Formal Analysis of Non-profiled Deep-learning Based Side-channel Attacks

Akria Ito[1][0000−0002−4602−7570], Rei Ueno[2][0000−0002−9754−6792], Rikuma Tanaka[2], and Naofumi Homma[2]

[1] NTT Social Informatics Laboratories, Nippon Telegraph and Telephone Corporation,
3–9–11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan
akira.itoh@ntt.com

[2] Tohoku University, 2–1–1 Katahira, Aoba-ku, Sendai-shi, Miyagi, 980-8577, Japan
rei.ueno.a8@tohoku.ac.jp, rikuma.tanaka.q1@dc.tohoku.ac.jp,
naofumi.homma.c8@tohoku.ac.jp

Abstract. This paper formally analyzes two major non-profiled deep-learning-based side-channel attacks (DL-SCAs): differential deep-learning analysis (DDLA) by Timon and collision DL-SCA by Staib and Moradi. These DL-SCAs leverage supervised learning in non-profiled scenarios. Although some intuitive descriptions of these DL-SCAs exist, their formal analyses have been rarely conducted yet, which makes it unclear why and when the attacks succeed and how the attack can be improved. In this paper, we provide the first information-theoretical analysis of DDLA. We reveal its relevance to the mutual information analysis (MIA), and then present three theorems stating some limitations and impossibility results of DDLA. Subsequently, we provide the first probability-theoretical analysis on collision DL-SCA. After presenting its formalization with a proposal of our distinguisher for collision DL-SCA, we prove its optimality. Namely, we prove that the collision DL-SCA using our distinguisher theoretically maximizes the success rate if the neural network (NN) training is completely successful (namely, the NN completely imitates the true conditional probability distribution). Accordingly, we propose an improvement of the collision DL-SCA based on a dedicated NN architecture and a full-key recovery methodology using multiple neural distinguishers. Finally, we experimentally evaluate non-profiled (DL-)SCAs using a newly created dataset using publicly available first-order masked AES implementation. The existing public dataset of side-channel traces is insufficient to evaluate collision DL-SCAs due to a lack of substantive side-channel traces for different key values. Our dataset enables a comprehensive evaluation of collision (DL-)SCAs, which clarifies the current situation of non-profiled (DL-)SCAs.

Keywords: Side-channel attack (SCA)· Collision SCA · Deep learning · Non-profiled attack · Optimal distinguisher · Symmetric cipher.

1 Introduction

1.1 Background

Deep-learning-based side-channel attack. Recently, deep-learning-based side-channel attacks (DL-SCAs) have been extensively studied and developed [9,23,36,37,50, 57,60]. DL-SCA is known to be a powerful SCA on cryptographic modules. Actually, it has been shown that DL-SCA can achieve high performance in recovering the secret key [35] and distinguishing the input [57,60] even on implementations with SCA countermeasures, such as masking and random delay. In addition, some previous works showed that an optimal SCA in terms of success rate can be realized using the true conditional probability distribution of a *secret intermediate value* given a side-channel trace [24,26,27]. This indicates that DL-SCA can achieve such an optimal attack if the attacker can train a neural network (NN) model perfectly, as one of the goals of DL is to approximate the true conditional probability distribution. Currently, investigating the theory and practice of DL-SCAs is essential to understand the threat of SCAs against cryptographic implementation.

Profiled vs. non-profiled SCAs. DL-SCA was initially presented and has been mainly developed as a profiled attack [36]. The template attack [11] is the first profiled attack, which estimates the characteristics of side-channel leakage (*i.e.*, the conditional probability distribution) from a profiling device, and then estimates secret key using the side-channel leakage from the target device in the attack phase. DL-SCA is advantageous because DL is currently known to be a strong tool to estimate the characteristics as a conditional probability. In comparison to the template attack, it requires neither specific assumptions nor detailed knowledge of the target device. A major drawback of profiled attacks is the requirement of a profiling device in which the attacker can know or control the secret key.[3] Thus, the applicational scope and scenario of DL-SCAs are limited in comparison to non-profiled attacks (*e.g.*, differential/correlation power analysis (DPA/CPA) [7, 28]), although some literature has shown and discussed the possibility and potential of profiling attacks in practical scenarios (*e.g.*, [16, 45, 67]).

Differential deep-learning-based analysis (DDLA). In [58], Timon developed a non-profiled SCA using supervised DL, called differential deep-learning analysis (DDLA). Thereafter, several studies have been devoted to the improvement of DDLA performance (*e.g.*, [1, 17, 18, 25, 30–33, 62, 65]). Currently, DDLA-type attack is one of the most promising non-profiled DL-SCAs ever known. It was

[3] In this study, we define an SCA as a profiled attack if the SCA requires a profiling device in which the attacker can know or control the secret key. Conversely, we define an SCA as a non-profiled attack if it neither requires such a profiling device nor controls the secret key of the target device. Even non-profiled attacks may train a model, but the training dataset is acquired from the target device without knowing nor controlling the secret key.

demonstrated in [15,58] that DDLA can recover the secret key with fewer traces than CPA, which is the most popular non-profiled attack. It was also shown that DDLA is highly robust to translation of side-channel trace caused by, for example, jitter or random delay-based countermeasure. However, theoretical analysis of DDLA has not been well established: the efficiency of DDLA has not been proven explicitly nor analytically and how much DDLA exploits the advantage of DL is unclear. Additionally, it is unclear when, how, and why the DDLA works effectively. For example, major SCAs usually have a higher performance if the signal-to-noise ratio (SNR) of side-channel leakage measurement is higher. However, it is not always true for DDLA, as mentioned in Section 3.

Collision DL-SCA. In [54], Staib and Moradi presented another non-profiled SCA with supervised DL, namely collision DL-SCA. It is a DL-based variant of conventional collision SCAs such as correlation-enhanced power analysis collision attacks (CEPACA) [39] and moments-correlating collision DPA (MCC-DPA) [41]. In the collision DL-SCA, the attacker trains an NN to infer a plaintext from side-channel leakage of an Sbox computation, then estimates the difference between key values of the profiled byte and other bytes. The collision DL-SCA has been demonstrated to achieve a successful key recovery from a software (*i.e.*, serialized) implementation. However, there exists no formal analysis of their attack methodology (*e.g.*, why, when, and how the attack succeeds), although some intuitive descriptions of the attack methodology are available. Furthermore, due to a lack/limitation of a public dataset adequate for collision SCA evaluation, a comprehensive performance evaluation of collision (DL-)SCAs has been difficult. For example, success rate [55] is a common metric to evaluate and compare SCAs; however, in [54], the existing public dataset is insufficient for collision SCAs to evaluate the success rate due to limited traces for different/various key values. Thus, formal analyses and comprehensive performance evaluations on the collision DL-SCA would clarify its advantage in terms of both theoretical and practical perspectives.

1.2 Our contributions

This paper provides the first formal study on non-profiled DL-SCAs, namely, DDLA and collision DL-SCA. In addition, we propose a performance improvement of collision DL-SCA by designing a dedicated NN architecture and key estimation method. The contributions of this study are fourfold.

Information-theoretical analyses on DDLA. Although some literature demonstrated the advantage of DDLA over major non-profiled SCAs (*e.g.*, CPA), limited theoretical analyses on DDLA are available, which makes it challenging to prove its validity, optimality, and conditions for attack success. This paper first provides a formal analysis of DDLA through some proofs from an information-theoretical perspective. Our theorems reveal its relevance of MIA, and claim some limitations and impossibility results of DDLA.

Probability-theoretical analysis on collision DL-SCA. We formalize the collision DL-SCA from a probability-theoretical perspective. This formalization allows us to theoretically explain why the attack succeeds, clarify its applicability, and prove its optimality. This is the first formal analysis on collision DL-SCA.

Improvement of collision DL-SCA. We propose a method to improve the collision DL-SCA on the basis of our formal analyses. Our analyses reveal that there is a more efficient distinguisher for collision DL-SCAs than the conventional method presented in [54]. In addition, we propose a novel NN architecture to decrease the number of side-channel measurements required for training (equivalently that for attacking), which allows the collision DL-SCA to recover the secret key with fewer traces than the conventional method.

Comprehensive success rate evaluation. We demonstrate the validities of our theories through some experimental attacks on the same AES implementation on ARM Cortex-M4 as ASCAD, which is one of the most common public datasets in evaluating DL-SCA [4]. In this paper, we call it ASCAD implementation. So far, we have had difficulty in evaluating collision SCAs due to a lack of public datasets containing a sufficient number of side-channel traces for different key values[4]. Thus, we newly create a dataset using ASCAD implementation to comprehensively evaluate the success rate of collision SCAs. The evaluation results confirm the improvement of our collision DL-SCA and clarify the current situation of non-profiled (DL-)SCA.

