Theoretical Security Evaluation against Side-channel Cube Attack with Key Enumeration

Haruhisa Kosuge and Hidema Tanaka

National Defence Academy of Japan, Yokosuka, Japan
{ed16005,hidema}@nda.ac.jp

Abstract. Side-channel cube attack (SCCA) is executed in a situation where an adversary can access some information about the internal states of the cipher. The adversary can obtain a system of linear equations by a set of chosen plaintexts called cube and recover the secret key using the system. Error tolerance is a challenging task in SCCA. To recover the secret key based on likelihoods under an error-prone environment, we propose SCCA with key enumeration (SCCA-KE). Precise likelihoods are computed to obtain lists for sub-key candidates and an optimal list for the complete key candidate is generated by key enumeration. Then, we propose an evaluation method for SCCA-KE which includes information-theoretic evaluation and experimental evaluation by rank estimation. We apply the proposed evaluation method to PRESENT and show some conditions required to thwart SCCA-KE in realistic assumptions. Using the evaluation method, the evaluator can consider countermeasures with a sufficient security margin.

Keywords: Block cipher, Side-channel attack, Side-channel cube attack, Key enumeration, Rank estimation, Guessing entropy, PRESENT

1 Introduction

1.1 Background

Conventionally, security of block ciphers is analyzed under an assumption that an adversary is given a black-box access to cryptosystems. However, security of cryptographic devices against side-channel attacks can not be assured by the analysis. In contrast to side-channel attacks to recover the secret key directly from leaked values (internal states of the cipher) [15], some attacks which exploit an algebraic structure of block cipher (algebraic attacks) are proposed [25][30]. Side-channel cube attack (SCCA) is also an algebraic attack proposed by Dinur and Shamir [10]. SCCA can overcome protections on the first round function such as random delay [31], since the attack exploits internal states passing through multiple round functions. Also, SCCA can recover the secret key under an assumption that the adversary can access a small information of bits of internal states. In this paper, we study SCCA as a theoretical evaluation. In the evaluation, it is assumed that the adversary can access only 1-bit information of internal states. Even in such condition, the adversary can recover the

secret key using a set of chosen plaintexts called *cube*. Obviously there are more advantageous bits (containing many information on the secret key) than other ones. As an evaluator, it is important to locate such advantageous bits. Then, the evaluator can take countermeasures efficiently.

In SCCA, a measured value is represented by a multivariate polynomial with variables of secret-key and plaintext bits. From the polynomial, the adversary attempts to obtain a linear equation by a cube. Using multiple cubes, he can obtain a system of linear equations. Cubes can be searched by computer experiments (*cube search*). In an error-free environment, he can recover the secret key if the number of independent equations is more than or equal to the one of the variables (secret-key bits).

If the measurements are prone to error, the analysis becomes complicated. There are two error models which suppose binary erasure channel (BEC model) [10] and binary symmetric channel (BSC model) [17]. In this paper, we investigate SCCA under BSC model. BSC model assumes that a measured value is different from the correct value with a crossover probability ρ. In CHES2013, Li et al. proposed an attack using maximum likelihood decoding (ML decoding) [17]. We call the attack *previous method*.

1.2 Contribution

First, we propose a new algorithm, SCCA with *key enumeration* [28] (SCCA-KE), which takes divide-and-conquer strategy (DC strategy) in a similar way to the previous method. In the previous method, lists for sub-key candidates (*sub lists*) are sorted by hamming distance from measured values; however, such sub lists are not enumerated based on precise likelihoods (the method is not based on ML decoding). In SCCA-KE, precise likelihoods are computed to construct optimal sub lists. Then, an optimal list for the secret-key (complete key) candidates is generated from the sub lists by key enumeration [28]. Using the optimal list, the number of times to test candidates is minimized.

Next, we propose an evaluation method for SCCA-KE. In the beginning, brute-force search for cubes of relatively small sizes is executed. Based on the searched cubes, two evaluations are executed, information-theoretic evaluation and experimental evaluation by *rank estimation* [29]. The former evaluation is to obtain a lower bound of *guessing entropy* which is equal to an expected time complexity. The latter evaluation is to obtain a success rate of SCCA-KE. Rank estimation is an evaluation method to estimate a time complexity of key enumeration. Note that we use a rank estimation algorithm proposed by Glowacz et al. [13] for its efficiency and preciseness. We set t_{adv} as the number of secret-key candidates which the adversary can test (it indicates his computing power). If a lower bound of guessing entropy is sufficiently larger than t_{adv}, the evaluator can assure security. Also, t_{adv} is a threshold in the experimental evaluation. If a rank is lower than t_{adv}, the attack is regarded as successful. We execute the experiment a sufficient number of times to obtain a reliable success rate.

We apply the proposed evaluation method to PRESENT [6]. In the evaluations, we set $t_{adv} = 2^{60}$ and the number of leaked values is $q_{adv} = 2^{15}$ at most.

We consider three *leakage models*, single bit [32], least significant bit (LSB) of hamming weight (HW) of 8-bit internal state [17] and LSB of HW of 4-bit internal state. Note that HW is in binary representation and 4 or 8-bit internal state corresponds to a register size. We obtain a lower bound of guessing entropy and success rate by changing crossover probability ρ.

As a result of the evaluation, we conclude that intensive protections for the second and third round are required to thwart SCCA-KE under $t_{adv} = 2^{60}$ and $q_{adv} = 2^{15}$. If it is difficult, noise insertion to achieve $\rho \geq 0.4$ is required. In this way, the evaluation method contributes to an efficient evaluation for SCCA in various settings, and the evaluator can consider a countermeasure with a sufficient security margin.

Notations. We use bold fonts for vectors, sans serif ones for functions and calligraphic ones for sets or lists (list is a set of elements with order).

2 Side-channel cube attack

2.1 Outline of side-channel cube attack

A side-channel leakage is represented by a nonlinear multivariate polynomial whose variables are plaintext and secret-key bits (we call *plaintext* and *secret-key variables*). The adversary can obtain a linear equation by a set of chosen plaintexts called *cube*. If he has a sufficient number of linear equations whose terms include secret-key variables, he can recover their values with trivial complexity in an error-free environment. Note that it is not trivial if measurements are prone to error (see Sec. 2.2).

Let $\mathsf{f} : \mathbb{F}_2^M \times \mathbb{F}_2^N \to \mathbb{F}_2$ be a multivariate polynomial of plaintext variables $\mathbf{v} = (v_1, v_2, ..., v_M) \in \mathbb{F}_2^M$ and secret-key variables $\mathbf{k} = (k_1, k_2, ..., k_N) \in \mathbb{F}_2^N$. We define a cube by an index set of plaintext variables $\mathcal{I} \subset \{1, 2, ..., M\}$. Using a term $\prod_{i \in \mathcal{I}} v_i$ called *maxterm*, f is divided into two polynomials:

$$\mathsf{f}(\mathbf{v}, \mathbf{k}) = \left(\prod_{i \in \mathcal{I}} v_i \right) \cdot \mathsf{q}_{\mathcal{I}}(\mathbf{v}, \mathbf{k}) + \mathsf{r}(\mathbf{v}, \mathbf{k}), \tag{1}$$

where $\mathsf{q}_{\mathcal{I}}$ is called *superpoly* of $\mathcal{I}$, and r is a polynomial in which there is no term divisible by $\prod_{i \in \mathcal{I}} v_i$.

We denote a cube by $\mathcal{C}_{\mathcal{I}}$ such that plaintext variables indexed by $\mathcal{I}$ take all possible combinations and the other plaintext variables are 0 (constant), e.g., $\mathcal{C}_{\{0,1\}} = \{(0, 0, ..., 0), (1, 0, ..., 0), (0, 1, ..., 0), (1, 1, ..., 0)\}$. Note that we call $|\mathcal{I}|$ as *cube size*. Then, we have [9]:

$$\mathsf{q}'_{\mathcal{I}}(\mathbf{k}) = \bigoplus_{\mathbf{v} \in \mathcal{C}_{\mathcal{I}}} \mathsf{f}(\mathbf{v}, \mathbf{k}), \tag{2}$$

where $\mathsf{q}'_{\mathcal{I}}(\mathbf{k}) = \mathsf{q}_{\mathcal{I}}(\mathbf{v}, \mathbf{k})$ such that any plaintext variable included in $\mathsf{q}'_{\mathcal{I}}(\mathbf{v}, \mathbf{k})$ is 0, e.g., if $\mathsf{q}_{\mathcal{I}}(\mathbf{v}, \mathbf{k}) = k_1 k_2 \oplus v_1 k_3$ and $1 \notin \mathcal{I}$ then $\mathsf{q}'_{\mathcal{I}}(\mathbf{k}) = k_1 k_2$. In this way, a value of right-hand side (RHS) of a superpoly $\mathsf{q}'_{\mathcal{I}}$ is obtained by the summation of $\mathsf{f}(\mathbf{v}, \mathbf{k})$. If the following two conditions hold, the adversary can use a superpoly $\mathsf{q}'_{\mathcal{I}}$ for key recovery.

