bmod 2, \\
d[i + \tau(k)] + p[i] + q[i] &= (k(N + 1) + 1 - k(p' + q') - ed')[i + \tau(k)] \bmod 2, \\
d_p[i + \tau(k_p)] + p[i] &= (k_p(p' - 1) + 1 - ed'_p)[i + \tau(k_p)] \bmod 2, \\
d_q[i + \tau(k_q)] + q[i] &= (k_q(q' - 1) + 1 - ed'_q)[i + \tau(k_q)] \bmod 2.
\end{aligned}$$

This means that we have four linearly independent equations in the five unknowns $p[i], q[i], d[i + \tau(k)], d_p[i + \tau(k_p)]$, and $d_q[i + \tau(k_q)]$ of $\mathbf{slice}(i)$. Each Hensel lift, therefore, yields exactly two candidate solutions. Then, the total number of candidates is given by $2^{n/2}$.

Henecka et al.'s algorithm [5] and Paterson et al.'s algorithm (in short, the PPS algorithm) [13] perform t Hensel lifts for some fixed parameter t. For each surviving candidate solution on $\mathbf{slice}(0)$ to $\mathbf{slice}(it - 1)$, a tree with depth t and whose 2^t leaf nodes represent candidate solutions on $\mathbf{slice}(it)$ to $\mathbf{slice}((i + 1)t - 1)$, is generated. This causes $5t$ new bits. For each new node generated, a pruning phase is carried out. A solution is kept for the next iteration if the likelihood of the corresponding noisy variants of the secret key for the $5t$ new bits is in the highest L nodes of the $L2^t$ nodes as for the PPS algorithm [13]. Kunihiro and Honda [10] adopted a similar approach to the PPS algorithm [13]. They introduced a concept of score function, and their algorithms keep the top L nodes with the highest score.

2.2 Our Noise Model

Let F_0 and F_1 be probability distributions of an observed value when the correct secret key bits are 0 and 1, respectively. That means we assume

that each the observed value x follows the *fixed* probability distribution F_b. Though this assumption comes from simplification, it is frequently considered in the practice of side channel attacks. In this paper, we assume that F_0 and F_1 have probability densities f_0 and f_1, respectively. Without loss of generality, we assume that the means of F_b are $(-1)^b$. Throughout this paper, we assume that f_0 and f_1 are unknown to the attackers. That implies that we do not use any knowledge about explicit forms of probability density functions in designing algorithms.

We say that the probability density functions f_0 and f_1 are *imbalanced* when $f_0(x)$ and $f_1(-x)$ are (very) different. Suppose that $f_0 = \mathcal{N}(+1, \sigma_0^2)$ and $f_1 = \mathcal{N}(-1, \sigma_1^2)$. We say that f_0 and f_1 are imbalanced when $\sigma_0 \ll \sigma_1$. (Note that $f_1(-x) = \mathcal{N}(x; +1, \sigma_1^2)$). In this paper, we mainly focus on the case that f_0 and f_1 are imbalanced.

2.3 Previous Works on Key-Recovery for Analog Observed Data

In [10], a score function is introduced, that is calculated with observed data and a candidate sequence (if necessary, additional information such as the probability density functions of noise). A framework of key-recovery algorithm that uses the score function in Pruning phase is then proposed. Definition 1 gives the syntax of the score function.

Definition 1 (Syntax of Score Function). The score function receives a candidate sequence $\mathbf{b} = (b_1, \ldots, b_n) \in \{0,1\}^n$ and the corresponding observed sequence $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and outputs a real number. We use the notation: $\mathbf{Score}_n(\mathbf{b}, \mathbf{x})$.

The score function $\mathbf{Score}_n(\mathbf{b}, \mathbf{x})$ is designed so that the following properties hold for any fixed $\mathbf{x}$: the score will be large if $\mathbf{b}$ is a correct candidate; the score will be small if $\mathbf{b}$ is incorrect.

We review a framework shown in [10, 13] for the RSA key-recovery algorithm. Our proposed algorithms are based on the same framework. It is pointed out in [10] that the setting $t = 1$ is enough for gaining high success rates. We use slightly different notations of generalized PPS algorithm from [10]. Algorithm 2 in Appendix A.1 shows the framework for key-recovery. Revising the algorithm framework itself is not our target. This paper mainly focuses on designing the score function.

The following two score functions have been proposed in [10]. Denote a candidate sequence $\mathbf{b} = (b_1, \ldots, b_n)$ and an observed sequence $\mathbf{x} =$

$(x_1, \ldots, x_n)$. The first one is defined by

$$\mathrm{ML}(\mathbf{b}, \mathbf{x}) := \sum_{i=1}^{n} \log \frac{f_{b_i}(x_i)}{g(x_i)}, \tag{2}$$

where $g(x) = (f_0(x) + f_1(x))/2$. The second one is defined by

$$\mathrm{DPA}(\mathbf{b}, \mathbf{x}) := \sum_{i=1}^{n} (-1)^{b_i} x_i. \tag{3}$$

Eq. (2) and Eq. (3) are called as ML-based score and DPA-like score, and the algorithms employing Eq. (2) and Eq. (3) are called as ML-based algorithm and DPA-like algorithm, respectively. Note that the ML-based algorithm requires the complete information about probability density functions as inputs. In contrast, the DPA-like algorithm does not require them as shown in Eq. (3).

We summarize the success condition for the ML-based algorithm, and the DPA-like algorithm [10]. First, we introduce a differential entropy [2].

Definition 2. The differential entropy $h(f)$ of a probability density function f is defined as

$$h(f) = -\int_{-\infty}^{\infty} f(y) \log f(y) \mathrm{d}y.$$

Theorem 1 (Cor.1, [10]). Assume that the probability density functions for $b = 0, 1$ are given by f_b. The error probability of the ML-based Algorithm converges to zero as $L \to \infty$ if

$$h\left(\frac{f_0 + f_1}{2}\right) - \frac{h(f_0) + h(f_1)}{2} > \frac{1}{5}. \tag{4}$$

Theorem 2 (Thm.2, [10]). Assume that the probability density functions for $b = 0, 1$ are given by f_b. Denote the variance of F_b by σ_b^2. The error probability of the DPA-like Algorithm converges to zero as $L \to \infty$ if

$$h\left(\frac{f_0 + f_1}{2}\right) - \log \sqrt{\pi \mathrm{e}(\sigma_0^2 + \sigma_1^2)} > \frac{1}{5}. \tag{5}$$

Consider the case that f_0 and f_1 are imbalanced. Without loss of generality, we assume that $\sigma_0 \ll \sigma_1$. In this case, the left-hand side of Eq. (5) heavily depends on only the variance σ_1, which is unnatural. We will give improvement of the success condition by incorporating the values σ_b^2 to the score function in Section 4.