The source code and dataset used in our experiment are publicly available at [blinded for the review].

1.3 Existing studies on non-profiled DL-SCA

In this study, we focus on a non-profiled DL-SCA on symmetric cipher (especially, AES), although DL-SCA on public key cryptography has also been actively studied (*e.g.*, [10, 19, 34, 42, 43, 52, 57, 60, 63, 64, 69]).

As aforementioned, DDLA is a major non-profiled DL-SCA, which was followed by many studies such as [1, 17, 18, 25, 30–33, 62, 65]. Such DDLA variants have attempted to improve the DDLA performance by using several unique techniques, such as the preprocess of side-channel trace, noise reduction, multi-output NN (or multi-task NN), and advanced NN architecture/training methods. Nevertheless, they basically train NNs towards the same (or very similar) goal as the original DDLA; hence, the utilized distinguisher is essentially equivalent to DDLA. While the experimental and engineering aspects of the distinguisher have been extensively studied, its theoretical effectiveness and optimality are yet to be well established.

In contrast to DDLA and its variants using *supervised* learning, another non-profiled DL-SCA using *unsupervised* learning has been presented in [48, 49],

[4] Although the ASCAD dataset includes a variable-key dataset, the number of traces for each key value is too small to evaluate collision SCAs.

called SCAUL and SCARL. The method first trains an autoencoder on side-channel traces, then performs clustering on the autoencoder outputs with a guess of the secret key, and finally estimates a correct key as the candidate with the highest score in the clustering. This type of non-profiled DL-SCA is evaluated only on non-protected implementation; its effectiveness and extension for protected (*e.g.*, masked) implementation are unknown.

Recently, a new collision DL-SCA has been presented in [54]. The collision DL-SCA utilizes supervised learning in a way different from DDLA. That is, the conditional probability distribution estimated by the collision DL-SCA is different from DDLA, and the distinguishers are also essentially different from each other. However, its theoretical validity is still unknown because of a lack of analysis, discussion, and formal proof. In addition, its performance in terms of success rate and the number of traces for full-key recovery have not been comprehensively evaluated in the literature compared with the conventional non-profiled SCAs, due to a lack of appropriate public datasets. Therefore, this paper provides formal analyses and comprehensive performance evaluations for our proposed DL-SCA.

1.4 Paper organization

The rest of this paper is organized as follows: Section 2 introduces DL-SCA. Section 3 and Section 4 provide formal analyses on DDLA and collision DL-SCA, respectively. Section 5 presents improvements of collision DL-SCA accordingly. Section 6 conducts a performance evaluation of major non-profiled (DL-)SCAs, namely, CPA, collision SCAs, and DDLAs, and compares our improved collision DL-SCA with the conventional one. Finally, Section 7 concludes this paper.

2 Preliminaries

2.1 Notation

A calligraphic letter (*e.g.*, $\mathcal{X}$) represents a set; an uppercase variable (*e.g.*, X) represents a random variable over the corresponding set (*i.e.*, X for $\mathcal{X}$); and a lowercase variable (*e.g.*, x) is an element of the corresponding set (*i.e.*, $x \in \mathcal{X}$), unless defined otherwise. Pr denotes a probability measure. Throughout this paper, p denotes the true density or mass function, and q denotes the probability density or mass function represented by an NN. For example, the true probability mass function of discrete random variables X and Y is $p_{X,Y}(x, y) = \Pr(X = x, Y = y)$. We may omit the subscripted random variables if the random variables of the probability distribution are obvious. For example, we may write $p(x, y)$ instead of $p_{X,Y}(x, y)$. Let $\mathbb{E}$ denote the expectation operator. For example, $\mathbb{E}_X f(X)$ denotes the expectation of $f(X)$ in terms of X, where $f : \mathcal{X} \to \mathbb{R}$ is a (measurable) function. The conditional probability distribution is denoted by $p_{X|Y}(x \mid y) = p(x \mid y)$.

Let $\boldsymbol{X}$ denote a random variable of a side-channel trace. A side-channel traces is represented as a multi-dimensional real vector $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^{m_t}$, where $m_t \in \mathbb{N}$

is the number of sample points. This study focuses on SCAs on block ciphers, particularly AES. Let n_k be the bit length of the partial key, and let n_t be the bit length of the partial plaintext and ciphertext. The secret intermediate value is denoted as $z = \phi(k, t) \in \mathcal{Z} = \{0,1\}^{n_z}$, where ϕ is a selection function[5], integer n_z denotes the bit length of z, a bit string $k \in \mathcal{K} = \{0,1\}^{n_k}$ is a key, and a bit string $t \in \mathcal{T} = \{0,1\}^{n_t}$ is public information, such as plaintexts and ciphertexts. Their random variables are also defined in the aforementioned manner. Here, let K denote the random variable of a key. T and K are assumed to have uniform distributions. If we require to specify the key value for Z, we write $Z^{(k)} = \phi(k, T)$. We assume that we have the Markov chain $(T, K) \to Z \to \boldsymbol{X}$. In this paper, we consider a case that the selection function ϕ is a bijection when the input/output t or the partial key k is fixed, which is true in many major SCAs (*e.g.*, SCA on AES software implementation). This assumption indicates that the distribution of Z is independent of the key used. Many practical selection functions can be proven to satisfy this condition. For example, a typical selection function for each byte of software AES implementation (*i.e.*, $Z = \mathrm{Sbox}(K \oplus T)$) satisfies this condition.

2.2 Overview of profiled DL-SCA

This subsection briefly reviews profiled DL-SCAs closely related to non-profiled DL-SCAs. A profiled DL-SCA has two phases: profiling and attack phases. During the profiling phase, we train an NN to model the conditional distribution corresponding to the device leakage characteristics. Let $\mathcal{S}_p = \{ (\boldsymbol{X}_l, Z_l) \mid 1 \le l \le m_{\mathsf{pro}} \}$ be a training dataset used in the profiling phase, $\boldsymbol{X}_l$ denotes the side-channel trace (*e.g.*, power consumption and electromagnetic radiation) of the l-th observation, Z_l denotes the corresponding intermediate value, and $|\mathcal{S}_p| = m_{\mathsf{pro}}$ is the number of traces used in the profiling phase. We assume that $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{m_{\mathsf{pro}}}$ and $Z_1, Z_2, \ldots, Z_{m_{\mathsf{pro}}}$ are independent and identically distributed (i.i.d) random variables over $\mathcal{X}$ and $\mathcal{T}$, respectively. Let $\theta \in \mathbb{R}^{n_\theta}$ be the NN model parameters, where n_θ is the dimension of the parameters. The profiling phase aims to estimate the optimal model parameters $\hat{\theta}$ using the training dataset $\mathcal{S}_p$. The optimal parameters are usually given by solving the minimization problem of the cross entropy (CE) loss function, defined as

$$\mathrm{CE}(\theta) = -\mathbb{E}_{Z,\boldsymbol{X}} \log q(Z \mid \boldsymbol{X}; \theta) = -\int \sum_z p_{Z,\boldsymbol{X}}(z, \boldsymbol{x}) \log q(z \mid \boldsymbol{x}; \theta)\, d\boldsymbol{x}, \quad (1)$$

where Z and $\boldsymbol{X}$ are the random variables of a label z and a trace $\boldsymbol{x}$, respectively, and q_θ represents the conditional probability distribution modeled by the NN with the parameters θ.

[5] In this paper, we assume that ϕ does not consist in a *leakage function* (*e.g.*, Hamming weight). The selection function is defined to specify the intermediate value (including the register transition related to it), which is utilized in the SCA. Selection and leakage functions are distinctly considered.

$\mathrm{CE}(\theta)$ in Equation (1) takes the minimum value if and only if $p_{Z|\boldsymbol{X}} = q(\cdot \mid \cdot\,; \theta)$ [6, 22]. We can obtain a model that approximates the true distribution p if we can determine the optimal parameters $\hat{\theta}$ that make $\mathrm{CE}(\hat{\theta})$ sufficiently small; however, we cannot calculate Equation (1) because it contains the integral and summation of the unknown probability distribution p. Therefore, in general, we approximate $\mathrm{CE}(\theta)$ using the training data $\mathcal{S}_p$ as follows:

$$\mathrm{CE}(\theta) \approx L(\theta) = -\frac{1}{m_{\mathsf{pro}}} \sum_{l=1}^{m_{\mathsf{pro}}} \log q(Z_l \mid \boldsymbol{X}_l; \theta). \tag{2}$$

The approximated CE in Equation (2) is called negative log-likelihood (NLL). The NLL is expected to converge in probability to $\mathrm{CE}(\theta)$ as $m_{\mathsf{pro}} \to \infty$ for fixed q_θ, according to the weak law of large numbers.