1. $\mathsf{q}'_{\mathcal{I}}$ is not a constant polynomial: If $\mathsf{q}'_{\mathcal{I}}(\mathbf{k})$ is constant for a sufficient large number of randomly chosen $\mathbf{k}$, $\mathsf{q}'_{\mathcal{I}}$ is regarded as a constant polynomial and rejected.
2. $\mathsf{q}'_{\mathcal{I}}$ is a linear polynomial: Linearity test known as BLR test [4] is used. When the following equation holds for a sufficient large number of randomly chosen $\mathbf{k}^o$ and $\mathbf{k}^{o'}$, $\mathsf{q}'_{\mathcal{I}}$ is regarded as a linear function and accepted.

$$\mathsf{q}'_{\mathcal{I}}(\mathbf{k}^o) \oplus \mathsf{q}'_{\mathcal{I}}(\mathbf{k}^{o'}) \oplus \mathsf{q}'_{\mathcal{I}}(\mathbf{0}) = \mathsf{q}'_{\mathcal{I}}(\mathbf{k}^o \oplus \mathbf{k}^{o'}), \tag{3}$$

where $\mathbf{0}$ denotes a secret-key whose values are all 0.

We call the tests to check the above conditions as *cube test* and $\mathsf{q}'_{\mathcal{I}}(\mathbf{k})$ is generally computed for 100 times to obtain reliable cubes [10].

Let $x_{\mathbf{v}} = \mathsf{f}(\mathbf{v}, \mathbf{k})$ be a leaked value. Suppose that the adversary obtains a system of L linear equations by using L cubes as follows.

$$\begin{aligned} a_{1,1} \cdot k_1 \oplus a_{1,2} \cdot k_2 \oplus ... \oplus a_{1,n} \cdot k_n &= x_1 \\ a_{2,1} \cdot k_1 \oplus a_{2,2} \cdot k_2 \oplus ... \oplus a_{1,n} \cdot k_n &= x_2 \\ &\vdots \\ a_{L,1} \cdot k_1 \oplus a_{L,2} \cdot k_2 \oplus ... \oplus a_{L,n} \cdot k_n &= x_L \end{aligned} \tag{4}$$

Note that $a_{i,j} \in \mathbb{F}_2$ ($i \in \{1, 2, ..., L\}$, $j \in \{1, 2, ..., n\}$) is a coefficient of i-th linear equation. A RHS value x_i is obtained by $x_i = \bigoplus_{\mathbf{v} \in \mathcal{C}_{\mathcal{I}_i}} x_{\mathbf{v}} \oplus \mathsf{q}'_{\mathcal{I}_i}(\mathbf{0})$ $(= \mathsf{q}'_{\mathcal{I}_i}(\mathbf{k}) \oplus \mathsf{q}'_{\mathcal{I}_i}(\mathbf{0}))$, where $\mathsf{q}'_{\mathcal{I}_i}(\mathbf{0})$ is a constant. If the number of independent equations is more than or equal to the one of secret-key variables n, $\mathbf{k}$ is uniquely recovered.

Generally, not all secret-key variables are obtained from a system of linear equations, since there are remaining variables for which the adversary has no information. Let $\mathbf{sk}$ $(= \mathbf{k}||\mathbf{rk})$ and $\mathbf{rk}$ be the complete N-bit variables and n'-bit remaining variables, respectively ($N = n + n'$ is the secret-key length).

2.2 Error-tolerant side-channel cube attack

In SCCA, there are many works which assume that measurements are error-free [1][11][14][32][33]; therefore, error tolerance is not well studied. This is the drawback of the entire study for SCCA. If measurements are prone to error, SCCA is not always successful even if the number of independent equations are enough. BEC model [10] and BSC model [17] were proposed for the study considering error tolerance.

In BEC model, each measurement outputs a problematic measurement $\perp$ instead of 0/1 with probability ϵ. Then, the adversary assigns a new variable to each unknown values of $\perp$, and the number of variables increases and is expected to be $n + \sum_{i=1}^{L} \epsilon \cdot |\mathcal{C}_{\mathcal{I}_i}|$ ($|\mathcal{C}_{\mathcal{I}_i}| = 2^{|\mathcal{I}_i|}$). If the number of independent equations is more than or equal to the number of variables, n-bit secret-key variables are uniquely obtained. In this way, SCCA under BEC model can be error tolerant by increasing the number of independent equations.

In BSC model, each measured value $y_{\mathbf{v}}$ follows crossover probability $\rho = Pr[x_{\mathbf{v}} \oplus y_{\mathbf{v}} = 1] = 1/2 - \mu$ ($x_{\mathbf{v}}$ is a correct value). Using piling-up lemma [23], crossover probability for $x_i = \bigoplus_{\mathbf{v} \in \mathcal{C}_{\mathcal{I}_i}} x_{\mathbf{v}} \oplus \mathsf{q}'_{\mathcal{I}_i}(\mathbf{0})$ is obtained as [17]:

$$p_i = Pr[x_i \oplus y_i = 1] = \frac{1}{2} - 2^{|\mathcal{C}_{\mathcal{I}_i}|-1}\mu^{|\mathcal{C}_{\mathcal{I}_i}|}, \tag{5}$$

where $y_i = \bigoplus_{\mathbf{v} \in \mathcal{C}_{\mathcal{I}_i}} y_{\mathbf{v}} \oplus \mathsf{q}'_{\mathcal{I}_i}(\mathbf{0})$ is obtained by summation of measured values of $y_{\mathbf{v}}$. Note that we call $\mathbf{x}$ (resp. $\mathbf{y}$) a vector of correct (resp. measured) RHS values.

2.3 Previous method for BSC model [17]

In order to recover n-bit secret-key variables from a system of linear equations of Eq. (4) under BSC model with crossover probability p_i ($i \in \{1, 2, ..., L\}$), the authors of [17] proposed a method based on ML decoding (previous method). The key-recovery problem can be reduced to a decoding problem of [L,n] linear block code following the idea of Siegenthaler's cryptanalysis of stream ciphers [26].

Let $A = (a_{i,j})$ ($i \in \{1, 2, ..., L\}$, $j \in \{1, 2, ..., n\}$) be $L \times n$ matrix (a generator matrix of [L,n] linear block code), and $\mathbf{a}_i$ be i-th column vector of A. A correct RHS value is obtained by $x_i = \mathbf{k} \cdot \mathbf{a}_i$, and the adversary is given y_i. We denote a system of linear equations by $(A, \mathbf{k}, \mathbf{x})$ or $(A, \mathbf{k}, \mathbf{y})$. In the previous method, the adversary selects a key candidate $\mathbf{k}^o$ based on the hamming distance $\mathsf{HD}(\mathbf{k}^o)$ as:

$$\arg\min_{\mathbf{k}^o} \mathsf{HD}(\mathbf{k}^o), \ \ \mathsf{HD}(\mathbf{k}^o) = \sum_{i=1}^{L} (x_i^o \oplus y_i), \tag{6}$$

where $x_i^o = \mathbf{k}^o \cdot \mathbf{a}_i$. If a crossover probability p_i is constant for any $i \in \{1, 2, ..., L\}$, this decoding algorithm is ML decoding which has the smallest error probability. This algorithm is not ML decoding if p_i is different for $|\mathcal{C}_{\mathcal{I}_i}|$ (see detail in Sec. 3.1).

Since a general decoding problem is NP-complete [5], time complexity increases exponentially on n. If n is large, it is intractable for the adversary to execute ML decoding. In order to reduce time complexity, the previous method takes DC strategy to divide $\mathbf{k}$ into η sub keys $\{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_\eta\}$. Gathering row vectors of A whose variables include at least a variable of $\mathbf{k}_j$, a matrix A_j is defined and a vector of correct (resp. measured) RHS is $\mathbf{x}_j$ (resp. $\mathbf{y}_j$). Hereinafter, $(A_j, \mathbf{k}_j, \mathbf{x}_j)$ or $(A_j, \mathbf{k}_j, \mathbf{y}_j)$ is referred to as *sub system*. The adversary computes hamming distances in each matrix A_j and secret-key variables of sub key $\mathbf{k}_j$.

In order to improve success rate, the method chooses τ_{const} candidates in ascending order by $\mathsf{HD}(\mathbf{k}_j)$ in each sub system of linear equations. Let $\mathcal{T}_{\mathbf{k}_j}$ be a sub list of τ_{const} candidates for $\mathbf{k}_j$. Since $\mathbf{k}$ is reconstructed from η sub keys, the adversary has $(\tau_{const})^\eta$ candidates after the procedures, and we denote a set of the candidates by $\mathcal{T}_{\mathbf{k}} (= \mathcal{T}_{\mathbf{k}_1} \times \mathcal{T}_{\mathbf{k}_2} \times ... \times \mathcal{T}_{\mathbf{k}_\eta})$. Also, there are $2^{n'}$ candidates for the remaining variables $\mathbf{rk}$. We denote a list for $\mathbf{rk}$ as $\mathcal{T}_{\mathbf{rk}}$. The correct key is searched

from the candidates of $\mathcal{T}_{\mathbf{k}} \times \mathcal{T}_{\mathbf{rk}}$ ($|\mathcal{T}_{\mathbf{k}} \times \mathcal{T}_{\mathbf{rk}}| = (\tau_{const})^{\eta} \cdot 2^{n'}$), by using some actual plaintext-ciphertext pairs. Hence, it requires a time complexity $(\tau_{const})^{\eta} \cdot 2^{n'}$.