Remark 1. Throughout the paper, we only consider the case that we employ (p, q, d, d_p, d_q) as secret key tuple. However, we can easily extend to more general case. For the (p, q), (d_p, d_q), (p, q, d), and (d, d_p, d_q) cases, we just replace $1/5$ with $1/2, 1/2, 1/3$, and $1/3$, respectively in Theorems 1–4 and Eq. (11).

3 Generalized Algorithm via Estimation of Distributions

In actual attack situations, the attacker does not know the exact form of f_b. Then, we cannot apply the ML-based score directly. On the other hands, if one could obtain a closer estimation of probability density functions, one can hope to attain the key-recovery from larger noise. The second best strategy is then (i) to estimate f_b in some way (discussed in Section 5) and (ii) to run the key recovery algorithm with the score function designed by estimated probability density functions. In this section, we will derive the success condition under the condition that we have learned the estimation of the distributions. We denote the estimated distributions of f_0 and f_1 by $f_0^{(\mathrm{E})}$ and $f_1^{(\mathrm{E})}$, respectively.

Before giving the detailed analysis, we introduce the Kullback-Leibler divergence [2].

Definition 3. For the probability density functions p and q, the Kullback–Leibler divergence $D(p||q)$ of p and q is defined as

$$D(p||q) = \int_{-\infty}^{\infty} p(y) \log \frac{p(y)}{q(y)} \mathrm{d}y.$$

It is well-known that the Kullback–Leibler divergence $D(p||q)$ is non-negative and it is zero if and only if $p = q$. It is considered as some kind of the distance between p and q.

We introduce a new notion of the score function based on the estimated probability density functions, which would be a natural modification of ML-based score. We define the new score as

$$R^{(\mathrm{E})}(\mathbf{b}, \mathbf{x}) = \sum_i \log f_{b_i}^{(\mathrm{E})}(x_i). \tag{6}$$

In the modification, we replace the true densities f_b with their estimations $f_b^{(\mathrm{E})}$ (and ignore the denominator). Using $R^{(\mathrm{E})}(\mathbf{b}, \mathbf{x})$ as a score function, we have the following theorem.

Theorem 3. Assume that the probability density functions for $b = 0, 1$ are given by f_b. The error probability of Algorithm with the score $R^{(\mathrm{E})}(\mathbf{b}, \mathbf{x})$ converges to zero as $L \to \infty$ if

$$\left(h\left(\frac{f_0 + f_1}{2} \right) - \frac{h(f_0) + h(f_1)}{2} \right) - \frac{D(f_0 || f_0^{(\mathrm{E})}) + D(f_1 || f_1^{(\mathrm{E})})}{2} > \frac{1}{5}. \quad (7)$$

Proof. A proof strategy is almost the same as that of Theorem 2 in [11]. The score $R^{(\mathrm{E})}(\mathbf{b}, \mathbf{x})$ is essentially equivalent to the score

$$R'^{(\mathrm{E})}(\mathbf{b}, \mathbf{x}) = \sum_i \log \frac{f_{b_i}^{(\mathrm{E})}(x_i)}{g(x_i)}$$

since $g(x_i)$ does not depend on $\mathbf{b}$. It is enough for proving the theorem to calculate

$$I^{(\mathrm{E})} := \sum_{b \in \{0,1\}} \frac{1}{2} \int_x \left(\log \frac{f_b^{(\mathrm{E})}(x)}{g(x)} \right) f_b(x) \mathrm{d}x.$$

The exact form of $I^{(\mathrm{E})}$ is calculated as follows.

$$\begin{aligned}
I^{(\mathrm{E})} &= \sum_{b \in \{0,1\}} \frac{1}{2} \int_x \left(\log \frac{f_b^{(\mathrm{E})}(x)}{g(x)} \right) f_b(x) \mathrm{d}x \\
&= -\int_x (\log g(x)) g(x) \mathrm{d}x + \frac{1}{2} \sum_{b \in \{0,1\}} \int_x \left(\log f_b^{(\mathrm{E})}(x) \right) f_b(x) \mathrm{d}x \\
&= h(g) + \frac{1}{2} \sum_{b \in \{0,1\}} \int_x \left(\log \frac{f_b^{(\mathrm{E})}(x)}{f_b(x)} f_b(x) \right) f_b(x) \mathrm{d}x \\
&= h(g) - \frac{1}{2} \sum_{b \in \{0,1\}} h(f_b) - \frac{1}{2} \sum_{b \in \{0,1\}} \int_x \left(\log \frac{f_b(x)}{f_b^{(\mathrm{E})}(x)} \right) f_b(x) \mathrm{d}x \\
&= h\left(\frac{f_0 + f_1}{2} \right) - \frac{h(f_0) + h(f_1)}{2} - \frac{D(f_0 || f_0^{(\mathrm{E})}) + D(f_1 || f_1^{(\mathrm{E})})}{2}
\end{aligned}$$

The rest of the proof is the same as that of Theorem 2 in [11]. Then, we have the theorem. □

The former half of the left hand side in Eq. (7), $h((f_0+f_1)/2) - (h(f_0)+h(f_1))/2$, is equivalent to the condition when the true distributions are

known (see Theorem 1). Its latter half $(D(f_0||f_0^{(\mathrm{E})}) + D(f_1||f_1^{(\mathrm{E})}))/2$ corresponds to the *information loss* or *penalty* caused by mis-estimations. From the definition, it is always non-negative. If the probability density function is correctly estimated (which means that the both of $f_0^{(\mathrm{E})} = f_0$ and $f_1^{(\mathrm{E})} = f_1$ hold), the information loss vanishes since $D(f_0||f_0^{(\mathrm{E})}) = D(f_1||f_1^{(\mathrm{E})}) = 0$. Conversely, if the accurate estimation fails, the success condition is much worse than expected due to the information loss caused by mis-estimation of f_0 and f_1.

4 New Score Function with a Priori Known Variances

In this section, we propose an effective score function when the noise distributions are unknown but their average and variances are a priori known. Note that we remove this requirement in Section 5. Our score function explicitly uses the values of the variances of the noise distributions. Specifically, the proposed score is much more effective than previous one when the variance of F_0 and F_1 are different.

First, we point out drawbacks of DPA-like algorithm introduced in [10]. The DPA-like algorithm works with only observed data even if the probability density functions are not known. From the nature of the DPA-like score, it can not use any other side information of probability density function such as variances even if they are available.

We try to incorporate the side information into the DPA-like function. It is natural to consider the weighted variant of DPA-like score, which is defined by

$$\text{w-DPA}(\mathbf{b}, \mathbf{x}) := \sum_{i=1}^{n} w_{b_i}(-1)^{b_i} x_i \tag{8}$$

for some kind of weights w_0 and w_1. The performance on weighted variant of DPA-like score heavily relies on how to set w_0 and w_1. If the observed value is *reliable*, the corresponding weight should be large. We propose a new score by following this idea.