During the attack phase, we estimate the secret key k^* of the target device using the trained model. Let $\mathcal{S}_a = \{\, (\boldsymbol{X}_i, T_i) \mid 1 \le i \le m_{\mathsf{atk}} \,\}$ be a dataset used during the attack phase, where $|\mathcal{S}_a| = m_{\mathsf{atk}}$ is the number of attack traces, $\boldsymbol{X}_i$ is the side-channel trace at the i-th observation, and T_i is the corresponding plaintext or ciphertext. During the attack phase, we calculate the NLL for each hypothetical key candidate $k \in \mathcal{K}$ using the intermediate value $Z_i^{(k)}$ calculated from T_i as

$$L^{(k)}(\hat{\theta}) = -\frac{1}{m_{\mathsf{atk}}} \sum_{i=1}^{m_{\mathsf{atk}}} \log q(Z_i^{(k)} \mid \boldsymbol{X}_i; \hat{\theta}). \tag{3}$$

Thereafter, the correct key is estimated to be the key candidate with the smallest NLL value. This is equivalent to approximate computing and comparing

$$\mathrm{CE}^{(k)}(\theta) = -\mathbb{E} \log q(Z^{(k)} \mid \boldsymbol{X}; \hat{\theta}),$$

for each key candidate k.

For simplicity, we henceforth denote the number of traces by m instead of m_{atk} and m_{pro}, if obvious.

3 Information-theoretical analyses on DDLA

3.1 Overview

After introducing DDLA in Section 3.2, we formalize it in Section 3.3, and then discuss the three problems associated with the DDLA from an information-theoretical perspective. (i) DDLA must require some information loss in the leakage function for a successful attack (Section 3.4). (ii) Simultaneous estimation of the leakage function and correct key during the attack phase is impossible (Section 3.5). Finally, (iii) DDLA does not work in the ideal case for the attacker at all, where the side-channel leakage contains no noise and the secret intermediate values unboundedly leak out from side-channel traces (Section 3.6). These analyses reveal some limitations and impossibility results of the DDLA, as concretely claimed in the last paragraph of each subsection entitled "Intuitive meanings." Some proofs of this section are described in Section D.

3.2 Introduction to DDLA

DDLA is a major and pioneering non-profiled DL-SCA [58]. In non-profiled attacks, the attacker attempts the key recovery solely from an attack dataset because the profiling dataset (or training dataset) is unavailable in the scenarios. The DDLA employs a leakage function from the secret intermediate value to a leakage $g : \mathcal{Z} \to \mathcal{Y}$, where $\mathcal{Y}$ is a set of leakage values (*e.g.*, Hamming weight (HW) and least/most significant bit (LSB/MSB)). In DDLA, we train an NN that estimates $Y^{(k)}$ from $\boldsymbol{X}$ for each key candidate k (meaning that we train $|\mathcal{K}|$ NNs). The attacker estimates the correct key as a candidate that minimizes the loss function, because it is supposed that the NN training with a correct key guess would be the best in inferring the secret intermediate value $Y^{(k)}$.

It should be noted that a major known drawback of DDLA is to require a huge computational cost, because it performs the NN training for each key candidate. In the case of AES, the attacker/evaluator should perform the NN training 256 times for a key byte. This indicates that DDLA requires $4{,}096 = 256 \times 16$ NN trainings, which incurs a non-negligibly high computational cost.

3.3 Formalization

Let $g : \mathcal{Z} \to \mathcal{Y}$ be a leakage function, where $\mathcal{Z}$ is the set of all the intermediate values and $\mathcal{Y}$ is the set of all the leakage values. Existing studies on DDLA used the LSB, MSB, and HW of the intermediate value as the (hypothetical) leakage function g. Note that we never assume any relation between the leakage function g and the side-channel information $\boldsymbol{X}$. For example, some prior works [13, 47] assume that there exists a true leakage function g_{true} and that the side-channel information can be represented as $\boldsymbol{X} = g_{\text{true}}(Z^{(k^*)}) + N$, where N is some noise. In contrast, we do not even assume such a true leakage function exists. In this paper, we consider the leakage function to be a function that the attacker hypothetically uses for convenience to extract from the intermediate values the information necessary for DDLA to succeed. Therefore, the leakage function does not necessarily have to represent the physical leakage information, and can be any function such that the DDLA is successful. For example, a DDLA on the first-order masked implementation with the LSB of the intermediate value succeeds even though the side-channel traces would not contain sample points proportional to the LSB value [58].

Although any performance metrics, such as accuracy and a loss value, can be used to perform DDLAs, we focus on DDLAs with the CE loss function to simplify our analysis. This means that the DDLA attacker identifies the correct key candidate k which minimizes the loss function $-\sum_i \log q(Y_i^{(k)} \mid \boldsymbol{X}_i; \theta^{(k)})$. In this section, we suppose that the attacker is assumed to use an infinite number of traces for attacks and perfectly train the NN models (*i.e.*, we have $q(\cdot, \cdot; \theta^{(k)}) = p(\cdot, \cdot)$), since we are interested in understanding the limit of the DDLA capability. Thus, the estimated key in DDLA with a leakage function g is denoted as

$$\hat{k} = \arg\min_k -\mathbb{E}_{Y^{(k)}, \boldsymbol{X}} \log p_{Y|\boldsymbol{X}}(Y^{(k)} \mid \boldsymbol{X}).$$

Recall that, for any fixed correct key k, the intermediate value $Z^{(k)}$ is uniformly distributed because T is supposed to be distributed uniformly. Therefore, the distribution of the leakage $Y^{(k)} = g(Z^{(k)})$ is independent of the key value, and its entropy $H(Y^{(k)})$ should be constant for any k. This indicates that the DDLA estimates the correct key as

$$\begin{aligned}\hat{k} &= \arg\min_k -\mathbb{E}\log p_{Y|\boldsymbol{X}}(Y^{(k)} \mid \boldsymbol{X}) \\ &= \arg\max_k \Big(H(Y^{(k)}) - H(Y^{(k)} \mid \boldsymbol{X})\Big) \\ &= \arg\max_k I(Y^{(k)}; \boldsymbol{X}). \end{aligned} \tag{4}$$

Equation (4) reveals that the DDLA is a variant of mutual information analysis (MIA), in which the attacker identifies the key candidate that maximizes the mutual information $I(Y^{(k)}; \boldsymbol{X})$ [20]. In fact, the DDLA has the same restrictions of the MIA as we will show in the following subsections. In the following, we analyze the mutual information.

For the analyses, we introduce a functional representation of a partial key difference. Recall that a selection function $\phi : \mathcal{K} \times \mathcal{T} \to \mathcal{Z}$ is assumed to be a bijection when a secret key k or an input t is fixed, which implies that $\phi(k, \cdot)$ has an inverse function $\phi^{-1}(k, \cdot)$ for any fixed key k. This assumption holds for many practical AES implementations as discussed in Section 4.6. We denote $\phi_k = \phi(k, \cdot)$ and $\phi_k^{-1} = \phi^{-1}(k, \cdot)$. Then, we define the functional form of the key difference as follows.

Definition 1. *Let $(k, k') \in \mathcal{K} \times \mathcal{K}$ be a secret key pair. The key difference of the key pair (k, k') is defined by $\delta_{k,k'} = \phi_k^{-1} \circ \phi_{k'}$, where $\circ$ denotes the composition operator.*

We may omit the subscripts of $\delta_{k,k'}$ if they are unimportant for understanding. Here, the key difference is defined as a bijection function δ instead of a value. This abstract definition is because we do not assume a concrete structure for the selection function. In fact, this definition is an extension of the traditional key difference. For example, as in AES, if the selection function $\phi(k, t)$ is given by $\mathrm{Sbox}(k \oplus t)$ for any t, we have

$$\delta_{k,k'}(t) = \phi_k^{-1} \circ \phi_{k'}(t) = \mathrm{Sbox}^{-1}(\mathrm{Sbox}(t \oplus k')) \oplus k = t \oplus k \oplus k'. \tag{5}$$

Equation (5) indicates that the form of the key difference δ is determined by the value of XORed keys $k \oplus k'$. Thus, the function $\delta(\cdot)$ represents the key difference as a function (note that a one-to-one correspondence exists between $\delta(\cdot)$ and the key difference value).