In order to reduce time complexity, the authors recommend to use overlapping sub keys where each vector shares 3 or 4 variables with neighboring vectors. We call the strategy *overlapping* DC strategy. The authors claim that it is possible to reduce time complexity since only candidates that agree in the overlapping secret-key variables are tested in overlapping DC strategy.

3 Side-channel cube attack with key enumeration

We consider SCCA-KE under BSC model. Note that SCCA-KE is applicable if a conditional probability distribution of leaked values given a key candidate, i.e., $Pr[\mathbf{y}|\mathbf{k}]$, is approximated. In this paper, we assume that leaked values are independent and a crossover probability $\rho = Pr[y_i \oplus x_i = 1]$ defines the distribution. As an example to use other models, we show how to obtain a conditional probability (crossover probability) by converting HW model with Gaussian noise to BSC model in Appendix A. Note that asymmetric crossover probability ($Pr[y_i = 0|x_i = 1] \neq Pr[y_i = 1|x_i = 0]$) can be handled by SCCA-KE. In practice, the adversary measures leakages from training devices and approximates the distribution [27].

3.1 Divide-and-conquer strategy and key enumeration

Let $\mathcal{T}_{\mathbf{sk}} = \{\mathbf{sk}^1, \mathbf{sk}^2, ..., \mathbf{sk}^{\tau}\}$ be a list of secret-key candidates which can be obtained by measurements. The adversary tests secret-key candidates from $\mathbf{sk}^1$ to $\mathbf{sk}^{\tau}$ and disregard other candidates not in $\mathcal{T}_{\mathbf{sk}}$, where τ is determined by his computing power. Likelihood or posterior probability is used to make the list $\mathcal{T}_{\mathbf{sk}}$. We get the same list if a uniform prior distribution can be assumed [19]. In this paper, we use log likelihood for the convenience of implementation.

Taking DC strategy, we use key enumeration to obtain $\mathcal{T}_{\mathbf{sk}}$ from η sub lists $\{\mathcal{T}_{\mathbf{k}_1}, \mathcal{T}_{\mathbf{k}_2}, ..., \mathcal{T}_{\mathbf{k}_\eta}\}$ and a sub list for the remaining variables $\mathcal{T}_{\mathbf{rk}}$. Since there is no information on $\mathbf{rk}$, candidates of $\mathcal{T}_{\mathbf{rk}}$ are randomly sorted (all candidates have the same log likelihood). SCCA-KE uses the following *optimal list* for $\mathbf{k}$.

Definition 1 (optimal list). *Let $\boldsymbol{y}$ be a vector of RHS values of a system of linear equations obtained by independent measurements. In the list of candidates $\mathcal{T}_{\boldsymbol{k}} = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_\tau\}$, the list $\mathcal{T}_{\boldsymbol{k}}$ is an optimal list of key candidates if the following conditions hold.*
For any (o, o') such that $\mathbf{k}^o, \mathbf{k}^{o'} \in \mathcal{T}_{\mathbf{k}}$ and $o < o'$,

$$\sum_{i=1}^{L} \log_2(Pr[y_i|\mathbf{k}^o]) \geq \sum_{i=1}^{L} \log_2(Pr[y_i|\mathbf{k}^{o'}]), \tag{7}$$

and for any (o, o'') such that $\mathbf{k}^o \in \mathcal{T}_{\mathbf{k}}$, $\mathbf{k}^{o''} \notin \mathcal{T}_{\mathbf{k}}$,

$$\sum_{i=1}^{L} \log_2(Pr[y_i|\mathbf{k}^o]) \geq \sum_{i=1}^{L} \log_2(Pr[y_i|\mathbf{k}^{o''}]). \tag{8}$$

Note that $Pr[y_i|\mathbf{k}^o] = Pr[x_i^o \oplus y_i]$, where $\mathbf{x}^o = (x_1^o, x_2^o, ..., x_L^o) = \mathbf{k}^o \cdot A$. The optimal list is in descending order by log likelihood and all candidates has higher log likelihoods than any candidates not in the optimal list. Obviously, the optimal list can function as ML decoding.

We construct η optimal sub lists $\{\mathcal{T}_{\mathbf{k}_1}, \mathcal{T}_{\mathbf{k}_2}, ..., \mathcal{T}_{\mathbf{k}_\eta}\}$. Note that there is no overlapping variables among sub keys (overlapping DC strategy is not taken). Then, a log likelihood of $\mathbf{k}^o = \mathbf{k}_1^o||\mathbf{k}_2^o||...||\mathbf{k}_\eta^o$ is obtained by:

$$\sum_{j=1}^{\eta}\sum_{i=1}^{L_j} \log_2(Pr[y_{j,i}|\mathbf{k}_j^o]), \tag{9}$$

where L_j is the number of linear equations of j-th sub system. Note that $\mathcal{T}_{\mathbf{rk}}$ is always optimal, since there is no information on $\mathbf{rk}$. Therefore, $\mathcal{T}_{\mathbf{sk}} = \mathcal{T}_{\mathbf{k}} \times \mathcal{T}_{\mathbf{rk}}$ is an optimal list if $\mathcal{T}_{\mathbf{k}}$ is optimal.

In the previous method [17], $\mathcal{T}_{\mathbf{k}}$ is not optimal as follows.

1. If a crossover probability p_i is different for $|\mathcal{C}_{\mathcal{I}_i}|$, a list sorted by hamming distance (see Eq. (5)) is not optimal. In other word, this method is not based on ML decoding. In general, the adversary exploits cubes of various sizes.
2. Since there may be a candidate not in $\mathcal{T}_{\mathbf{k}}$ such that its likelihood is more than some elements of $\mathcal{T}_{\mathbf{k}}$, $\mathcal{T}_{\mathbf{k}}$ is not optimal. In the previous method, the adversary chooses τ_{const} key candidates as $\mathcal{T}_{\mathbf{k}_j}$ for any $j \in \{1, 2, ..., \eta\}$. Then, a likelihood of the last candidate of $\mathcal{T}_{\mathbf{k}}$ is not maximized.
3. Since log likelihoods for η sub systems are not independent in the overlapping DC strategy, a total log likelihood is not calculated by simple summation such as Eq. (9). To calculate it, log likelihoods for overlapping subkey candidates should be subtracted after the summation. No existing key-enumeration algorithm support the calculation.

3.2 Proposed algorithm

We show a proposed algorithm for SCCA-KE in Algorithm 1. Note that we use a notation $(\mathbf{k}^o, \lambda_{\mathbf{k}^o}) \in \mathcal{U}_{\mathbf{k}}$ instead of $\mathbf{k}^o \in \mathcal{T}_{\mathbf{k}}$, since log likelihoods are used in key enumeration (see Eq. (9)). Algorithm 1 has three steps as follows.

Step-1 A system of linear equations $(A, \mathbf{k}, \mathbf{y})$ is divided into η sub systems. We show a toy example as follows.

$$\left(\begin{pmatrix}1&0&0&0&0&0&1\\0&1&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&1\\0&0&1&0&1&0&0\\0&0&0&1&0&1&0\end{pmatrix}, (k_1,k_2,k_3,k_4,k_5,k_6), (y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8)\right)$$

$$\Rightarrow \left(\begin{pmatrix}1&1\\0&1\end{pmatrix},(k_1,k_4),(y_1,y_7)\right), \left(\begin{pmatrix}1&1&0\\0&1&1\end{pmatrix},(k_2,k_5),(y_2,y_3,y_5)\right), \left(\begin{pmatrix}1&0\\1&1\end{pmatrix},(k_3,k_6),(y_4,y_6)\right)$$

Algorithm 1 Side-channel cube attack with key enumeration (SCCA-KE).

input A system of linear equations $(A, \mathbf{k}, \mathbf{y})$.
Divide $(A, \mathbf{k}, \mathbf{y})$ into η sub-systems $(A_j, \mathbf{k}_j, \mathbf{y}_j)$ $(j \in \{1, 2, ..., \eta\})$.
for $j = 1 \to \eta$ **do**
  for $o = 1 \to 2^{n_j}$ **do**
    $\mathbf{x}_j^o \leftarrow \mathbf{k}_j^o \cdot A_j$
    $\lambda_{\mathbf{k}_j^o} \leftarrow \sum_{i=1}^{L_j} \log_2(Pr[x_{j,i}^o \oplus y_{j,i}])$ ▷ $Pr[x_{j,i}^o \oplus y_{j,i}]$ is obtained by Eq.(5).
  end for
  Store all $(\mathbf{k}_j^o, \lambda_{\mathbf{k}_j^o})$ in $\mathcal{U}_{\mathbf{k}_j}$ in descending order by $\lambda_{\mathbf{k}_j^o}$.
end for
Store all $(\mathbf{rk}^o, -n')$ in $\mathcal{U}_{\mathbf{rk}}$ $(o \in \{1, 2, ..., 2^{n'}\})$.
$\mathbf{sk}^\psi \leftarrow \mathsf{KE}(\mathcal{U}_{\mathbf{k}_1}, \mathcal{U}_{\mathbf{k}_2}, ..., \mathcal{U}_{\mathbf{k}_\eta}, \mathcal{U}_{\mathbf{rk}}, \tau)$ ▷ KE is a key enumeration algorithm of [28].
return $\mathbf{sk}^\psi$ or "*failure*" ▷ "*failure*" indicates $\mathbf{sk}^\psi$ is not found in τ candidates.