4.1 New Score Function: Variance-based Score

We consider the case where F_0 and F_1 (and hence also f_0 and f_1) are unknown, but, their variances are known a priori. We denote by σ_0^2 and σ_1^2 the variances of F_0 and F_1. Under the situation, we have a chance to choose an adequate score function including the explicit values of the variances.

We introduce a new score function (Variance-based Score):

$$\mathrm{V}(\mathbf{b}, \mathbf{x}) := \sum_i \frac{(-1)^{b_i} x_i}{\sigma_{b_i}^2}. \tag{9}$$

It can be considered that $w_0 = 1/\sigma_0^2$ and $w_1 = 1/\sigma_1^2$ in the context of weighted variant of DPA. We denote Algorithm 2 employing Variance-based Score $\mathrm{V}(\mathbf{b}, \mathbf{x})$ as a score function by *V-based algorithm*. Note that in evaluating the score function by Eq. (9), we explicitly use the variances σ_0^2 and σ_1^2. Consider the case when $\sigma_0^2 = \sigma_1^2 = \sigma^2$. Then, the score function can be transformed into

$$\mathrm{V}(\mathbf{b}, \mathbf{x}) = \frac{\sum_i (-1)^{b_i} x_i}{\sigma^2} = \frac{1}{\sigma^2} \sum_i (-1)^{b_i} x_i = \frac{1}{\sigma^2} \mathrm{DPA}(\mathbf{b}, \mathbf{x}).$$

Since the part $1/\sigma^2$ does not affect the order of score value, we can ignore it and recover the DPA-like score. The Variance-based score then includes the DPA-like score in the special case.

Our strategy for designing a score function can be interpreted as follows: The observed data from the distribution with larger variance will not be reliable. Then, its contribution is set to be small if the variance is large, and vice versa. Appendix B explains how the Variance-based score is derived via estimation of distributions.

4.2 Theoretical Analysis for V-based algorithm

In this section, we discuss the success condition of the V-based algorithm for recovering the secret key. The following theorem shows the success condition on f_0 and f_1 when we use V-based algorithm for recovering the RSA secret key.

Theorem 4. Assume that the probability density function for $b = 0, 1$ are given by f_b. The error probability of the V-based algorithm converges to zero as $L \to \infty$ if

$$h\left(\frac{f_0 + f_1}{2}\right) - \log \sqrt{2\pi \mathrm{e} \sigma_0 \sigma_1} > \frac{1}{5}. \tag{10}$$

Proof. A proof is almost the same as the proof [11] of Theorem 2 in [10]. We denote by $f_b^{(\mathrm{G})}$ the probability density function of Gaussian distributions with average $(-1)^b$ and σ_b^2, respectively. The Variance-based score

is essentially equivalent to the score

$$R^{(\mathrm{G})}(\mathbf{b}, \mathbf{x}) = \sum_i \log \frac{f_{b_i}^{(\mathrm{G})}(x_i)}{g(x_i)}.$$

As the same discussion in [11], it is enough to calculate

$$I^{(\mathrm{G})} := \sum_{b\in\{0,1\}} \frac{1}{2} \int_x \left(\log \frac{f_b^{(\mathrm{G})}(x)}{g(x)} \right) f_b(x) \mathrm{d}x.$$

According to Theorem 1 in [10], the condition is given by $I^{(\mathrm{G})} > 1/5$. The exact form of $I^{(\mathrm{G})}$ is calculated as follows.

$$\begin{aligned}
I^{(\mathrm{G})} &= \sum_{b\in\{0,1\}} \frac{1}{2} \int_x \left(\log \frac{f_b^{(\mathrm{G})}(x)}{g(x)} \right) f_b(x) \mathrm{d}x \\
&= -\int_x (\log g(x)) g(x) \mathrm{d}x + \frac{1}{2} \sum_{b\in\{0,1\}} \int_x \left(\log f_b^{(\mathrm{G})}(x) \right) f_b(x) \mathrm{d}x \\
&= h(g) - \frac{1}{2} \sum_{b\in\{0,1\}} \left\{ \frac{\log(2\pi\sigma_b^2)}{2} + \frac{1}{2(\ln 2)\sigma_b^2} \int_x (x - (-1)^b)^2 f_b(x) \mathrm{d}x \right\} \\
&= h(g) - \log(\sqrt{2\pi \mathrm{e} \sigma_0 \sigma_1}).
\end{aligned}$$

Then, we have the theorem. □

We give a comparison between the DPA-like algorithm and the V-based algorithm. The difference between the left hand side of two inequalities: Eqs. (5) and (10) is given by

$$\log \sqrt{\pi \mathrm{e} (\sigma_0^2 + \sigma_1^2)} - \log \sqrt{2\pi \mathrm{e} \sigma_0 \sigma_1} = \frac{1}{2} \log \frac{\sigma_0^2 + \sigma_1^2}{2\sigma_0\sigma_1}.$$

Since the arithmetic mean is always larger than or equal to the geometric mean, it holds that $\frac{\sigma_0^2+\sigma_1^2}{2} \geq \sqrt{\sigma_0^2\sigma_1^2} = \sigma_0\sigma_1$. Then, the difference is always non-negative. Furthermore, the difference is 0 if and only if $\sigma_0 = \sigma_1$. It shows that V-based algorithm is superior to the DPA-like algorithm except the case that $\sigma_0 = \sigma_1$. As the ratio between σ_0 and σ_1 becomes larger, our improvement is more significant.

4.3 Optimality of Variance-based Score

We show that our proposed variance-based score is optimal in the framework of weighted variant of DPA-score. If we adopt w_0 and w_1 as weights, the success condition is given by

$$h\left(\frac{f_0+f_1}{2}\right) - \log\sqrt{2\pi\mathrm{e}} - \frac{\log \mathrm{e}}{4}(-\ln w_0 - \ln w_1 + \sigma_0^2 w_0 + \sigma_1^2 w_1 - 2) > \frac{1}{5}. \tag{11}$$

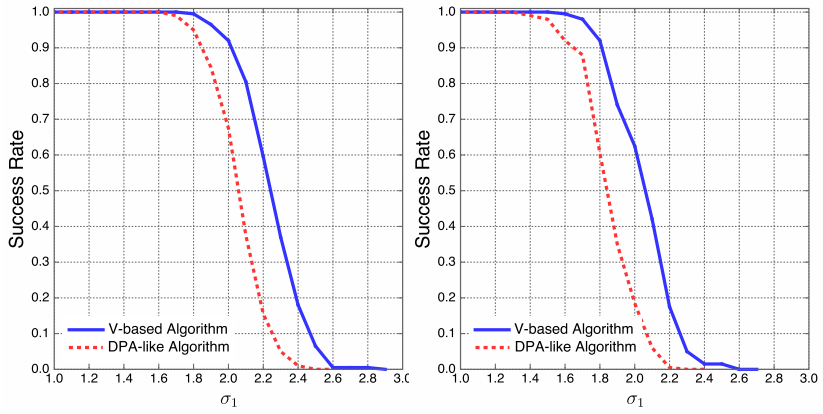
Denote the the left hand side of Eq. (11) by $H(w_0, w_1)$. By solving a simultaneous equation $\frac{\partial H}{\partial w_0} = \frac{\partial H}{\partial w_1} = 0$, we obtain $w_0 = 1/\sigma_0^2$ and $w_1 = 1/\sigma_1^2$, which maximizes $H(w_0, w_1)$. We recover the Variance-based score introduced in Section 4.1. This shows its optimality.