Then, we introduce the following assumption for the key differences.

Assumption 1 *Let $k_1, k_2, k_3 \in \mathcal{K}$ be any partial keys. There exists a partial key $k_4 \in \mathcal{K}$ such that $\phi_{k_1}^{-1} \circ \phi_{k_2} = \phi_{k_3}^{-1} \circ \phi_{k_4}$ holds.*

The assumption guarantees the existence of a key difference starting from k_3, which corresponds to the key difference between any key pair k_1 and k_2. This is true for many selection functions of major ciphers. For example, if the selection function is given by $\phi_k(t) = \mathrm{Sbox}(k \oplus t)$ as in AES, the partial key $k_4 = k_1 \oplus k_2 \oplus k_3$ satisfies the assumption. Note that this assumption is equivalent to the key independence condition (KIC) presented by Ito *et al.* in [26]. This assumption is natural because the number of key differences $|\{\, \phi_{k_1}^{-1} \circ \phi_{k_2} \mid k_1, k_2 \in \mathcal{K} \,\}|$ is equal to that of the key candidates $|\mathcal{K}|$, as shown in Proposition 1. The proof of Proposition 1 is described in Section D.1.

Proposition 1. *From Assumption 1, we have* $|\{\, \delta_{k_1,k_2} = \phi_{k_1}^{-1} \circ \phi_{k_2} \mid k_1, k_2 \in \mathcal{K} \,\}| = |\mathcal{K}|$.

3.4 Unavoidable information loss in leakage function

Similary to MIAs, the performance of DDLA depends greatly on the choice of leakage function g. For example, if the side-channel trace contains little information on the MSBs of the intermediate values, the MSBs should not be used as the leakage function. The leakage function g should be chosen appropriately depending on what information on the intermediate values is contained in the side-channel traces. This is obvious from the fact that, the larger the amount of mutual information $I(Y^{(k^*)}; \boldsymbol{X})$ for the correct key k^*, the more likely the DDLA will select the correct key, as shown in Equation (4). For an inappropriate leakage function g, the mutual information value $I(Y^{(k^*)}; \boldsymbol{X})$ would be small. Conversely, the mutual information value would be large if an appropriate function is used. Therefore, intuitively, the leakage function g should be selected such that the mutual information $I(Y^{(k^*)}; \boldsymbol{X})$ is maximized.

However, we reveal below that this intuition is not necessarily correct; that is, a leakage function g that maximizes $I(Y^{(k^*)}; \boldsymbol{X})$ sometimes results in a trivial success rate (as low as a random guess), and g must cause an *information loss* for a successful DDLA. Specifically, we will show the following two facts. The first is that the mutual information $I(Y^{(k^*)}; \boldsymbol{X})$ is maximized when the leakage function g is injective (Proposition 2). This seems to imply that the most powerful attack can be conducted using an injective leakage function. However, this is not true. In other words, we then show that the DDLA with an injective leakage function always fails regardless of the relation between the side-channel information $\boldsymbol{X}$ and the intermediate value Z (Theorem 1). Therefore, an injective function must not be used as the leakage function in DDLA.

It should be noted that similar facts are already known in previous studies on formal analysis of MIA [13, 47]. These previous studies on MIA needed some assumptions like a specific form for leakage (*e.g.*, $\boldsymbol{X} = g(Z^{(k^*)}) + N$ holds, where N is noise) to prove these facts. However, such assumptions are not essential for DDLAs because they can succeed regardless of whether such assumptions hold. Actually, it has been shown that DDLA can successfully retrieve the secret key from the implementations with some countermeasures such as masking and random delay [58], where the assumptions would not hold. Therefore, we will prove these facts in a more general case than in the previous studies.

To this end, we prove that the mutual information $I(Y^{(k^*)}; \boldsymbol{X})$ is maximized if g is an injective function, which loses no information on intermediate values.

Proposition 2. *The mutual information $I(Y^{(k^*)}; \boldsymbol{X})$ is maximized if the leakage function g is an injection.*

Proof. Let $\operatorname{Im} g := g(\mathcal{Z})$ be the image of the functions g. Because g is an injective function from $\mathcal{Z}$ to $\mathcal{Y}$, the function $g' : \mathcal{Z} \ni z \mapsto g(z) \in \operatorname{Im} g$ whose codomain is restricted in the image $\operatorname{Im} g$ is a bijection function over this image. Therefore, the equality $I(\boldsymbol{X}; Y^{(k^*)}) = I(\boldsymbol{X}; Z^{(k^*)})$ holds.

Proposition 2 states that using an injective function as the leakage function g can extract most information about the intermediate values in the side-channel traces. Here, the leakage function g can take any injective function; the identity function of $\mathcal{Z} \to \mathcal{Y}$ is the simplest one (note that $\mathcal{Z} = \mathcal{Y}$ in this case). An identity function is an attractive choice in a non-profiled SCA such as DDLA, where the attacker is assumed to have no information about the target device. However, Theorem 1 implies that DDLA with an injective leakage function always results in a trivial success rate.

Theorem 1. *Let g be a leakage function. If g is an injective function, then it holds that*

$$I(Y^{(0)}; \boldsymbol{X}) = I(Y^{(1)}; \boldsymbol{X}) = \cdots = I(Y^{(|\mathcal{K}|-1)}; \boldsymbol{X}).$$

The proof of this theorem is described in Section D.2.

Intuitive meanings: Theorem 1 states that DDLA with an injective leakage function cannot distinguish the correct key and any wrong key at all from the mutual information $I(Y^{(k)}; \boldsymbol{X})$ (*i.e.*, the value of loss function). In other words, the DDLA cannot achieve a non-trivial success rate unless the leakage function g is not injective. Thus, the leakage function g in DDLA must incur an information loss on the intermediate value $Z^{(k^*)}$ for a successful attack. This information loss would cause a degradation of the success rate of key recovery, compared to an optimal attack that fully exploits the leakage.

3.5 Optimization difficulty of DDLA

The problem pointed out in Section 3.4 arises because DDLA is successful only if $I(Y^{(k^*)}; \boldsymbol{X}) > I(Y^{(k)}; \boldsymbol{X})$ for any key candidate $k \in \mathcal{K} \setminus \{k^*\}$. In other words, in DDLA, it is mandatory to choose a hypothetical leakage function g such that the attacker can distinguish the mutual information of the correct key from other key candidates, rather than a hypothetical leakage function that maximizes the mutual information of the correct key. Namely, a leakage function g that maximizes $I(Y^{(k^*)}; \boldsymbol{X}) - \max_{k \neq k^*} I(Y^{(k)}; \boldsymbol{X})$ should be used in DDLA. Indeed, the recent study [13] pointed out the importance of the difference in mutual information and provided some interesting analyses of the difference. In this

study, we want to analyze whether the appropriate hypothetical leakage model g and the secret key k^* can simultaneously estimated in terms of the difference $I(Y^{(k^*)}; \boldsymbol{X}) - \max_{k \neq k^*} I(Y^{(k)}; \boldsymbol{X})$. Note that this point is not discussed in [13]. Naturally, the leakage function g strongly depends on how secret information is leaked from the device, and it would be difficult to find such g in a non-profiled scenario. In fact, Theorem 2 states that estimating the appropriate leakage function g and the correct key k^* is impossible, even if an infinite number of traces are available for the attack.

Theorem 2. *Define a function* $h_g(k) = I(g(Z^{(k)}); \boldsymbol{X}) - \max_{k' \neq k} I(g(Z^{(k')}); \boldsymbol{X})$. *Let* k_1 *and* k_2 *be any two different keys. There exist two functions* $g_1 = \arg\max_g h_g(k_1)$ *and* $g_2 = \arg\max_g h_g(k_2)$ *such that* $h_{g_1}(k_1) = h_{g_2}(k_2)$.

The proof of Theorem 2 is described in Section D.3.

Intuitive meanings: Theorem 2 states that, for any key candidate k, there exists a leakage function g that maximizes $I(Y^{(k)}; \boldsymbol{X}) - \max_{k' \neq k} I(Y^{(k')}; \boldsymbol{X})$, and the value of the mutual information $I(Y^{(k)}; \boldsymbol{X}) - \max_{k' \neq k} I(Y^{(k')}; \boldsymbol{X})$ has an identical maximum value independent of k. Therefore, in DDLA, no methods are available for finding the correct key and the leakage function g simultaneously. Thus, DDLA essentially requires some knowledge or assumption on leakage characteristics to determine an appropriate g, although it is a non-profiled attack. This indicates that the performance of DDLA depends on the preciseness of the assumption on leakage to determine g, in contrast to profiled DL-SCA and collision DL-SCA. In Section 6, we demonstrate that the choice of g (LSB or MSB) significantly impacts the success rate, which reveals the instability of DDLA as a non-profiled SCA.