Step-2 In each sub system, log likelihoods $\lambda_{\mathbf{k}_j^o}$ of all sub-key candidates are computed. Sub-key candidates and their log likelihoods $(\mathbf{k}_j^o, \lambda_{\mathbf{k}_j^o})$ are stored in $\mathcal{U}_{\mathbf{k}_j}$ in descending order by $\lambda_{\mathbf{k}_j^o}$. As mentioned in Sec. 2.1, there are n' remaining secret-key variables $\mathbf{rk}$; therefore, $\mathcal{U}_{\mathbf{rk}}$ for $\mathbf{rk}$ is prepared. Since there is no information on $\mathbf{rk}$, all sub-key candidates have the same log likelihood $\log_2(2^{-n'}) = -n'$.

Step-3 Using key enumeration, the correct key $\mathbf{sk}^\psi$ is searched. Note that the adversary determines τ as the number of candidates to enumerate and test. If the correct key can not be found in τ enumerated candidates, Algorithm 1 fails. Note that the optimal algorithm of [28] is used for KE. If another algorithm is used, $\mathcal{U}_{\mathbf{sk}}$ and $\mathcal{U}_{\mathbf{k}}$ are not optimal (they become sub optimal).

Contrary to the previous method shown in Sec. 3.1, optimal lists are generated in SCCA-KE for the following three reasons (each item corresponds to one of Sec. 3.1).

1. Log likelihoods of candidates are obtained by considering cube sizes. If a conditional probability distribution for leaked values given a key candidate is well approximated, a list $\mathcal{U}_{\mathbf{k}}$ sorted by log likelihoods is optimal and the secret key can be recovered based on ML-decoding.
2. Using key enumeration, secret-key candidates are enumerated in $\mathcal{U}_{\mathbf{k}}$ in descending order by $\lambda_{\mathbf{k}^o}$. Therefore, the log likelihood of the last candidate $\lambda_{\mathbf{k}^\tau}$ is more than any candidate not in $\mathcal{U}_{\mathbf{k}}$.
3. SCCA-KE does not take the overlapping DC strategy. Therefore, the key enumeration algorithm [28] can be executed to obtain $\mathcal{U}_{\mathbf{k}}$.

3.3 Complexity estimation

The total complexity for SCCA-KE is estimated as:

$$\begin{aligned} t &= t_{prep} + t_{ke} + \tau, \\ m &= m_{prep} + m_{ke}, \end{aligned} \tag{10}$$

where t_{prep} and t_{ke} (resp. m_{prep} and m_{ke}) are time (resp. memory) complexity for constructing the optimal sub lists (preparation for key enumeration) and execution of key enumeration. Note that t_{ke} does not include the time complexity for testing τ key candidates.

Taking DC strategy, complexity for constructing the sub lists become:

$$t_{prep} = \sum_{j=1}^{\eta} L_j \cdot 2^{n_j},$$
$$m_{prep} = \sum_{j=1}^{\eta} 2^{n_j}, \tag{11}$$

where n_j is the number of variables in a sub system. Note that this computation is required for the evaluation method shown in Sec. 4.3; therefore, it should be tractable in practice. If there is a sub system with large n_j, it is still hard to obtain a sub list. On this point, we have an observation as follows (see Appendix B for its intuitive explanation).

Observation. *Suppose that a system of linear equations with n secret-key variables is divided into η sub systems which do not contain any overlapping variables. The number of secret-key variables n_j is generally small in each sub system ($j \in \{1, 2, ..., \eta\}$).*

This is the observation obtained by experiments on PRESENT (see in Sec. 5.1) and other works [1][14]. If there is an exception, the adversary should consider techniques shown in Appendix C.

Using the optimal algorithm [28], t_{ke} and m_{ke} increase in the number of key candidates to enumerate and test τ. Especially, high memory complexity for key enumeration becomes a practical limitation [7]. In order to reduce complexity for key enumeration, efficient algorithms [7][8][20][21][24] can be used even if they are sub-optimal algorithms.

Since our goal is to propose an evaluation method for SCCA-KE, we only consider time complexity to test key candidates τ. Any key-enumeration algorithm can not achieve a time complexity less than τ; therefore, τ is the lower bound of time complexity. Also, we do not consider memory complexity, since there are sub-optimal algorithms with constant memory complexity. By considering security against SCCA-KE with all key-enumeration algorithms, we only consider the time complexity τ in the security evaluation against SCCA-KE.

4 Evaluation method for side-channel cube attack with key enumeration

In this section, we propose an evaluation method for SCCA-KE. The evaluator (of which the goal is to analyze security of his cryptographic device) considers the most effective attack to take countermeasures with a sufficient security margin. For any given leaked values which are measured using cubes, SCCA-KE is the most effective (error tolerant) key-recovery method, since it is based

on the optimal list of Definition 1. Note that the above statement is true if an approximation of conditional probability distribution $Pr[\mathbf{y}|\mathbf{k}]$ is accurate. We evaluate a security against SCCA-KE under an assumption that the approximation is accurate. The proposed evaluation method has three steps corresponding to subsections.

Step-1 (Sec. 4.1) Cubes are searched by a computer experiment. The evaluator should simulate any leakage which possibly occurs in his device.

Step-2 (Sec. 4.2) The expected time complexity is information-theoretically estimated. We consider *guessing entropy* which is equal to an expected time complexity. Since guessing entropy is computationally hard to obtain, we compute the lower bound [22], alternatively.

Step-3 (Sec. 4.3) Success rates of attacks are experimentally estimated by rank estimation. The evaluator uses rank estimation instead of key enumeration, since it can efficiently estimate time complexity of the attack. As an efficient and precise algorithm, we use a rank estimation algorithm proposed by Glowacz et al. [13].

As mentioned in Sec. 3.3, we assume that the time complexity of SCCA-KE is $t = \tau$ (the number of key candidates to test) and ignore the memory complexity. The evaluator sets a threshold of time complexity t_{adv} which is the number of candidates that the adversary can enumerate and test. The threshold is used in Step-2 and 3, and it should be determined in light of situations such as future computer technology and usages of the cryptographic device.

4.1 Cube search

A cube search algorithm has not been established. There are two existing strategies, random-walk search [10][17] and brute-force search [14]. We recommend the evaluator to execute brute-force search, since a sufficient number of trials to find all cubes has not been proven in the current random-walk search.

The brute-force search of cubes executes cube tests $\binom{N}{|\mathcal{I}|}$ times (see Sec. 2.1). A cube test executes a reduced-round cipher function $100 \times 2^{|\mathcal{I}|}$ times to obtain reliable results. We summarize the numbers of trials in Table 1 when $N = 64$.

We claim that it is not necessary to search cubes of large sizes if measurements are error-prone such as $\rho \geq 0.1$. Fig. 1 shows relations between crossover probability for RHS p_i and cube size $|\mathcal{I}|$ for $\rho \in (0 : 0.5]$ (see Eq. (5)). When the crossover probability is $\rho \geq 0.1$, the one for RHS becomes $p_i \approx 0.5$ if a cube of $|\mathcal{I}| > 6$ is used. In the evaluation of PRESENT shown in Sec. 5, we obtain a sufficient condition of ρ (closer to 0.5 than 0.1) to thwart SCCA-KE. Hence, we set a restriction, i.e., $|\mathcal{I}| \leq 6$, in the computer experiment of Sec. 5.1.

4.2 Information-theoretic evaluation

As a security metric, guessing entropy [22] was introduced to side-channel attack [16]. We define guessing entropy as follows.

Table 1. The number of trials for the brute-force cube search.

cube size $\|\mathcal{I}\|$	#trials
3	$2^{24.99}$
4	$2^{29.92}$
5	$2^{34.51}$
6	$2^{38.80}$
7	$2^{42.85}$
8	$2^{46.69}$

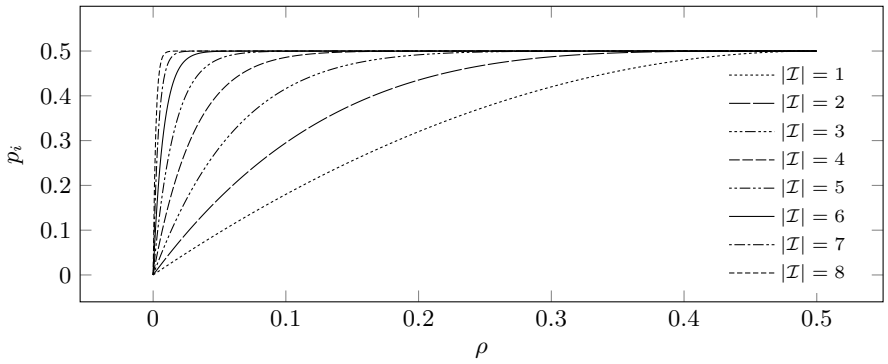


Fig. 1. Cross over probabilities for different cube sizes.