4.4 Experimental Results for V-based algorithm

We give experiment results on DPA-like algorithm [10] and our proposed V-based algorithm, which uses Eq. (9) as a score function. We implemented our algorithm in gcc with NTL 6.0, GMP 5.1.3 library and tested it on Intel Xeon 6-Core processor at 2.66 GHz with 32 GB memory. We set the public exponent to $e = 2^{16} + 1$. In all experiments shown in this section, we generated the output $\overline{\mathbf{sk}}$ for each $\mathbf{sk}$ from the Gaussian distribution. Denoting the correct secret bit in $\mathbf{sk}$ by b, we concretely generated $\overline{\mathbf{sk}}$ as follows: the observed value follows $\mathcal{N}(-1, \sigma_1^2)$ if $b = 1$; and the observed value follows $\mathcal{N}(+1, \sigma_0^2)$ if $b = 0$. In our experiments on 1024 bit RSA, we prepared 200 different tuples of secret keys $\mathbf{sk}$, e. g., $\mathbf{sk} = (p, q, d, d_p, d_q)$. We set a parameters L as $L = 2^{12}$.

We especially focus on the case where the $\sigma_1^2 \neq \sigma_0^2$. Figure 1 shows the success rates of DPA-like algorithm and V-based algorithm for $\sigma_0 = 0.4$ and $\sigma_0 = 1.0$. The vertical axis represents the success rates, and the horizontal axis shows the value of σ_1.

We give some discussion for the case of $\sigma_0 = 0.4$ from Fig. 1(a). When $\sigma_1 \leq 1.6$, the both algorithms succeed in recovering the secret key with success rate 1. Further, when $\sigma_1 \geq 2.6$, the both algorithms fail to do that for all trials. Meanwhile, when $1.7 \leq \sigma_1 \leq 2.5$, the two algorithms show the different behavior. For example, when $\sigma_1 = 2.1$, our algorithm recovers the secret key with success rate 0.8; while DPA-like algorithm recovers one with success rate 0.35. For another example, when $\sigma_1 = 2.4$, our algorithm recovers one with success rate 0.20; while DPA-like algorithm fails to recover the keys for all trials. These observations show that our V-based algorithm has superior performance to the DPA-like algorithm.



(a) $\sigma_0 = 0.4$ (b) $\sigma_0 = 1.0$

Fig. 1. Comparison between DPA-like and V-based algorithms

The running time for the V-based algorithm to find the secret key is at most 36.9 seconds under our computer circumstance for any cases.

Figure 2 shows the success rate of V-based algorithm for some typical values of σ_0 ($\sigma_0 = 0.1, 0.4, 0.7, 1.0$). This graph shows that for any values σ_1, the success rate will decrease if σ_0 increase.

5 Estimation of Variances by the EM algorithm

Our new score function $\mathrm{V}(\mathbf{b}, \mathbf{x})$ requires the additional inputs: variances σ_0^2 and σ_1^2 of F_0 and F_1. It is a significant disadvantage against the DPA-like algorithm. To solve this problem, we will use the help of the EM algorithm [1, 3] in estimating the variances from the observed data. The EM algorithm is a popular algorithm in the area of machine learning and is used to estimate hidden parameters of mixture distributions.

We will use the EM algorithm to estimate the variances σ_0^2 and σ_1^2 as a pre-processing of the V-based algorithm. That means, we first run the EM algorithm to estimate the variances and then run the V-based algorithm with the estimated variances to recover the secret key. It enables us to recover the secret key by using only the observed data, as well as the DPA-like algorithm. Unlike DPA-like algorithm, we succeed in taking account of the values of the variances in the combined algorithm. It can lead to a significant improvement of the bound for key-recovery against DPA-like algorithm, which will be examined in Section 5.2.

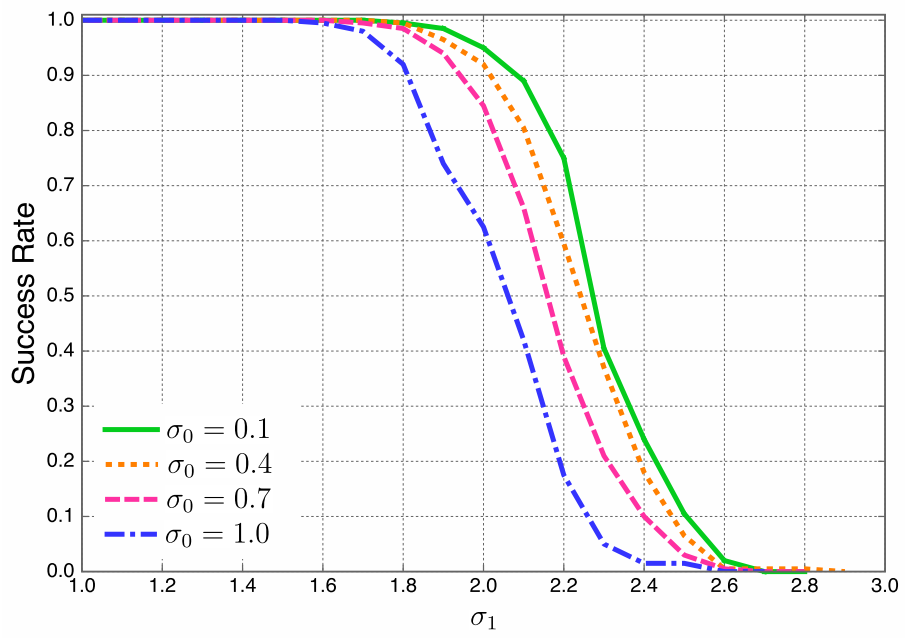


Fig. 2. Success Rates of V-based algorithm for $\sigma_0 = 0.1, 0.4, 0.7, 1.0$

5.1 Variance Estimation by the EM algorithms

Before giving the detailed explanation of the EM algorithm, we present another view of our noise model. It can be regarded as follows:

- The probability density functions $f_b(x; \boldsymbol{\theta_b})$ are defined by hidden parameters $\boldsymbol{\theta_b}$ for $b = 0, 1$.
- The observed value follows the mixture distribution $p(x)$ of f_0 and f_1, where $p(x) = \alpha f_0(x; \boldsymbol{\theta_0}) + (1 - \alpha) f_1(x; \boldsymbol{\theta_1})$ for $0 \leq \alpha \leq 1$.