3.6 Contradiction of DDLA

In this section, we prove that DDLA always fails in an ideal situation for the attacker, where the side-channel leakage contains no noise and is unbounded, such that the intermediate value can be uniquely identified from the side-channel trace.

If a function f exists such that $Z^{(k^*)} = f(\boldsymbol{X})$ holds with probability 1, we say that the intermediate value $Z^{(k^*)}$ can be uniquely identified from the side channel trace $\boldsymbol{X}$. Obviously, a usual SCA attacker (*e.g.*, CPA, DPA, and even simple power analysis (SPA)) trivially wins in the key recovery in the ideal situation with such an unbounded leakage. However, if such a function f exists, then $I(Z^{(k)}; \boldsymbol{X}) = H(Z^{(k^*)})$ holds for any k. Hence, the DDLA cannot distinguish the correct and any wrong keys, regardless of the leakage function g, as shown in Theorem 3. The proof of this theorem is shown in Section D.4.

Theorem 3. *Assume that there exists a function* f *such that* $Z^{(k^*)} = f(\boldsymbol{X})$ *holds with probability 1. Then, for any leakage function* g*, it holds that*

$$I(Y^{(0)}; \boldsymbol{X}) = I(Y^{(1)}; \boldsymbol{X}) = \cdots = I(Y^{(|\mathcal{K}|-1)}; \boldsymbol{X}) = H(g(Z^{(k^*)})).$$

Intuitive meanings: Theorem 3 states that DDLA has the contradiction as an SCA: it does not always work at all in an ideal situation for the attacker, in which other usual SCAs trivially win. This indicates that the DDLA is not always optimal in terms of success rate. In addition, this also indicates that, for a successful DDLA, side-channel leakage must contain some noise and be bounded in some way, and it is an open problem what types of noise/bound improve or deteriorate the performance of DDLA. This implies that the DDLA performance is not always improved when the device is more leaky, in contrast to usual SCAs. Thus, its theoretical and practical advantages over other non-profiled SCAs still remain unclear.

4 Formal analyses on collision DL-SCA

4.1 Overview

Staib and Moradi proposed a collision DL-SCA, which is one of the non-profiled SCA based on supervised learning in [54]. In the literature, its performance was experimentally evaluated; however, the theoretical and formal aspects of collision DL-SCA were not discussed, and the conditions for its applicability and the optimality of the attack are unclear. In this section, we show its formal analyses through three items. First, we introduce a formal definition of collision DL-SCA distinguisher (Section 4.3). In the formal definition, we propose a distinguisher almost the same as but slightly different from the one proposed in [54] to achieve an optimal distinguisher. Then, we prove the optimality of our collision DL-SCA distinguisher, which establishes our distinguisher as a better choice than the one proposed in [54] (Section 4.4). This optimality proof shows sufficient conditions for the application of collision DL-SCA (Section 4.5). Finally, we investigate the applicability of collision DL-SCA to various AES implementations (Section 4.6).

4.2 Description of collision DL-SCA

We first review the collision DL-SCA proposed by Staib and Moradi in [54]. Although their proposal included a leakage identification method, we only focus on their distinguisher in this paper.

In a non-profiled scenario, the attacker does not know the secret key k' of a byte when the attacker prompts to perform an NN training. To calculate the label (*i.e.*, intermediate value Z) for the training of NN, we need the correct key value k' actually used on the device. This indicates the attacker cannot train an NN to imitate $p_{Z'|\boldsymbol{X}'}$ in a non-profiled scenario. To avoid this problem, the basic idea of the collision DL-SCA is to train an NN to predict plaintext/ciphertext T' for a given side-channel trace $\boldsymbol{X}'$ and unknown fixed key k'.[6] Thus, the

[6] When training a DNN-based classifier, we usually perform model training such that it approximates a conditional probability distribution. In this case, the model is expected to approximate the conditional probability $p_{T|\boldsymbol{X}}$ to predict T for a given $\boldsymbol{X}$. However, this is not true because side-channel leakages depend not only on plain-

collision DL-SCA utilizes the conditional probability distribution $p_{T'|\boldsymbol{X}',K'}$ for an unknown fixed key k' instead of $p_{Z'|\boldsymbol{X}'}$. Because the attacker knows the plaintext/ciphertext in a usual non-profiling scenario, the attacker can perform an NN training to imitate the probability distribution using a dataset $(\boldsymbol{X}'^m, T'^m)$ acquired from the target device, while the attacker does not know the value of k'.

The collision DL-SCA focuses on another Sbox with a key value k, and estimates the difference between k' and k as $\delta = k' \oplus k$.[7] Namely, consider $(\boldsymbol{X}^m, T^m)$ as an attack dataset of another partial plaintext and the corresponding Sbox leakage with another correct key k. Staib and Moradi proposed an estimation method of the partial key difference $\delta = k' \oplus k$ according to the probabilities of $T \oplus \delta$ as follows:

$$d(\boldsymbol{X}^m, T^m; \hat{\theta}) = \arg\max_{\delta} \sum_i q(T_i \oplus \delta \mid \boldsymbol{X}_i; \hat{\theta}), \tag{6}$$

where $q(\cdot \mid \cdot; \hat{\theta})$ is the probability distribution modeled by the NN with the trained model parameter $\hat{\theta}$, which approximates the true conditional distribution $p_{T'|\boldsymbol{X}',K'}$. For example, in the case of SCA on AES, we would consider k' as the first partial byte of the key, while k is of other bytes. The reason why the key difference δ can be predicted by using the conditional distribution $p_{T'|\boldsymbol{X}',K'}$ is explained in Section 4.3.

4.3 Formal definition of collision DL-SCA

We first explain the attack procedure of the collision DL-SCA from a probability-theoretical perspective for the sake of formalization. Let T', Z', k', and $\boldsymbol{X}'$ be a plaintext/ciphertext, secret intermediate value, partial key, and side-channel trace, respectively. Also, let T, Z, k, and $\boldsymbol{X}$ be another byte of the plaintext/ciphertext, secret intermediate value, partial key, and their side-channel trace, respectively. In collision DL-SCA, we train an NN that infers T' from $\boldsymbol{X}'$. As mentioned in Section 2, we assume that the Markov chain $(T', k') \to Z' \to \boldsymbol{X}'$ holds, which corresponds to the NN training. A Markov chain in the opposite direction $\boldsymbol{X}' \to Z' \to (T', k')$ also holds, which corresponds to the NN inference. Here, if we input $\boldsymbol{X}$ to the trained NN, the corresponding Markov chain forms $(T, k) \to Z \to \boldsymbol{X} \to \hat{Z} \to (\hat{T}, k')$, where $\hat{Z}$ and $\hat{T}$ are the estimations of Z and T, respectively. Here, the former chain $(T, k) \to Z \to \boldsymbol{X}$ represents a physical phenomenon of Z leaking through $\boldsymbol{X}$, while the latter chain $\boldsymbol{X} \to \hat{Z} \to (\hat{T}, k')$

text/ciphertexts T but also on the secret key k used in the device due to the Markov chain $(k, T) \to Z \to \boldsymbol{X}$. Therefore, the adversary trains the model to approximate the conditional probability distribution $p_{T|\boldsymbol{X},K}$, even if the adversary only uses the plaintext/ciphertext T and the side-channel trace $\boldsymbol{X}$.

[7] In this study, we define a key difference by the functional form $\phi_{k'}^{-1} \circ \phi_k$. However, this subsection describes the original collision DL-SCA proposed by Staib and Moradi [54]; hence, we denote $\delta = k' \oplus k$ instead of the functional form according to the original work.

represents the NN inference. In this chain, $\hat{Z}$ and $\hat{T}$ are estimated by assuming that k' is used for the secret key (although it is not true and k' is actually used), because the NN trained with a different byte is used for inference. Therefore, the value of $\hat{T}$ is basically inequivalent to that of T because of the difference between k and k' (except for the case that $k = k'$). Regarding the relation between $\hat{T}$ and T, it should hold that $Z = \hat{Z}$,[8] which is followed by $\phi(T, k) = \phi(\hat{T}, k')$. Therefore, letting $\delta_{k',k} = \phi_{k'}^{-1} \circ \phi_k$, we have $\hat{T} = \delta(T)$, which represents an collision SCA. Here, we can estimate the key difference δ because the adversary knows the plaintext T, and $\hat{T}$ is inferred by the trained NN.