Definition 2 (guessing entropy [22][27]). *Suppose that the adversary obtains a list of candidates* $\mathcal{T}_{\boldsymbol{sk}} = \{\mathbf{sk}^1, \mathbf{sk}^2, ..., \mathbf{sk}^{2^N}\}$ *given a vector of leaked values* $\boldsymbol{y}$*. Guessing entropy for the adversary is:*

$$G = \mathop{\mathbb{E}}_{\mathbf{y}} \mathop{\mathbb{E}}_{\mathbf{sk}^\psi} [\mathsf{I}(\mathcal{T}_{\mathbf{sk}})], \tag{12}$$

where $\mathsf{I}(\mathcal{T}_{\mathbf{sk}})$ *is an index* o *such that* $\mathbf{sk}^o \in \mathcal{T}_{\mathbf{sk}}$ *is the correct key* $\mathbf{sk}^\psi$*.*

Guessing entropy is equal to an expected time complexity for key enumeration (t_{ke} is ignored), since $\mathsf{I}(\mathcal{T}_{\mathbf{sk}})$ equals to the number of times to test candidates until the correct key is found given $\mathbf{y}$. From the assumption mentioned in Sec. 3.3, an expected time complexity of SCCA-KE is $\mathbb{E}[t] = G$.

In SCCA-KE, guessing entropy G is obtained as [16]:

$$\begin{aligned} G &= \sum_{\mathbf{y}} Pr[\mathbf{y}] \sum_{o=1}^{2^N} o \cdot Pr[\mathbf{sk}^o|\mathbf{y}] \\ &= 2^{n'} \sum_{\mathbf{y}} Pr[\mathbf{y}] \sum_{o'=1}^{2^n-1} o' \cdot Pr[\mathbf{k}^{o'}|\mathbf{y}] + \frac{2^{n'}+1}{2}. \end{aligned} \tag{13}$$

It is computationally hard if n is large, since we should compute posterior probabilities of 2^n candidates. We use a lower bound of guessing entropy LG [3] ($G \geq LG$). In SCCA-KE, we can reduce time complexity for LG as follows.

Proposition 1 *In SCCA-KE, we can compute a lower bound of guessing entropy LG as follows.*

$$LG = \frac{1}{1+N} \sum_{\mathbf{y}} \left(\sum_{\mathbf{sk}^{\psi}} Pr[\mathbf{sk}^{\psi}, \mathbf{y}]^{\frac{1}{2}} \right)^2$$
$$= \frac{2^{n'}}{1+N} \prod_{j=1}^{\eta} \left(2^{-n_j} \sum_{o=1}^{2^{n_j}} \sum_{\substack{o'=1 \\ o' \neq o}}^{2^{n_j}} \left(\prod_{\substack{i=1 \\ x^o_{j,i} \neq x^{o'}_{j,i}}}^{L_j} 2 \cdot p_{j,i}^{\frac{1}{2}} \cdot (1-p_{j,i})^{\frac{1}{2}} \right) + 1 \right), \quad (14)$$

where $\mathbf{x}^o_j = \mathbf{k}^o_j \cdot A_j = (x^o_{j,1}, x^o_{j,2}, ..., x^o_{j,L_j})$ *be a vector of correct RHS values for a sub-key candidate* $\mathbf{k}^o_j$ *and* $p_{j,i} = Pr[x^o_{j,i} \oplus y_{j,i} = 1]$.

See Appendix D for the proof. If sub systems of linear equations $(A_j, \mathbf{k}_j, \mathbf{x}_j)$ are given, Eq. (14) is computed by $\sum_{j=1}^{\eta} L_j \cdot 2^{n_j} \cdot (2^{n_j} - 1)$ times of multiplications. If the observation shown in Sec. 3.3 is correct, the computation is tractable.

The evaluator obtains a lower bound of expected time complexity of SCCA-KE from Eq. (14). If this is sufficiently larger than t_{adv}, he can assure security of his device. Therefore, he should consider countermeasures which can increase LG. Note that guessing entropy can not prove that the success rate is negligible. Hence, it is important to execute an experimental evaluation shown in Sec. 4.3.

4.3 Experimental evaluation by rank estimation

Rank estimation is an evaluation tool for side-channel attacks enhanced by key enumeration [29]. The algorithm is to evaluate the rank of a correct key when the evaluator knows the correct key. We show the algorithm for the experimental evaluation in Algorithm 2. The algorithm outputs a success rate SR for randomly chosen correct keys and leakages (these two values are not independent). Algorithm 2 has three steps as follows.

Step-1 In the same manner as Algorithm 1, a system of linear equations is divided. Also, a vector of crossover probabilities $\mathbf{p}$ is divided into η vectors $\mathbf{p}_j = (p_{j,1}, p_{j,2}, ..., p_{j,L_j})$. The evaluator repeats **Step-2** and **3** for s_{all} times by changing a value of the correct key $\mathbf{sk}^{\psi} = \mathbf{k}^{\psi} || \mathbf{rk}^{\psi}$.

Step-2 The following procedures are repeated for all $j \in \{1, 2, ..., \eta\}$. A correct sub key $\mathbf{k}^{\psi}_j$ is chosen at random, and a vector of correct RHS values $\mathbf{x}^{\psi}_j$ is obtained by $\mathbf{k}^{\psi}_j$. A vector of measured RHS values $\mathbf{y}$ are chosen at random according to $\mathbf{p}_j$. A log likelihood of the correct sub key $\lambda_{\mathbf{k}^{\psi}_j}$ is derived. For all candidates of $\mathbf{k}_j$, log likelihoods are computed. All results are stored in a list $\mathcal{U}_{\mathbf{k}_j}$ and $(\mathbf{k}^o_j, \lambda_{\mathbf{k}^o_j})$ is sorted in descending order by $\lambda_{\mathbf{k}^o_j}$.

Algorithm 2 Experimental evaluation for SCCA-KE by rank estimation.

input Threshold t_{adv}, a matrix A with variables $\mathbf{k}$ and probabilities $\mathbf{p}$.
Divide $(A, \mathbf{k}, \mathbf{y})$ into η sub-systems $(A_j, \mathbf{k}_j, \mathbf{y}_j)$ $(j \in \{1, 2, ..., \eta\})$.
Divide $\mathbf{p}$ into η vectors $\mathbf{p}_j$ corresponding to $(A_j, \mathbf{k}_j, \mathbf{y}_j)$ $(j \in \{1, 2, ..., \eta\})$.
$s \leftarrow 0$ ▷ s: counter variable for *success*.
for $trial = 1 \rightarrow s_{all}$ **do**
  for $j = 1 \rightarrow \eta$ **do**
    Choose a value of the correct sub key $\mathbf{k}_j^{\psi}$ at random.
    $\mathbf{x}_j^{\psi} \leftarrow \mathbf{k}_j^{\psi} \cdot A_j$
    Obtain $\mathbf{y}_j$ from $\mathbf{x}_j^{\psi}$ at random according to $\mathbf{p}_j$.
    $\lambda_{\mathbf{k}_j^{\psi}} \leftarrow \sum_{i=1}^{L_j} \log_2(Pr[x_{j,i}^{\psi} \oplus y_{j,i}])$ ▷ $\lambda_{\mathbf{k}_j^{\psi}}$: log likelihood of the correct sub key.
    for $o = 1 \rightarrow 2^{|n_j|}$ **do**
      $\mathbf{x}_j^{o} \leftarrow \mathbf{k}_j^{o} \cdot A_j$
      $\lambda_{\mathbf{k}_j^{o}} \leftarrow \sum_{i=1}^{L_j} \log_2(Pr[x_{j,i}^{o} \oplus y_{j,i}])$ ▷ $Pr[x_{j,i}^{o} \oplus y_{j,i} = 1] = p_{j,i}$.
    end for
    Store all $(\mathbf{k}_j^{o}, \lambda_{\mathbf{k}_j^{o}})$ in $\mathcal{U}_{\mathbf{k}_j}$ in descending order by $\lambda_{\mathbf{k}_j^{o}}$.
  end for
  $\mathbf{x} \leftarrow \mathbf{k}^{\psi} \cdot A$ ▷ RE is a rank estimation algorithm of [13].
  $\tau \leftarrow \mathsf{RE}(\mathcal{U}_{\mathbf{k}_1}, \mathcal{U}_{\mathbf{k}_2}, ..., \mathcal{U}_{\mathbf{k}_\eta}, \lambda_{\mathbf{k}_1^{\psi}}, \lambda_{\mathbf{k}_2^{\psi}}, ..., \lambda_{\mathbf{k}_\eta^{\psi}}, n')$
  if $t_{adv} \geq \tau$ **then**
    $s \leftarrow s + 1$
  end if
end for
return $SR = s/s_{all}$

Step-3 From $\mathcal{U}_{\mathbf{k}_j}$ and $\lambda_{\mathbf{k}_j^{\psi}}$ $(j \in \{1, 2, ..., \eta\})$, a rank τ of the correct key is estimated by a rank-estimation algorithm RE. We use a rank estimation algorithm proposed by Glowacz et al. [13]. See Appendix E for the algorithm. If a total complexity τ is less than t_{adv}, the attack is regarded as successful and s increases by 1. Return to **Step-1**.

Step-4 After s_{all} times of trials, a success rate $SR = s/s_{all}$ is outputted.

The time complexity of Algorithm 2 is estimated as $s_{all} \cdot (t_{prep} + t_{re})$, where t_{prep} is obtained by Eq. (11) and t_{re} is a time complexity for rank estimation (see Eq. (17)).