In the usual setting in the EM algorithm, the form of f_0 and f_1 are known (say, f_0 is the Gaussian distribution, etc.), but, the set of parameters $\Theta = \{\alpha, \boldsymbol{\theta_0}, \boldsymbol{\theta_1}\}$ are a priori unknown (or hidden). The EM algorithm is usually used to estimate these parameters from the observed data.

We show the EM algorithm in more details. We denote by D a set of the observed values. Assume that all the observed value $x_i \in D$ follows the mixture density: $p(x) = \alpha_0 f_0(x; \boldsymbol{\theta_0}) + \alpha_1 f_1(x; \boldsymbol{\theta_1})$. We introduce Membership Weight γ_{ik} for $x_i \in D$ given parameters Θ as follows:

$$\gamma_{ik} = \frac{\alpha_k \bar{f}_k(x; \boldsymbol{\theta_k})}{\alpha_0 \bar{f}_0(x; \boldsymbol{\theta_0}) + \alpha_1 \bar{f}_1(x; \boldsymbol{\theta_1})} \tag{12}$$

for $1 \leq i \leq |D|$ and $k = 0, 1$. Note that $\alpha_0 + \alpha_1 = 1$ and $\alpha_0, \alpha_1 \geq 0$. Intuitively, the γ_{ik} corresponds to a probability that x_i comes from the bit k. If we know the exact form of f_k, we use f_k itself for $\bar{f}_k$ for $k = 0$ and 1. However, in our attack scenario, we have no knowledge about f_k as described before. Then, we cannot use the EM algorithm as-is. We use the Gaussian distribution in place of true unknown distribution, which enables the EM algorithm to work. We adopt the probability density function of Gaussian distribution for $\bar{f}_k = \mathcal{N}(\mu_k, \sigma_k^2)$. In this setting, the purpose of the EM algorithm will estimate means and variances for mixture distributions. Appendix C gives some discussion on a relation between cumulants and our estimation.

Next, we focus on our attack scenario. The attacker now wants to know the means μ_0 and μ_1, and variances σ_0^2 and σ_1^2 by using the EM algorithm. In this scenario, it is implicitly assumed that the noise distributions f_0 and f_1 are the Gaussian. Then, we can explicitly write Θ as $\Theta = \{\alpha_0, \alpha_1, \mu_0, \mu_1, \sigma_0, \sigma_1\}$. Algorithm 3 in Appendix A.2 shows the EM algorithm for estimating all the parameters Θ.

It is proved that the log-likelihood of the mixture distribution monotonically decreases by using the EM algorithm. On the other hand, it is hard to estimate precisely in advance the number of iteration required until the log-likelihood converges. We will verify that the computational time for the estimation phase for variances is negligible to the total time for the whole key-recovery by measuring an actual running time of the EM algorithm.

Algorithm 1 shows the whole proposed algorithm. This algorithm is composed of two phase: Parameter Estimation Phase and Key-Recovery Phase. That means we use Algorithm 3 as a pre-processing of the key-recovery algorithm. We call the whole algorithm KRP algorithm.

Algorithm 1 KRP algorithm (Key Recovery with Pre-processing Algorithm)

Input: Public Key (N, e), observed noisy sequences $\overline{\mathbf{sk}}$
Output: Correct Secret Key $\mathbf{sk}$
Parameter: $L \in \mathbb{N}$
Parameter Estimation Phase Run Algorithm 3 (EM algorithm) to estimate the variances σ_0^2 and σ_1^2 from the observed sequence.
Key-Recovery Phase: Run V-based algorithm with inputs (estimated) σ_0^2 and σ_1^2, the observed sequence, and L.

5.2 Experimental Results for KRP algorithm

We first examine the running time of Algorithm 3 for various input length of the observed sequence. We repeat the EM algorithm 100 times given an initial parameter for Θ and calculate the average of the running time. The environment for computation is the same as that in Section 4.4. In the experiments, we iterate E-step and M-step until convergence.

Table 1 shows the average time for the EM algorithm.

Table 1. Average of Running Time for the EM algorithm

Input Data Length	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
Computational Time (ms)	1.90	3.94	6.22	8.38	10.5	12.8	15.1	17.5	19.9	21.8

In general, we can estimate parameters with higher accuracy if we use more data for estimation. On the other hand, it causes more running time. We can see that the running time is at most 21.8ms even if we use full sequences for estimating the variances. Since the running time for the V-based algorithm is in average 36.9 seconds as shown in Section 4.4, the running time of the EM-algorithm is negligible to the whole running time. From now on, we ignore the running time of the EM Algorithm.

Figure 3 shows success rates of KRP algorithm for $\sigma_0 = 0.1, 0.4, 0.7$ and 1.0. The success rates for any σ_0 show similar behavior. Namely, if σ_1 is small (for example, if $\sigma_1 \leq 1.7$ for $\sigma_0 = 0.4$), the success rates are 1. If σ_1 is bigger than some threshold that depends on σ_0, the success rate decreases gradually. Further, if σ_1 is large enough (for example, if $\sigma_1 \geq 2.6$ for $\sigma_0 = 0.4$), the success rates are 0 for any σ_0.

Comparison between KRP algorithm and V-based algorithm Next, we compare KRP algorithm and V-based algorithm. Remember that V-based algorithm requires additional input: σ_0 and σ_1 but KRP algorithm works without them. In the experiments, we consider the case that both of f_0 and f_1 are the Gaussian distributions: $f_k = \mathcal{N}((-1)^k, \sigma_k^2)$ for $k = 0$ and 1. Here, we run the experiments under the same environments as in Section 4.4.