We then formalize the attack procedure aforementioned and define the collision DL-SCA distinguisher. As mentioned, an attacker first trains an NN to infer plaintext/ciphertext T' from a side-channel trace $\boldsymbol{X}'$ for an unknown fixed key k'. Let $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$ be the true distribution to be modeled by the NN. Here, the conditioning on K' indicates that the side-channel traces are acquired when the partial key $K' = k'$ is used on the cryptographic device. Thus, the secret key k' is not given as input to the model. Let $q(T \mid \boldsymbol{X}; \theta)$ denote the NN to imitate $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$ with parameters θ. In the training phase, the attacker finds the parameters $\hat{\theta}$ such that $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k') \approx q(T \mid \boldsymbol{X}; \hat{\theta})$. The NN is trained in the same way as the ordinal NLL-based training in the profiled DL-SCA. After training, the attacker estimates the key difference $\delta = \phi_{k'}^{-1} \circ \phi_k$ using the trained NN $q(\cdot \mid \cdot; \hat{\theta})$. For this, the attacker inputs the side-channel trace $\boldsymbol{X}$ corresponding to another byte with a key k. If the NN has no estimation errors, then $\hat{T} = \delta(T)$ holds as aforementioned. Therefore, even if the NN inference contains errors, the NN output $q(t \mid \boldsymbol{X}; \hat{\theta})$ is likely to be maximized when $t = \delta_{k',k}(T)$. Thus, we define the distinguisher $d : \mathcal{X}^m \times \mathcal{T}^m \to \{0,1\}^{n_k}$ with likelihood function as

$$\begin{aligned} d(\boldsymbol{X}^m, T^m) &= \arg\max_{\delta \in \mathcal{D}} \prod_i q(\delta(T_i) \mid \boldsymbol{X}_i; \hat{\theta}) \\ &= \arg\max_{\delta \in \mathcal{D}} \sum_i \log q(\delta(T_i) \mid \boldsymbol{X}_i; \hat{\theta}), \end{aligned} \tag{7}$$

where $\mathcal{D} = \{\, \delta_{k',k} \mid k', k \in \mathcal{K} \,\}$ is a set of the key difference candidates. Note that our distinguisher in Equation (7) is given by the product of the probabilities although Staib and Moradi's distinguisher in Equation (6) is represented by the sum of the probabilities. The reason for the different form of our distinguisher is that, as shown later, ours is the optimal distinguisher whereas Staib and Moradi's distinguisher is not. In Equation (7), the attacker identifies the correct partial key difference according to the maximum likelihood among at most $|\mathcal{K}|^2$ candidates. If we assume that the selection function satisfies Assumption 1, the number of the key difference candidates can be reduced to $|\mathcal{K}|$ because we have

[8] If the NN is *ideally* accurate such that the inference result is always correct, then $Z = \hat{Z}$ holds. However, in practice, the NN includes some inference errors, which indicates the need for a maximum likelihood estimation using multiple traces to enhance the inference accuracy.

$|\mathcal{D}| = |\mathcal{K}|$. It should be noted that the distinguisher defined in Equation (7) uses the log-likelihood, while the previous study in [54] used the sum of probabilities. In Section 4.4, we prove that our distinguisher is better in terms of the optimal distinguisher.

Software AES implementation For example, in the case of the software AES implementation, the selection function is given by $\phi(t, k) = \mathrm{Sbox}(t \oplus k)$. Therefore, we have the key difference $\delta_{k,k'}(t) = k \oplus k' \oplus t$. Thus, Equation (7) can be rewritten as

$$d(\boldsymbol{X}^m, T^m) = \arg\max_{k \oplus k'} \sum_i \log q(k \oplus k' \oplus T_i \mid \boldsymbol{X}_i; \hat{\theta}).$$

Note that the number of key difference candidates does not depend on the secret key k because Assumption 1 holds.

4.4 Optimality of our distinguisher

This section describes the relationship between the collision DL-SCA and an optimal distinguisher. The collision DL-SCA utilizes an NN that approximates $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$; hence, the collision DL-SCA is validated from the perspective of optimality if we can construct an optimal distinguisher using $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$. This is given by Theorem 4.[9]

Theorem 4 (Optimal distinguish rule in estimating partial key difference). *Let $\boldsymbol{X}^m$ and T^m be side-channel traces and corresponding plaintexts used for the attack, respectively. Assume that we have a Markov chain $(K, T^m) \to (Z^m, T^m) \to (\boldsymbol{X}^m, T^m) \to \hat{K}$ as in [12], where K is a partial key, Z is an intermediate value, and $\hat{K}$ is an estimated key. Side channel leakage from each byte is assumed to occur independently and follow the same distribution. Suppose that Assumption 1 holds. Let k' be a secret key of another byte. Let Δ be a random variable for key difference δ in a functional form. Then, a distinguisher*

$$d(\boldsymbol{X}^m, T^m) = \arg\max_{\delta} \sum_i \log p_{T|\boldsymbol{X},K}(\delta(T_i) \mid \boldsymbol{X}_i, k')$$

is optimal in estimating Δ.

The proof of Theorem 4 is described in Section D.5. Theorem 4 theoretically validates the collision DL-SCA with our distinguisher in Equation (7) and implies that the collision DL-SCA would achieve a success rate as high as an optimal attack if the NN can sufficiently approximate $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$. In addition, the collision DL-SCA would have an SR as optimal as the profiled DL-SCA, if the number of traces available for the attack is sufficiently large to imitate $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$.

[9] In the theorem, the distinguisher is given by the conditional probability $p_{T|\boldsymbol{X},K}$ not $p_{T'|\boldsymbol{X}',K'}$. However, it does not matter because we have $p_{T|\boldsymbol{X},K} = p_{T'|\boldsymbol{X}',K'}$ as $(T', \boldsymbol{X}', K')$ can be seen as an independent copy of $(T, \boldsymbol{X}, K)$.

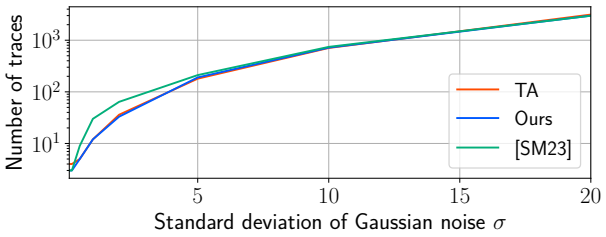

Fig. 1: Results of collision DL-SCAs and template attack (TA) on simulation data.

Experimental verification of optimality. We performed an experimental attack of our distinguisher using simulated side-channel traces to validate its optimality. In the simulation, we generated side-channel traces as $\boldsymbol{X} = \mathrm{HW}(Z) + N$, where $\mathrm{HW}(Z)$ is the Hamming weight of Z and N is a zero-mean Gaussian noise. For comparison, we also performed the template attack (TA) [11], which has been proven to be optimal in [24] if the noise is an additive Gaussian as in this simulation. We also compare them with the collision DL-SCA of Staib and Moradi in [54], which uses a sum of probabilities in Equation (6) as a distinguisher. We used a three-layer NN consisting of an input and hidden layers with 256 units and an output layer to model the probability distribution $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$. The ELU and softmax functions were used as the activation functions of the intermediate and the last layers, respectively. Ten million and one million traces were generated for training the model and estimating the secret key, respectively. Figure 1 shows the simulation result: the number of required traces to achieve an 80% SR by our distinguisher using $q(T \mid \boldsymbol{X}, \hat{\theta})$ and the TA for different noise variances. In Figure 1, the horizontal axis denotes the standard deviation for the noise. We confirmed that our distinguisher achieved the SR for a given number of traces, which was almost the same as that of the TA, as the curves for TA and Ours were fairly consistent. In addition, our distinguisher and TA achieve higher performance (*i.e.*, achieve a success rate of 80% with fewer attack traces) than the Staib–Moradi collision DL-SCA. This simulation experimentally validated the optimality of our distinguisher using $p_{T'|\boldsymbol{X}',K'}(T \mid \boldsymbol{X}, k')$, as stated by Theorem 4.

4.5 Conditions for attack applicability

The applicability of the collision DL-SCA depends on the following two conditions:

– The selection function $\phi(k, t)$ must be represented by a bijection for any fixed t or any fixed k and $\phi(k, t)$ should satisfy Assumption 1, which is equivalent to the *key-independence condition (KIC)* [26].

– The side-channel leakages for each Sbox computation are i.i.d and must be separable in the time domain.

These are sufficient conditions for our optimality proof shown in Section 4.4.