5 Application to PRESENT

We apply the evaluation method shown in Sec. 4 to PRESENT [6]. We suppose that PRESENT is implemented on a device encrypting a small amount of data (e.g., smart card). As a realistic assumption for the device, we set the adversary's condition as $t_{adv} = 2^{60}$ and $q_{adv} = 2^{15}$ (the number of leaked values at most). Note that 2^{15} leaked values require encryption processes of at least 256 KB of data. Using the assumption, we can assure sufficient security of the device. In Sec. 5.1, we show the results of cube search. In Sec. 5.2, we evaluate the security of PRESENT against SCCA-KE.

5.1 Cubes of PRESENT

As mentioned in Sec. 4.1, we execute brute-force cube search by restricting the cube size, i.e., $|\mathcal{I}| \leq 6$. We experiment under the following *leakage models*, where w_i ($i \in \{1, 2, ..., 64\}$) denotes an output bit of internal rounds of PRESENT.

1. Single-bit leakage [32]: An output bit of an internal round is leaked as w_i ($i \in \{1, 2, ..., 64\}$).
2. LSB leakage of HW (binary representation) of 8-bit state [17]: A LSB of HW of 8-bit internal states (output of two S-boxes) of PRESENT is leaked as $\sum_{j=1}^{8} w_{8\cdot(i-1)+j}$ ($i \in \{1, 2, ..., 8\}$). PRESENT is assumed to be implemented on an 8-bit processor.
3. LSB leakage of HW of 4-bit state: A LSB of HW of 4-bit internal states (output of one S-box) of PRESENT is leaked as $\sum_{j=1}^{4} w_{4\cdot(i-1)+j}$ ($i \in \{1, 2, ..., 16\}$). PRESENT is assumed to be implemented on a 4-bit processor.

Since LSB can be represented by the lowest-degree polynomial among all HW bits, this leakage model is the most advantageous for the adversary. For all (r, i) (i-th bit/LSB of r-th round) of all leakage models, we execute brute-force search.

As a result, we have results shown in Table 2, 3 and 4. We show cubes with $n \geq 20$, since this is the necessary condition for the successful attack ($t_{adv} = 2^{60}$ and secret-key length is 80). Even if the adversary can recover n ($n < 20$) secret-key variables with negligible complexity, he should recover $80 - n$ secret-key variables without any advantage. Note that "$\max n_j$" and "$\max L_j$" denotes the maximum values of n_j and L_j among all sub systems ($j \in \{1, 2, ..., \eta\}$). Both values determine whether execution of information-theoretic and experimental evaluations are tractable. Since $\max n_j = 16$ and $\max L_j = 2220$ at most in the tables, the above execution time is relatively small and this fact supports our observation shown in Sec. 3.3.

5.2 Security evaluation of PRESENT

We use six conditions which are advantageous for the adversary, and Table 5 shows them. In order to satisfy $q \leq q_{adv}$ ($q_{adv} = 2^{15}$), some cubes should be reduced (see underlined number of Table 5). Also, some cubes are used multiple times if the number of leaked values are smaller than q_{adv}. For simplicity, we use all cubes for the same number of times, and d denotes the number.

Choosing two conditions from each leakage model, we show the results for information-theoretic and experimental evaluations in Fig. 2 and 3, respectively. In the latter, we execute the experiment for 1,000 times ($s_{all} = 1,000$ in Algorithm 2) changing the correct key $\mathbf{sk}^{\psi}$ and RHS values $\mathbf{y}$.

From the results of evaluation, we have the following observations.

1. In PRESENT, SCCA is error tolerant when leakages are from early rounds such as the second and third rounds. This is a general statement for block cipher. Since an algebraic degree increases round by round, linear superpolies can not be obtained by cubes of small sizes. Obviously, cubes of large sizes such as $|\mathcal{I}| \geq 5$ are not error tolerant (see Table 1).

Table 2. The results of cube search for single bit leakage.

r	i	#cubes of each cube size						n	L	$\max n_j$	$\max L_j$
		1	2	3	4	5	6				
3	1	8	48	120	160	0	48	32	384	1	30
3	5	0	0	144	144	0	3456	32	3744	1	288
3	26	0	0	0	2880	0	0	32	2880	1	144
3	33	0	0	0	1920	0	0	32	1920	1	96
3	49	0	0	0	2880	0	0	32	2880	1	144
4	17	0	0	0	231	731	4813	51	5775	16	2209
4	49	0	0	0	240	731	4815	51	5786	16	2220

Table 3. The results of cube search for LSB leakage of 8-bit state.

r	i	#cubes of each cube size						n	L	$\max n_j$	$\max L_j$
		1	2	3	4	5	6				
2	1	0	48	0	0	144	0	64	192	3	9
2	2	8	24	0	48	96	0	56	176	8	36
3	1	0	0	0	0	0	108	24	108	3	18
3	2	0	0	0	68	80	36	30	184	3	26

Table 4. The results of cube search for LSB leakage of 4-bit state.

r	i	#cubes of each cube size						n	L	$\max n_j$	$\max L_j$
		1	2	3	4	5	6				
2	1	16	32	16	0	0	0	32	64	1	3
2	2	0	48	0	0	144	0	64	192	3	9
2	3	0	40	0	0	80	0	64	120	2	4
2	4	0	48	0	0	144	0	64	192	3	9
2	5	0	0	288	288	72	0	32	648	1	27
3	1	0	20	0	102	120	72	60	314	4	52
3	2	0	0	0	0	0	108	24	108	3	18
3	3	0	0	0	0	0	72	24	72	3	12
3	4	0	0	0	0	0	108	24	108	3	18
3	5	0	0	0	0	0	432	24	432	3	72
3	9	0	0	0	0	0	288	24	288	3	48
3	13	0	0	0	0	0	432	24	432	3	72

2. The second and third rounds give an advantage to the adversary. Therefore, SCCA can be thwarted by sufficient protections for the above rounds. Also, $\rho \geq 0.4$ is sufficient to prevent SCCA-KE from Fig. 2 and 3.

5.3 Comparison with the previous method

In Fig. 4, we show a comparison to the previous method by using a leakage model used in [17] (LSB leakage of HW of 8-bit state and $(r, i) = (2, 1)$, see Table 3). The experiment is executed under the same condition as Sec. 5.2 ($t_{adv} = 2^{60}$

Table 5. Cubes used in the evaluation.

Leakage model	r	i	#cubes of each cube size						n	L	$\max n_j$	$\max L_j$	q	d
			1	2	3	4	5	6						
Single	3	1	8	48	120	160	720	48	32	384	1	30	$2^{14.86}$	1
bit	4	49	0	0	0	240	731	0	32	971	16	780	$2^{14.73}$	1
LSB of HW	2	2	64	192	0	384	768	0	56	1408	8	288	$2^{14.94}$	8
of 8-bit state	3	2	0	0	0	340	400	180	30	920	3	130	$2^{14.86}$	5
LSB of HW	2	1	1808	3616	1808	0	0	0	32	7232	1	339	$2^{14.81}$	113
of 4-bit state	3	1	0	60	0	306	360	216	60	942	4	156	$2^{14.89}$	3

Fig. 2. Lower bounds of guessing entropy of single-bit leakage (left), LSB leakage of 8-bit state (middle) and LSB leakage of 4-bit state (right).

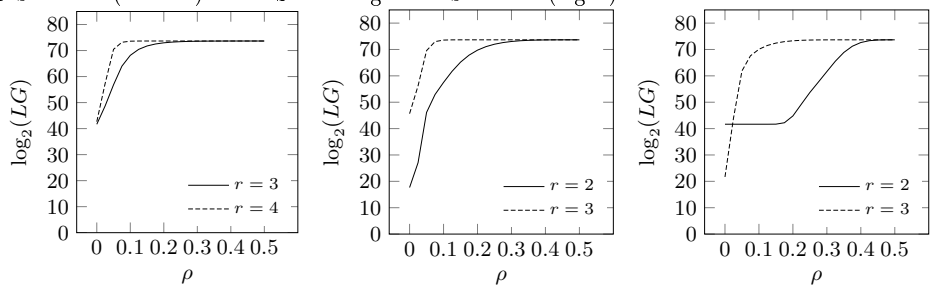
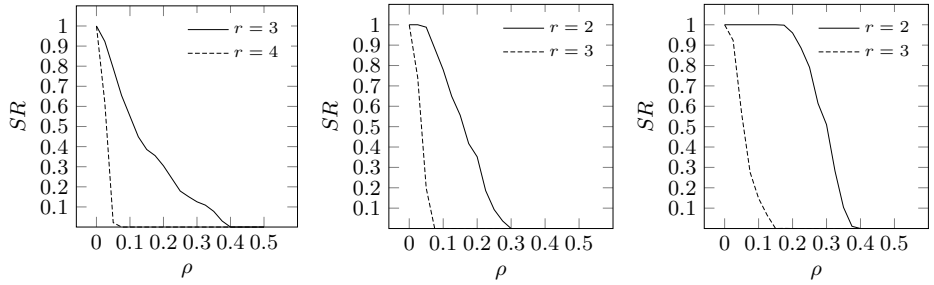


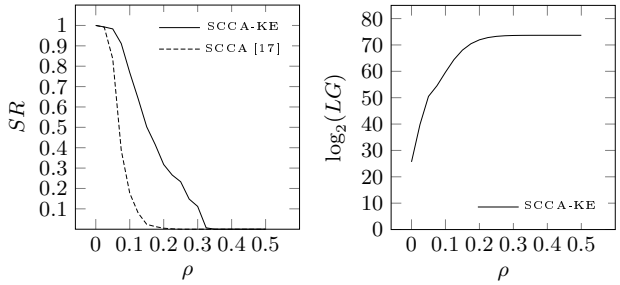
Fig. 3. Success rates of single-bit leakage (left), LSB leakage of 8-bit state (middle) and LSB leakage of 4-bit state (right).



and $q_{adv} = 2^{15}$), and all cubes are used $d = 6$ times. Obviously, we can conclude that SCCA-KE is more error tolerant than the previous method. Since guessing entropy can not be defined in the previous method, we only show LG for SCCA-KE as a reference.