Figure 4 shows the success rates of the KRP algorithm and the V-based algorithm for $\sigma_0 = 0.4$ and 1.0. We give some discussion for the case of $\sigma_0 = 0.4$. Fig. 4(a) shows that the success rates of the both algorithms are almost 1 if σ_1 is less than or equal to 1.8. We can see that when $1.8 \leq \sigma_1 \leq 2.6$, their success rates decrease gradually, but, they are almost

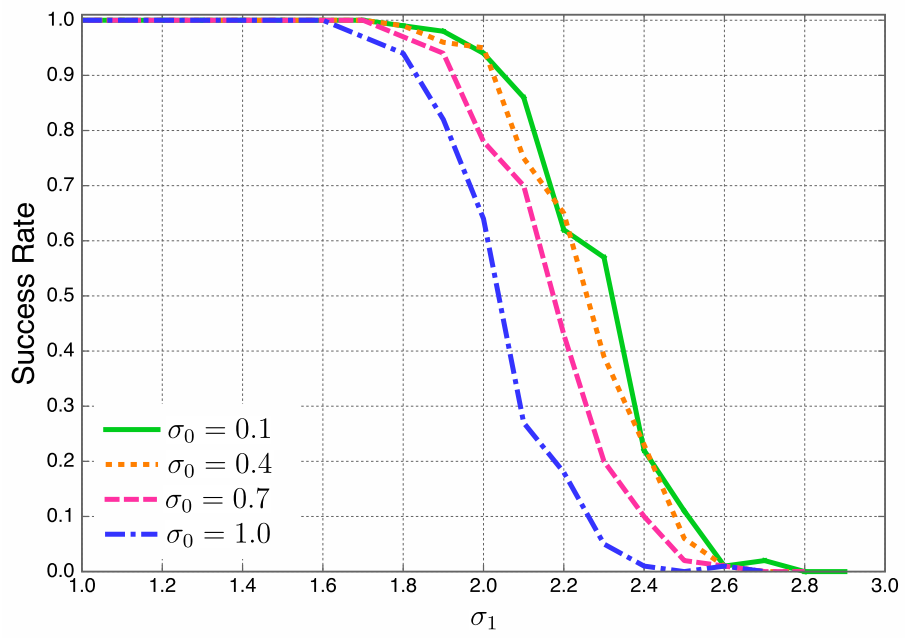


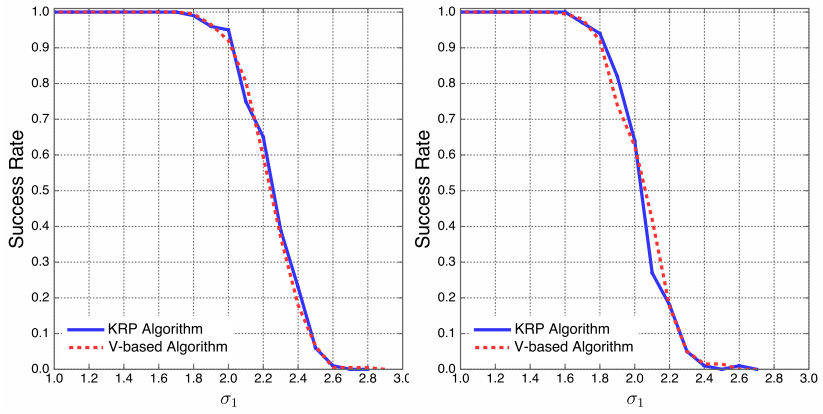
Fig. 3. Success rates of KRP algorithm for $\sigma_0 = 0.1, 0.4, 0.7, 1.0$

the same. Hence, we can say that there is no difference between their performance. The success rates for $\sigma_0 = 1.0$ denote the same tendency as for $\sigma_0 = 0.4$, that is, there is no difference in performance between the two algorithms. The above discussion shows that the EM algorithm succeeds in estimating the variances with enough accuracy and KRP algorithm. Consequently, the KRP algorithm, which does not receive σ_0 and σ_1 as inputs, has almost the same performance as the V-based algorithm.

Comparison between KRP and DPA-like algorithm Finally, we compare KRP and DPA-like algorithms [10]. Note that the both algorithms work given only the observed data, which means that they do not require additional information about the probability density functions.

We consider the case that the both of f_0 and f_1 are the Gaussian distributions: $f_k = \mathcal{N}((-1)^k, \sigma_k^2)$ for $k = 0$ and 1. Here, we execute the experiments under the same environments as in Section 4.4.

Figure 5 shows the success rates of KRP algorithm and DPA-like algorithm for $\sigma_0 = 0.4$ and $\sigma_0 = 1.0$. We can see that KRP algorithm attains higher success rates than the DPA-like algorithm from Figs. 5(a) and 5(b). Further, their computational time for recovering the keys are



(a) $\sigma_0 = 0.4$ (b) $\sigma_0 = 1.0$

Fig. 4. Success Rates of KRP algorithm and V-based algorithm

Table 2. Success Rates of Three Algorithms for $\sigma_0 = 0.4$

σ_1	$0 \cdots 1.6$	1.7	1.8	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8
DPA-like [10]	1	0.99	0.95	0.85	0.68	0.38	0.16	0.05	0.01	0	0	0	0
V-based (this paper)	1	1	1	0.97	0.92	0.81	0.60	0.37	0.18	0.07	0.01	0.01	0.01
KRP (this paper)	1	1	0.99	0.96	0.95	0.75	0.65	0.39	0.23	0.06	0.01	0	0

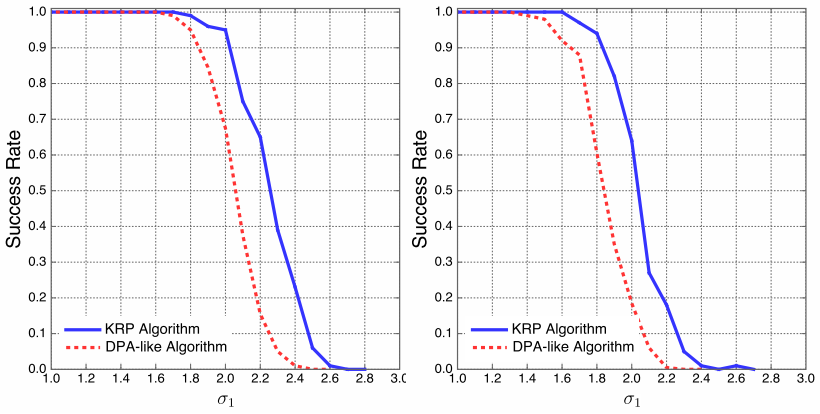
almost the same because the running time of the EM algorithm is negligible as described before. Summing up, we can conclude that our proposed KRP algorithm is superior to the DPA-like algorithm.

Tables 2, 3 and 4 summarize the success rates of the DPA-like algorithm, the V-based algorithm, and the KRP algorithm.

Table 3. Success Rates of Three Algorithms for $\sigma_0 = 1.0$

σ_1	$0 \cdots 1.3$	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1	2.2	2.3	2.4	2.5
DPA-like [10]	1	0.99	0.98	0.92	0.88	0.61	0.35	0.19	0.06	0.01	0	0	0
V-based (this paper)	1	1	1	1	0.98	0.92	0.74	0.63	0.42	0.18	0.05	0.02	0.02
KRP (this paper)	1	1	1	1	0.97	0.94	0.82	0.64	0.27	0.18	0.05	0.01	0

We can see that the proposed algorithms in this paper are superior to DPA-like algorithm [10]. Moreover, the proposed two algorithms have almost the same performance; while V-based algorithm requires the variances of the noise distributions and KRP algorithm does not.