The first condition is related to the selection function. Some major selection functions are not bijection (as discussed in Section 4.6) although they satisfy Assumption 1. However, the attack would be still applicable if a bijective *redundant form* of the selection function exists. It is sufficient for the attack applicability that we can decompose the non-bijective selection function as $\phi = \beta \circ \alpha_k$, where $\alpha : \mathcal{K} \times \mathcal{T} \ni (k, t) \mapsto \alpha(k, t) \in \mathcal{T}$ is a bijection for any fixed k or any fixed t and $\beta : \mathcal{T} \to \mathcal{Z}$ is a surjection function. For simplicity, we often write $\alpha_k(t)$ instead of $\alpha(k, t)$. If the selection function can be decomposed into such a form, the first condition is satisfied by regarding α and β as a selection function and a part of a leakage function, respectively. We discuss the attack applicability in terms of the selection function or its redundant form in Section 4.6.

The second condition indicates that it is necessary that the side-channel leakage for the j-th byte Sbox computation contains only information about the j-th byte, but never contains information about other bytes (*i.e.*, the side-channel leakage for the j-th byte is independent of other bytes). This is because we should use distinct traces for NN training and δ estimation for each byte. It depends on the target implementation whether this condition holds, as described in Section 4.6 (which is actually true for some major AES implementations).

In addition, as mentioned in [54], for the applicability, each Sbox is computed serially such that the side-channel traces can be separated for each Sbox computation in the time domain. On the one hand, Sbox evaluation using a (masked) lookup table (including ASCAD implementation [4]) and bit- and share-slicing Sbox evaluation [2, 51], which are majorly used for masking, satisfy this condition. Hence, they can be targeted by the collision DL-SCA. On the other hand, the collision DL-SCA is inapplicable to Sbox evaluation using a technique called byte-slicing [29,53], which processes several Sboxes in parallel by storing multiple bytes into a register.

4.6 Concrete examples of AES

We hereafter consider an attack on various AES implementations, although the collision DL-SCA is applicable to other block ciphers in general. Let $T[j]$, $K[j]$, and $Z[j]$ denote the j-th byte partial plaintext, secret key, and Sbox output, where $Z[j] = \mathrm{Sbox}(T[j] \oplus K[j])$. Without loss of generality, let $k = k[1]$, indicating that we use the first byte to train an NN for example.

Software implementation. In attacking a software implementation of AES, the selection function is usually given by

$$\phi(K[j], T[j]) = \mathrm{Sbox}(T[j] \oplus K[j]). \tag{8}$$

Apparently, this selection function is a bijection for any fixed t or any fixed k, and meets Assumption 1; hence, the collision DL-SCA is applicable to this case

in terms of the selection function. Here, the key difference is given by $k \oplus k'$ because $\delta(t) = \phi_k^{-1} \circ \phi_{K[j]}(t) = k \oplus K[j] \oplus t$ for all t. Thus, a bijective map $\delta \mapsto k \oplus k'$ exists.

Byte-serial hardware implementation. Byte-serial hardware is a common architecture for low-area implementation, and has been frequently employed with masking, especially Threshold Implementation (TI) (*e.g.*, [5,14,40,56,59]). Byte-serial hardware usually processes each byte from the first- to sixteenth-byte[10], and the leakage function is given by XOR of two consecutive bytes, representing a byte-wise register switching. Therefore, in attacking byte-serial hardware, the selection function frequently consists of two consecutive bytes of plaintext and key and is defined as

$$\phi(K[j], K[j+1], T[j], T[j+1]) = \mathrm{Sbox}(T[j] \oplus K[j]) \oplus \mathrm{Sbox}(T[j+1] \oplus K[j+1]). \tag{9}$$

To check the applicability of the collision DL-SCA, we examine whether this selection function can be decomposed into two functions α_k and β. In this case, the applicability holds as we consider a 16-bit bijective redundant form of ϕ in Equation (9) like $\phi = \beta \circ \alpha_{K[j]\|K[j+1]}$, where

$$\begin{aligned}\alpha_{K[j]\|K[j+1]}(T[j] \parallel T[j+1]) &= \mathrm{Sbox}(T[j] \oplus K[j]) \parallel \mathrm{Sbox}(T[j+1] \oplus K[j+1]) \\ &= S((T[j] \parallel T[j+1]) \oplus (K[j] \parallel K[j+1])),\end{aligned}$$

and the function β consists of a XOR operation between $\mathrm{Sbox}(T[j] \oplus K[j])$ and $\mathrm{Sbox}(T[j+1] \oplus K[j+1])$. Here $\cdot \parallel \cdot$ denotes a bit concatenation and $S(\cdot)$ is here a bijection function consisting of two parallel Sboxes (and this function also meets Assumption 1 as this function essentially consists of two parallel Sboxes corresponding to Equation (8)). Thus, the collision DL-SCA would be applicable to byte-serial hardware implementations *in principle* from the perspective of the selection function.

However, it is still very challenging *in practice* to mount the collision DL-SCA to byte-serial hardware implementations. The difficulty is posed by the fact that the leakage of other bytes (out of target in estimating $\delta[j]$) would always exist in the hardware and may contribute to the side-channel leakage, in contrast to software implementations. Thus, we require a highly accurate points-of-interest selection without leakage related to the computation of profiling bytes. In addition, for especially masked byte-serial hardware, the Sbox circuit is frequently pipelined (for the functional decomposition for an efficient TI realization [44]), which indicates several Sboxes are processed in parallel during multiple clock cycles. The effect of the aforementioned facts on the collision DL-SCA is unclear. Thus, although it would be feasible in principle, the practical feasibility

[10] Some architectures do not necessarily process them in the ascending/descending order (*e.g.*, [40]).

evaluation with state-of-the-art points-of-interest selection methods is important for future work.

Round-based hardware implementation. We first discuss the applicability to a round-based hardware implementation in terms of the selection function. Typical SCAs on round-based AES hardware focus on the register transition between the ciphertext and final-round input. In this case, the selection function is given by

$$\phi(K[j], C[j], C[j']) = \mathrm{Sbox}^{-1}(C[j] \oplus K[j]) \oplus C[j'],$$

where $C[j]$ denotes the j-th byte partial ciphertext, and j' is determinied by j and ShiftRows (for $j = 1$, 5, 9, and 13, $j = j'$). As well as the byte-serial implementation, we consider its 16-bit redundant form as

$$\alpha_{K[j]\|0}(C[j] \parallel C[j']) = \mathrm{Sbox}^{-1}(C[j] \oplus K[j]) \parallel C[j'] \oplus 0,$$

regarding 0 as a fixed eight-bit virtual key. On the one hand, this selection function ϕ has a bijective redundant form α (here, T is given as C) if $j \neq j'$ (and the selection function, in this case, meets Assumption 1). On the other hand, α is not bijection for a fixed $K[j]$, if $j = j'$ (it was proven in [26] that the KIC (condition related to Assumption 1) is not met for bytes with $j = j'$ (*i.e.*, $j = 1$, 5, 9, and 13)). The collision DL-SCA is not applicable to this case. In fact, the distribution of $\phi'(K[j], C[j], C[j'])$ changes depending on the value of $K[j]$, which indicates that we cannot calculate the NLL for δ using the probability distribution $p_{C|\boldsymbol{X},K}(C \mid \boldsymbol{X}, k^*)$. Thus, in the sense of the selection function, the collision DL-SCA is applicable to 12 bytes of round-based AES hardware implementation.

However, from the perspective of side-channel leakage separation, the collision DL-SCA is inapplicable to the round-based AES hardware implementation even for bytes meeting the aforementioned conditions, because round-based hardware computes 16 bytes simultaneously in parallel, and their leakages are not separable into $\boldsymbol{X}[1], \boldsymbol{X}[2], \ldots, \boldsymbol{X}[16]$ in the time domain. Extension of the collision DL-SCA to such an implementation remains a future work.

5 Improvement of collision DL-SCA

5.1 Overview

We present profiling and full-key recovery methods to improve the performance of collision DL-SCAs, which correspond to the NN training and inference, respectively. For the profiling, we propose a novel dedicated NN architecture to utilize side-channel traces of all bytes for training the NN. Thereafter, we propose a method to increase the full-key recovery success rate by effectively using the 16 classifiers obtained by our training method.

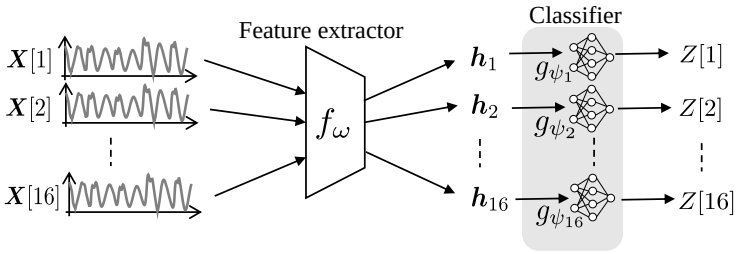

Fig. 2: Our NN architecture.