We execute experiments for the previous method by computing hamming distances of the correct key and the last candidate of the list of secret-key candidates $\mathcal{T}_{\mathbf{sk}}$ ($|\mathcal{T}_{\mathbf{sk}}| = 2^{60}$). If the former value is less than the latter one, we regard

Fig. 4. Success rates of SCCA-KE and the previous method [17] under LSB leakage of HW of 8-bit state and $(r, i) = (2, 1)$ (left), and lower bounds of guessing entropy of SCCA-KE in the same condition (right).



the attack as success. A system of linear equations for 64 secret-key variables is divided into four sub systems in the same manner as [17].

6 Conclusions and open problems

We propose SCCA-KE which improves error tolerance of the previous method [17]. Then, we propose an evaluation method with information-theoretic and experimental evaluations. Using the evaluation method, the evaluator can consider a countermeasure with a sufficient security margin. In this paper, we evaluate PRESENT in various situations under BSC model fixing the adversary's condition. It is also possible to evaluate the security for various adversary.

For further study, we list open problems.

1. Cube search strategy has not been established. Since it is computationally hard to execute brute-force cube search in larger cube sizes, more efficient algorithms should be designed.
2. After the cube search, we should select cubes in order to maximize the success rate considering the restriction of the number of leaked values and computing power. In other word, selection of cubes should be optimized.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 17K0645.

References

1. Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: On the security of NOEKEON against side channel cube attacks. In: International Conference on Information Security Practice and Experience. pp. 45–55. Springer (2010)
2. Akkar, M.L., Bevan, R., Dischamp, P., Moyart, D.: Power analysis, what is now possible... In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 489–502. Springer (2000)

3. Arikan, E.: An inequality on guessing and its application to sequential decoding. IEEE Transactions on Information Theory 42(1), 99–105 (1996)
4. Bellare, M., Coppersmith, D., Hastad, J., Kiwi, M., Sudan, M.: Linearity testing in characteristic two. IEEE Transactions on Information Theory 42(6), 1781–1795 (1996)
5. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory 24(3), 384–386 (1978)
6. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer Berlin Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-74735-2_31
7. Bogdanov, A., Kizhvatov, I., Manzoor, K., Tischhauser, E., Witteman, M.: Fast and memory-efficient key recovery in side-channel attacks. In: International Conference on Selected Areas in Cryptography. pp. 310–327. Springer (2015)
8. David, L., Wool, A.: A bounded-space near-optimal key enumeration algorithm for multi-dimensional side-channel attacks. IACR Cryptology ePrint Archive 2015, 1236 (2015)
9. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 278–299. Springer (2009)
10. Dinur, I., Shamir, A.: Side channel cube attacks on block ciphers. IACR Cryptology ePrint Archive 2009, 127 (2009)
11. Faisal, S., Reza, M., Susilo, W., Seberry, J.: Extended cubes: Enhancing the cube attack by extracting low-degree non-linear equations (2011)
12. Gérard, B., Standaert, F.X.: Unified and optimized linear collision attacks and their application in a non-profiled setting. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 175–192. Springer (2012)
13. Glowacz, C., Grosso, V., Poussier, R., Schueth, J., Standaert, F.X.: Simpler and more efficient rank estimation for side-channel security assessment. In: International Workshop on Fast Software Encryption. pp. 117–129. Springer (2015)
14. Islam, S., Afzal, M., Rashdi, A.: On the security of LBlock against the cube attack and side channel cube attack. In: International Conference on Availability, Reliability, and Security. pp. 105–121. Springer (2013)
15. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Annual International Cryptology Conference. pp. 388–397. Springer (1999)
16. Köpf, B., Basin, D.: An information-theoretic model for adaptive side-channel attacks. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 286–296. ACM (2007)
17. Li, Z., Zhang, B., Fan, J., Verbauwhede, I.: A new model for error-tolerant side-channel cube attacks. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 453–470. Springer (2013)
18. Lin, S., Costello, D.J.: Error control coding. Pearson Education India (2004)
19. MacKay, D.J.: Information theory, inference and learning algorithms. Cambridge university press (2003)
20. Manzoor, K., et al.: Efficient practical key recovery for side-channel attacks. Master's thesis, Aalto University, June 2014. http://cse. aalto. fi/en/personnel/antti-yla-jaaski/msc-thesis/2014-msc-kamran-manzoor. pdf (2014)

21. Martin, D.P., Óconnell, J.F., Oswald, E., Stam, M.: Counting keys in parallel after a side channel attack. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 313–337. Springer (2014)
22. Massey, J.L.: Guessing and entropy. In: Information Theory, 1994. Proceedings., 1994 IEEE International Symposium on. p. 204. IEEE (1994)
23. Matsui, M.: Linear cryptanalysis method for des cipher. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 386–397. Springer (1993)
24. Poussier, R., Standaert, F.X., Grosso, V.: Simple key enumeration (and rank estimation) using histograms: an integrated approach. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 61–81. Springer (2016)
25. Renauld, M., Standaert, F.X.: Algebraic side-channel attacks. In: International Conference on Information Security and Cryptology. pp. 393–410. Springer (2009)
26. Siegenthaler, T.: Decrypting a class of stream ciphers using ciphertext only. IEEE Transactions on computers 34(1), 81–85 (1985)
27. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 443–461. Springer (2009)
28. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: International Conference on Selected Areas in Cryptography. pp. 390–406. Springer (2012)
29. Veyrat-Charvillon, N., Gérard, B., Standaert, F.X.: Security evaluations beyond computing power. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 126–141. Springer (2013)
30. Veyrat-Charvillon, N., Gérard, B., Standaert, F.X.: Soft analytical side-channel attacks. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 282–296. Springer (2014)
31. Xinjie, Z., Shize, G., Zhang, F., Tao, W., Zhijie, S., Hao, L.: Enhanced side-channel cube attacks on PRESENT. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 96(1), 332–339 (2013)
32. Yang, L., Wang, M., Qiao, S.: Side channel cube attack on PRESENT. In: International Conference on Cryptology and Network Security. pp. 379–391. Springer (2009)
33. Zhao, X.j., Wang, T., Guo, S.: Improved side channel cube attacks on PRESENT. IACR Cryptology ePrint Archive 2011, 165 (2011)

A Conversion from HW model with Gaussian noise to BSC model

We convert a HW leakage model with Gaussian noise [2] to BSC model. The model is defined by $h = \mathsf{HW}(\mathbf{w}) + \mathsf{N}_\xi(0, \delta^2)$, where $\mathbf{w} \in \mathbb{F}_2^m$ is a value of register and $\mathsf{N}_\xi(0, \delta^2)$ is a Gaussian distribution for additive noise ξ (mean 0 and standard deviation δ).

We assume that LSB of a binary value of HW of $\lfloor h \rfloor$ leaks. The measured value is different from the correct one when $\lfloor \mathsf{N}_\xi(0, \delta^2) \rfloor$ is odd, since th value depends on whether $\lfloor h \rfloor$ is even or odd. Therefore, a crossover probability becomes

symmetric and it is obtained as:

$$\begin{aligned}\rho &= 2\sum_{i=0}^{\infty}\int_{2\cdot i+1}^{2\cdot i+2} \mathsf{N}_{\xi}(0,\delta^2)d\xi \\ &= \sum_{i=0}^{\infty}\left(\mathrm{erf}\left(\frac{2\cdot i+2}{\delta\cdot\sqrt{2}}\right)-\mathrm{erf}\left(\frac{2\cdot i+1}{\delta\cdot\sqrt{2}}\right)\right), \end{aligned} \tag{15}$$

where "erf" is the error function.

B Intuitive explanation for the observation

If the number of variables in a linear equation is large, the linear equation shares variables with many other equations and n^j becomes large. Therefore, we show the reason that the number of secret-key variables in each linear equation is small. In early rounds such as first and second rounds, the number of secret-key variables are small in all superpolies since diffusion of key variables is not enough. In other rounds, the number of secret-key variables in each superpoly increases. However, superpolies with many secret-key variables tend to be non-linear, since the output value has passed through many AND operations (such as in Sboxes). Therefore, the number of secret-key variables in each linear equation is small in any round and n^j tends to be small.

C Approaches for sub systems with large secret-key variables

If there is an exception for the observation shown in Sec. 3.3, complexity to make the sub lists (see Eq. (10)) is intractable. To solve the problem, we can take the following approaches.