(a) $\sigma_0 = 0.4$ (b) $\sigma_0 = 1.0$

Fig. 5. Success Rates of KRP algorithm and DPA-like algorithm

Table 4. Success Rates of Three Algorithms for $\sigma_0 = 1.5$

σ_1	$0 \cdots 0.8$	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1
DPA-like [10]	1	0.99	0.98	0.95	0.91	0.76	0.66	0.41	0.24	0.10	0.04	0.01	0	0
V-based (this paper)	1	1	1	1	1	0.99	0.94	0.87	0.78	0.55	0.37	0.17	0.09	0.01
KRP (this paper)	1	1	1	1	1	0.98	0.98	0.93	0.67	0.58	0.39	0.13	0.06	0

5.3 Performance of KRP Algorithm for non-Gaussian Distributions

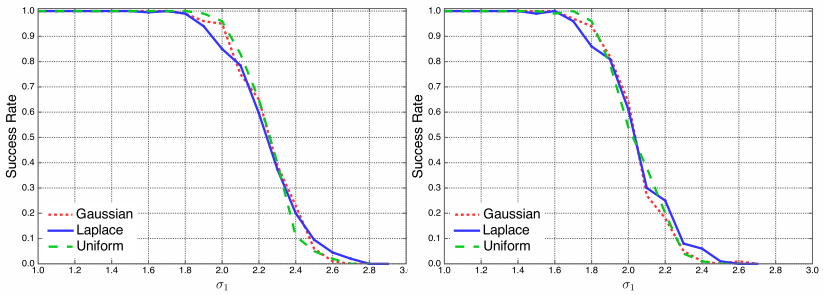
We have verified that our algorithms are effective when the true distributions are the Gaussian. As claimed in [7], the assumption of Gaussian noise may not always hold in practice. We will provide some evidence in this subsection that our algorithms are effective for *non-Gaussian distributions* We consider the case that the both of f_0 and f_1 are non-Gaussian distributions. As like [7], we will examine two non-Gaussian distributions. The first one comes from the exponential family, Laplace distribution. The second one is a uniform distribution, which is far from the Gaussian. The first has *sharp* form and the second has *flat* form compared to the Gaussian. It is important to study the performance under these two distributions in practice (see [7]).

We remind readers the Laplace distribution. The probability density function f_{L} for Laplace distribution $\mathcal{L}(\mu, a)$ with mean μ and a positive real a is given by $f_{\mathrm{L}}(x; \mu, a) = \frac{1}{2a} \exp\left(-\frac{|x-\mu|}{a}\right)$. The variance of the Laplace distribution $\mathcal{L}(\mu, a)$ is given by $2a^2$. We consider the case that $f_k = \mathcal{L}((-1)^k, a_k)$ for $k = 0$ and 1. The probability density function f_{U} for

Uniform distribution $\mathcal{U}(\mu, c)$ with mean μ and a positive real c is given by $f_{\mathrm{U}}(x; \mu, c) = \frac{1}{2c}$ when $-c + \mu \leq x \leq c + \mu$ and 0 otherwise. The variance of the Uniform distribution $\mathcal{U}(\mu, c)$ is given by $3c^2$. We consider the case that $f_k = \mathcal{U}((-1)^k, c_k)$ for $k = 0, 1$.

Figure 6 shows the success rates of KRP algorithm for the Gaussian, the Laplace, and the Uniform distributions with the same variances $\sigma_0 = 0.4$ and $\sigma_0 = 1.0$, respectively. Note that we set $a_i = \sigma_i/\sqrt{2}$ and $c_i = \sigma_i/\sqrt{3}$ to be the same variances. Also, note that we do not use any knowledge about true distributions at all in our experiments. We can see that KRP algorithm can attain almost the same (but slightly worse) success rates even if the true distribution is Laplace or Uniform distribution. These results give strong evidence that our algorithm works for any probability distribution.

One might think that we have showed the experimental results for only two non-Gaussian distributions. We again stress on some restrictions in our analysis. Our algorithms and analysis are valid only if the probability density functions f_0 and f_1 are *fixed* through the observation, which implies that our analysis is valid for any form of the noise distributions.



(a) $\sigma_0 = 0.4$ (b) $\sigma_0 = 1.0$

Fig. 6. Success Rates of KRP algorithm for Three Different Distributions

Acknowledgement

This research was supported by CREST, JST and supported by JSPS KAKENHI Grant Number 25280001 and 16H02780.

References

1. C. M. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006.

2. C. M. Cover and J. A. Thomas, "Elements of Information Theory, 2nd Edition," Wiley-Interscience, 2006.
3. A. P. Dempster and N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," in Journal of the Royal Statistical Society, Series B Vol. 39, no. 1, pp. 1–38, 1977.
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. A. Heuser, O. Rioul and S. Guilley, "Good is Not Good Enough: Deriving Optimal Distinguishers from Communication Theory," in Proc. of CHES2014, LNCS 8731, pp. 55–74, 2014.
8. P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," in Proc. of CRYPTO'99, LNCS 1666, pp.388–397, 1999.
9. N. Kunihiro, N. Shinohara and T. Izu, "Recovering RSA Secret Keys from Noisy Key Bits with Erasures and Errors," in Proc. of PKC2013, LNCS 7778, pp. 180–197, 2013.
10. N. Kunihiro and J. Honda, "RSA meets DPA: Recovering RSA Secret Keys from Noisy Analog Data," in Proc. of CHES 2014, LNCS 8731, pp. 261–278, 2014.
11. N. Kunihiro and J. Honda, "RSA meets DPA: Recovering RSA Secret Keys from Noisy Analog Data," IACR eprint: 2014/513, 2014.
12. N. Kunihiro and Y. Takahashi, "Improved Key Recovery Algorithms from Noisy RSA Secret Keys with Analog Noise," to appear in CT-RSA2017, 2017.
13. 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.
14. PKCS #1: RSA Cryptography Specifications Version 2.0. Available at `http://www.ietf.org/rfc/rfc2437.txt`.
15. 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.
16. 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 Algorithm Description

A.1 Key Recovery Algorithm in [10, 13]

Let $\boldsymbol{b}_{1,a} \in \{0,1\}^5$, $a \in \{1,2\}$, be the a-th candidate of the first slice $\mathbf{slice}(0)$. We write the two candidates of the first $(i+1)$ slices when the first i slices are $\boldsymbol{b}_{i,a} = (\mathbf{slice}(0), \cdots, \mathbf{slice}(i-1))$ by $\boldsymbol{b}_{i+1,2a-1}$, $\boldsymbol{b}_{i+1,2a} \in \{0,1\}^{5(i+1)}$. Similarly, for a secret key sequence $\boldsymbol{b}_{i,a}$, the observed sequence

Algorithm 2 Key Recovery Algorithm in [10]

Input: Public keys (N, e), observed data $\overline{\mathbf{sk}}$ (and some information about noise distributions if necessary)
Output: Secret keys $\mathbf{sk}$.
Parameter: $L \in \mathbb{N}$.
Initialization: Set $\mathcal{L}_0 := \{1\}$.
Loop: For $r = 1, 2, \cdots, n/2$ do the following.