1. By disregarding some linear equations, it is possible to divide a sub system into multiple sub systems. We show a toy example as follows.

$$\begin{aligned}&\left(\begin{pmatrix} 0\,0\,1 & \\ 0\,0\,0 & A_1 \\ & \\ A_2 & \begin{matrix}0\,0\,0\\1\,0\,0\end{matrix}\end{pmatrix}, (k_1,k_2,k_3,k_4), (y_1,y_2,y_3,y_4,y_5)\right) \\ &\rightarrow \left(\begin{pmatrix} 0\,0 & \\ 0\,0 & A_1 \\ & \\ A_2 & \begin{matrix}0\,0\\0\,0\end{matrix}\end{pmatrix}, (k_1,k_2,k_3,k_4), (y_1,y_2,y_4,y_5)\right) \\ &\Rightarrow (A_1,(k_1,k_2),(y_1,y_2)), (A_2,(k_3,k_4),(y_3,y_4))\end{aligned}$$

2. If a matrix A_j is sparse, approximation algorithms for decoding linear codes such as sum-product algorithm [18] can be a solution. This approach has already been taken in other side-channel attacks [12][30]. Application of such algorithms to SCCA is an open problem.

3. The overlapping DC strategy can be a solution if an efficient key-enumeration algorithm (and a rank-estimation one for the evaluation) to handle overlapping variables will be designed.

D Proof for Proposition 1

A lower bound of guessing entropy can be computed in each sub system.

$$
\begin{aligned}
LG &= \frac{1}{1+N}\sum_{\mathbf{y}}\left(\sum_{\mathbf{sk}^{\psi}} Pr[\mathbf{sk}^{\psi},\mathbf{y}]^{\frac{1}{2}}\right)^2 \\
&= \frac{1}{1+N}\sum_{\mathbf{y}}\left(\sum_{\mathbf{k}^{\psi}}\prod_{j=1}^{\eta} Pr[\mathbf{y}_j|\mathbf{k}_j^{\psi}]^{\frac{1}{2}}\cdot Pr[\mathbf{k}_j^{\psi}]^{\frac{1}{2}}\right)^2 \cdot \left(\sum_{\mathbf{rk}^{\psi}} Pr[\mathbf{rk}^{\psi}]^{\frac{1}{2}}\right)^2 \\
&= \frac{2^{n'}}{1+N}\sum_{\mathbf{y}}\left(\prod_{j=1}^{\eta} 2^{\frac{-n_j}{2}}\sum_{\mathbf{k}_j^{\psi}} Pr[\mathbf{y}_j|\mathbf{k}_j^{\psi}]^{\frac{1}{2}}\right)^2 \\
&= \frac{2^{n'}}{1+N}\prod_{j=1}^{\eta} 2^{-n_j}\sum_{\mathbf{y}_j}\left(\sum_{\mathbf{k}_j^{\psi}} Pr[\mathbf{y}_j|\mathbf{k}_j^{\psi}]^{\frac{1}{2}}\right)^2 \\
&= \frac{2^{n'}}{1+N}\prod_{j=1}^{\eta} LG_j
\end{aligned}
$$

Note that we assume that uniform prior distribution holds ($Pr[\mathbf{k}_j] = 2^{-n_j}$, $Pr[\mathbf{rk}] = 2^{-n'}$). Then, we can simplify LG_j by expanding the square part as follows.

$$
\begin{aligned}
LG_j &= 2^{-n_j}\sum_{\mathbf{y}_j}\sum_{o=1}^{2^{n_j}}\sum_{o'=1}^{2^{n_j}} Pr[\mathbf{y}_j|\mathbf{k}_j^{o}]^{\frac{1}{2}}\cdot Pr[\mathbf{y}_j|\mathbf{k}_j^{o'}]^{\frac{1}{2}} \\
&= 2^{-n_j}\sum_{o=1}^{2^{n_j}}\sum_{o'=1}^{2^{n_j}}\sum_{\mathbf{y}_j}\prod_{i=1}^{L_j} Pr[y_{j,i}\oplus x_{j,i}^{o}]^{\frac{1}{2}}\cdot Pr[y_{j,i}\oplus x_{j,i}^{o'}]^{\frac{1}{2}} \\
&= 2^{-n_j}\sum_{o=1}^{2^{n_j}}\sum_{o'=1}^{2^{n_j}}\prod_{i=1}^{L_j}\sum_{y_{j,i}} Pr[y_{j,i}\oplus x_{j,i}^{o}]^{\frac{1}{2}}\cdot Pr[y_{j,i}\oplus x_{j,i}^{o'}]^{\frac{1}{2}} \\
&= 2^{-n_j}\sum_{o=1}^{2^{n_j}}\sum_{o'=1}^{2^{n_j}}\prod_{i=1}^{L_j}\left(Pr[0\oplus x_{j,i}^{o}]^{\frac{1}{2}}\cdot Pr[0\oplus x_{j,i}^{o'}]^{\frac{1}{2}} + Pr[1\oplus x_{j,i}^{o}]^{\frac{1}{2}}\cdot Pr[1\oplus x_{j,i}^{o'}]^{\frac{1}{2}}\right) \\
&= 2^{-n_j}\sum_{o=1}^{2^{n_j}}\sum_{\substack{o'=1\\ o'\neq o}}^{2^{n_j}}\left(\prod_{\substack{i=1\\ x_{j,i}^{o}\neq x_{j,i}^{o'}}}^{L_j} 2\cdot p_{j,i}^{\frac{1}{2}}\cdot(1-p_{j,i})^{\frac{1}{2}}\right) + 1
\end{aligned}
\tag{16}
$$

Algorithm 3 Rank estimation algorithm of [13].

input Sub-key lists $\mathcal{U}_{\mathbf{k}_1}$,$\mathcal{U}_{\mathbf{k}_2}$,...,$\mathcal{U}_{\mathbf{k}_\eta}$, log likelihoods of the correct sub keys $\lambda_{\mathbf{k}_1^\psi}, \lambda_{\mathbf{k}_2^\psi}, ..., \lambda_{\mathbf{k}_\eta^\psi}$ and the number of remaing variables n'.

$b^\psi \leftarrow 0$
for $j = 1 \rightarrow \eta$ **do**
  $b^\psi \leftarrow b^\psi + \lfloor \lambda_{\mathbf{k}_j^\psi} / l_{bin} \rfloor$
  for $o = 1 \rightarrow 2^{n_j}$ **do**
    Calculate a bin index $b \leftarrow \lfloor \lambda_{\mathbf{k}_j^o} / l_{bin} \rfloor$.
    $\mathsf{H}_j(b) \leftarrow \mathsf{H}_j(b) + 1$
  end for
end for
$\mathsf{H}_{1:1} \leftarrow \mathsf{H}_1$.
for $j = 2 \rightarrow \eta$ **do**
  for $b = (j-1) \rightarrow j \cdot n_{bin} - (j-1)$ **do**
    for $b' = 1 \rightarrow n_{bin}$ **do**
      $\mathsf{H}_{1:j}(b+b') \leftarrow \mathsf{H}_{1:j}(b+b') + \mathsf{H}_{1:j-1}(b) \cdot \mathsf{H}_j(b')$
    end for
  end for
end for
return $\tau \leftarrow \sum_{b=b^\psi}^{\eta \cdot n_{bin} - (\eta-1)} \mathsf{H}_{1:\eta}(b) + n'$

In the last arrangement, we only consider products of probabilities such that $x_{j,i}^o \neq x_{j,i}^{o'}$, since a product of probabilities is always 1 if $x_{j,i}^o = x_{j,i}^{o'}$. Also, there are 2^{n_j} pairs such that $o = o'$; therefore, $2^{n_j} \cdot 2^{-n_j} = 1$ is added. □

E Rank estimation algorithm of [13]

Algorithm 3 shows the rank estimation algorithm. We obtain η lists $\mathcal{U}_{\mathbf{k}_j} = \{(\mathbf{k}_j^o, \lambda_{\mathbf{k}_j^o}) | \lambda_{\mathbf{k}_j^o} = \log_2(Pr[\mathbf{y}_j | \mathbf{k}_j^o])\}$. From the lists, histograms H_j (the size and width of bins are n_{bin} and l_{bin}) are constructed ($j \in \{1, 2, ..., \eta\}$), and $\mathsf{H}_j(b)$ denotes the number of candidates in the b-th bin. Convoluting histograms from H_1 to H_j, we can construct $\mathsf{H}_{1:j}$ (the number of bins is $j \times n_{bin} - (j-1)$). The rank-estimation algorithm outputs an estimated rank by using the last histogram $\mathsf{H}_{1:\eta}$ and the index of bin in which the correct key may be included. Let $b^\psi (\geq 0)$ be such index, and it is obtained by $b^\psi = \sum_{j=1}^{\eta} \lfloor \lambda_{\mathbf{k}_j^\psi} / l_{bin} \rfloor$, where $\mathbf{k}_j^\psi$ is the correct sub key. The algorithm executes t_{re} times additions in the convolution (t_{re} is a time complexity of Algorithm 3):

$$t_{re} = \sum_{j=2}^{m} \sum_{b=j-1}^{j \cdot n_{bin} - (j-1)} n_{bin}. \tag{17}$$

There is an estimation error caused by the convolution of histograms; however, the rank can be lower bounded by $\sum_{b=b^\psi + \lceil \eta/2 \rceil}^{\eta \times n_{bin} - (\eta-1)} \mathsf{H}_{1:\eta}(b)$ and upper bounded by $\sum_{b=b^\psi - \lceil \eta/2 \rceil}^{\eta \times n_{bin} - (\eta-1)} \mathsf{H}_{1:\eta}(b)$.