1. **Expansion Phase** Generate list $\mathcal{L}'_r$ of all ancestors with depth 1 from nodes in $\mathcal{L}_{r-1}$, that is, $\mathcal{L}'_r := \bigcup_{a \in \mathcal{L}_{r-1}} \{2a - 1, 2a\}$.
2. **Pruning Phase** If $2^r \leq L$ then set $\mathcal{L}_r := \mathcal{L}'_r$. Otherwise, set $\mathcal{L}_r$ to a subset of $\mathcal{L}'_r$ with size L such that $\mathbf{Score}_{5r}(\boldsymbol{b}_{r,a}, \boldsymbol{x}_r)$ are the largest so that for any $a \in \mathcal{L}_r$ and $a' \in \mathcal{L}'_r \setminus \mathcal{L}_r$

$$\mathbf{Score}_{5r}(\boldsymbol{b}_{r,a}, \boldsymbol{x}_r) \geq \mathbf{Score}_{5r}(\boldsymbol{b}_{r,a'}, \boldsymbol{x}_r)\,.$$

Output of Loop: List of candidates $\mathcal{L}_{n/2}$.
Finalization: For each candidate in $\mathcal{L}_{n/2}$, check whether the candidate is indeed a valid secret key with the help of public information.

is denoted by $\boldsymbol{x}_i \in \mathbb{R}^{5i}$. Under the these notation, the key-recovery algorithm is summarized in Algorithm 2.

We say that the recovery error occurred if the output $\mathcal{L}_{n/2}$ does not contain the correct secret key.

The computational cost of Algorithm 2 is summarized as follows. The costs of Expansion and Pruning phases in each loop are evaluated by $2L$ and $2L$. Since each phase is repeated $n/2$ times, the total cost of the Expansion phase and Pruning phase are given by nL and nL, respectively.

A.2 The EM Algorithm for Estimating Θ

We briefly review the EM algorithm [1]. The EM algorithm is an iterative method for finding maximum likelihood in statistical models, where the model depends on hidden parameters. The EM iteration alternates between performing an expectation (E)-step and a maximization (M)-step. The E-step creates a function for the expectation of the log-likelihood evaluated using the current estimate of the parameters. Moreover, the M-step computes parameters maximizing the expected log-likelihood found on the E-step. These estimated parameters are then used to determine the distribution of the hidden parameters in the next E-step.

Algorithm 3 shows the EM algorithm for estimating parameters Θ from the observed data.

Algorithm 3 The EM Algorithm for Estimating Θ

Input: Observed data set $D = \{x_1, \ldots, x_n\}$ and initial values for Θ

Output: Parameter Θ maximizing the log-likelihood of $p(D|\Theta)$.

Iteration: Repeat E-Step and M-Step until the log-likelihood of the mixture density converges.

E-Step: Compute the Membership weight $\gamma(i,k)$ for all $x_i \in D$.

M-Step: Obtain the updated state estimates $N_k, \mu_k^{\text{new}}, \sigma_k^{\text{new}}, \alpha_k^{\text{new}}$ for $k = 0$ and 1 via the following equations.

$$N_k = \sum_{i=1}^{N} \gamma(i,k) \quad \text{for } k = 0, 1$$

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(i,k) x_i \quad \text{for } k = 0, 1$$

$$\sigma_k^{2\,\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(i,k)(x_i - \mu_k^{\text{new}})^2 \quad \text{for } k = 0, 1$$

$$\alpha_k^{\text{new}} = \frac{N_k}{N} \quad \text{for } k = 0, 1$$

B V-based algorithm via Estimation of Distributions

We will give another view of V-based algorithm. In deriving a new score function, we go back to Eq. (6). In the discussion of Section 3 (especially, Theorem 3), it is critical to estimate $f_0^{(\mathrm{E})}$ and $f_1^{(\mathrm{E})}$ more precisely in designing the score function. But, how can we do? We will use the Gaussian distribution with variance σ_0^2 and σ_1^2 in place of unknown (but true) probability density functions f_0 and f_1 for their estimations $f_0^{(\mathrm{E})}$ and $f_1^{(\mathrm{E})}$. Substituting $f_0^{(\mathrm{E})} = \mathcal{N}(+1, \sigma_0^2)$ and $f_1^{(\mathrm{E})} = \mathcal{N}(-1, \sigma_1^2)$ into Eq. (6), we have

$$R^{(\mathrm{E})}(\mathbf{b}, \mathbf{x}) = \sum_i \left(\frac{(-1)^{b_i} x_i}{\sigma_{b_i}^2} \right) - \sum_i \left(\frac{x_i^2 + 1}{2\sigma_{b_i}^2} + \frac{1}{2} \log(2\pi\sigma_{b_i}^2) \right) \tag{13}$$

We will adopt its former part as a new score function since the terms in latter part of Eq. (13): $\sum_i \left(\frac{x_i^2+1}{2\sigma_{b_i}^2} + \frac{1}{2} \log(2\pi\sigma_{b_i}^2) \right)$ does not (almost) depend on the sequence $\boldsymbol{b}$ in the case that each b_i is randomly generated.

C Cumulants

The main technical tools in the paper is estimating the average and the variance of the true noise distributions. Theorem 3 claims that we can

improve the success condition if we succeed to precisely estimate the probability density functions. For better understanding, we review *cumulant* as the useful statistic.

The cumulant generating function $M(s)$ for the random variable X, which is the logarithm of the moment generating function, is defined by $M(s) = \ln \mathrm{E}[\mathrm{e}^{sX}]$. The cumulants c_j are obtained from a power series expansion of the cumulant generating function:

$$M(s) = \sum_{j=1}^{\infty} \frac{c_j}{j!} s^n$$

We consider the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. The cumulant generating function $M(s)$ is given by $M(s) = \mu s + \frac{\sigma^2}{2!} s^2$. Then, $c_1 = \mu, c_2 = \sigma^2$ and $c_j = 0$ for $j = 3, 4, \ldots$. We can verify it by following calculations.

$$\begin{aligned}
\mathrm{E}[\mathrm{e}^{sX}] &= \int_{-\infty}^{\infty} \mathrm{e}^{sx} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-(\mu+\sigma^2 s))^2 - 2\mu\sigma^2 s - \sigma^4 s^2}{2\sigma^2}\right) \mathrm{d}x \\
&= \exp\left(\mu s + \frac{\sigma^2}{2!} s^2\right)
\end{aligned}$$

Estimating precise probability density function is equivalent to obtaining all precise values of cumulants. In the discussion of Section 5, we conduct the attack after calculating c_1 and c_2. It is enough for the Gaussian distribution since its cumulants satisfy $c_j = 0$ for $j \geq 3$.