5.2 Dedicated NN architecture

In Section 4.4, we validated the importance of imitating the true conditional probability distribution as precisely as possible to achieve an optimal attack. The preciseness of the imitation heavily depends on the number of traces available for the NN training. However, in the non-profiled scenario of collision DL-SCAs, the number of training traces is equivalent to the number of attack traces. Thus, the number of training traces is highly limited in the collision DL-SCA (compared to the profiled DL-SCA), which indicates that it may be a bottleneck in achieving a higher success rate. To address this problem, we present an improved collision DL-SCA using a dedicated NN architecture that can utilize 16 times more traces for training the feature extraction part than the conventional collision DL-SCA strategy.

Our basic idea is to unify the feature extraction parts of each NN $q(T \mid \boldsymbol{X}; \theta_j)$ and to train the unified part using all side-channel traces from 16 bytes. Figure 2 illustrates an overview of the proposed NN architecture, which consists of a feature extractor f_ω to extract the information on intermediate values from side-channel traces and classifiers g_{ψ_j} to estimate the j-th byte plaintext from the extracted features. Here, ω and $(\psi_j)_{j=1}^{16}$ are the model parameters of the feature extractor and classifiers, respectively. The number of classifiers in our model should equal that of all Sboxes. Let $\mathcal{S} = \{(\boldsymbol{X}_l[j], T_l[j]) \mid 1 \leq l \leq m, 1 \leq j \leq 16\}$ be a training dataset, where $\boldsymbol{X}_l[j]$ and $T_l[j]$ denote the l-th side-channel trace and plaintext/ciphertext of j-th byte, respectively. Given an l-th trace $(\boldsymbol{X}_l[j])_{j=1}^{16}$, the proposed model predicts $\hat{T}_j$ by the following procedures.

1. For every $j \in \{1, 2, \ldots, 16\}$, we input the trace $\boldsymbol{X}_l[j]$ to the feature extractor f_ω and obtain the latent vector $\boldsymbol{H}_j = f_\omega(\boldsymbol{X}_l[j])$.
2. We obtain the estimated value $\hat{T}_j$ by inputting the extracted feature $\boldsymbol{H}_j$ to the classifier g_{ψ_j}. The estimation result is obtained as the probability distribution of $\hat{T}_j$ because the softmax function is used for the activation function of the last layer of the classifier $g_{\psi_j}(\boldsymbol{H}_j)$.

Thus, the proposed NN architecture shares the same feature extractor for inference of the plaintext/ciphertext information of all bytes. This allows feature

Algorithm 1 Proposed DL-SCA on AES with multiple neural distinguishers

Input: Side-channel traces $\boldsymbol{x}^m = (\boldsymbol{x}[1], \boldsymbol{x}[2], \ldots, \boldsymbol{x}[16])^m$ and plaintexts $t^m = (t[1], t[2], \ldots, t[16])^m$
Output: Estimated correct key $\hat{\boldsymbol{k}} = (\hat{k}[1], \hat{k}[2], \ldots, \hat{k}[16])$

$(\hat{\theta}_j)_{j=1}^{16} \leftarrow \mathrm{TrainNN}(\boldsymbol{x}^m, t^m)$;
for $j = 1$ to 16 **do**
  for each $j' \neq j$ **do**
    $\hat{\delta}[j, j'] \leftarrow \arg\max_{\delta} \sum_i \log q(t_i[j'] \oplus \delta \mid \boldsymbol{x}_i[j']; \hat{\theta}_j)$;
  end for
end for
for $j = 1$ to 16 **do**
  for each $k^* \in \mathcal{K} = \{0,1\}^8$ **do**
    $\hat{\boldsymbol{k}} \leftarrow (\hat{\delta}[j,1] \oplus k^*, \hat{\delta}[j,2] \oplus k^*, \ldots, \hat{\delta}[j,16] \oplus k^*)$; ▷ Let $\hat{\delta}[j, j'] = 0$ for $j = j'$
    if $\mathrm{Verify}(\hat{\boldsymbol{k}}) = \mathrm{true}$ **then**
      return $\hat{\boldsymbol{k}}$;
    end if
  end for
end for
return $\perp$;

extractor training to be performed using 16 times more traces in our training method than the conventional collision DL-SCA strategy. To train our model, we use the NLL loss function defined by

$$L(\hat{\theta}, (\hat{\psi}_j)_{j=1}^{16}) = -\frac{1}{m} \sum_l \sum_j \log \Pr\left(g_{\hat{\psi}_j}(f_{\hat{\omega}}(\boldsymbol{X}_l[j])) = Z_l[j]\right), \tag{10}$$

where $\Pr(g_{\hat{\psi}_j}(f_{\hat{\omega}}(\boldsymbol{X}_l[j])) = Z_l[j])$ is the estimated probability (*i.e.*, confidence) of $T_l[j]$ given the trace $\boldsymbol{X}_l[j]$. In the training of our model, we obtain the trained model parameters $\hat{\omega}$ and $(\hat{\psi})_{j=1}^{16}$ by minimizing Equation (10). In the following, let $q(T \mid \boldsymbol{X}; \hat{\theta}_j)$ be the probability distribution modeled by our NN with the trained parameters $\hat{\theta}_j = (\hat{\omega}, \hat{\psi}_j)$.

5.3 Algorithmic description of our collision DL-SCA with improved full-key recovery strategy

We need a full-key recovery of the target cipher to break a cryptosystem. This indicates that we need a simultaneous success of the partial key recovery of 16 bytes to break the system, which is more challenging than recovering a partial key. For example, CPA is a deterministic algorithm that estimates the secret key from samples of side-channel traces and plaintext (or ciphertext). In an unlucky case where many side-channel traces for a byte contain much noise, the CPA requires more traces for correct key recovery. The full-key recovery by CPA requires the number of traces to be as many as such an unlucky case among 16 CPA trials for a full-key recovery.

This is the same for the proposed DL-SCA. However, we can use an arbitrary byte j for the leakage profiling to model $p_{Z|\boldsymbol{X},K}(Z \mid \boldsymbol{X}, k[j])$, although we used the first byte for the leakage profiling as $p_{Z|\boldsymbol{X},K}(Z \mid \boldsymbol{X}, k[1])$ for the explanation in Section 4.6. In other words, the attacker can perform the NN training using the j-th byte side-channel trace for each j to mount the proposed DL-SCA 16 times (in the case of AES). Here, it is sufficient for the attacker to succeed in only one full-key recovery among 16 repetitions of the proposed DL-SCA. Such multiple trials would mitigate the negative effects of unlucky cases, due to the stochastic aspects of NN training and sampling (*i.e.*, measurements of side-channel traces, including noise). Thus, we can improve the full-key recovery success rate using multiple neural distinguishers, where we select the best byte for the profiling among 16 bytes.

Algorithm 1 illustrates the proposed DL-SCA on AES with 16 neural distinguishers. In Line 1, "TrainNN" is the NN training to obtain the trained NN parameters $\hat{\theta}_j$ for all bytes. This requires trainings of one feature extractor and 16 classifiers as aforementioned, which is sufficiently feasible and requires far less cost than DDLA. In Line 4, we estimate the partial key difference $\delta[j, j']$ for each j and j' ($j \neq j'$) using the proposed distinguisher with trained NN. Here, $\hat{\delta}[j, j']$ denotes the partial key difference between the j-th and j'-th bytes, estimated using the NN with $\hat{\theta}_j$. Note that $\hat{\delta}[j, j'] = 0$ for $j = j'$ trivially holds. Although $\delta[j, j'] = \delta[j', j]$ must hold for true values, it does not necessarily hold that $\hat{\delta}[j, j'] = \hat{\delta}[j', j]$ for their estimations due to the stochastic aspects of NN training and sampling. After the estimation, we perform a brute-force on the estimated key differences and the profiled key value to identify the correct full key. Namely, for $(\hat{\delta}[j, 1], \hat{\delta}[j, 2], \ldots, \hat{\delta}[j, 16])$ which are estimated using $\hat{\theta}_j$, we guess the j-th byte partial key $k[j]$ to obtain an estimated full key $\hat{\boldsymbol{k}}$. These calculations are completed with a complexity of $2^{12} = 2^8 \times 16$, which is sufficiently feasible. In Line 10, we verify the estimated full key in some way (*e.g.*, encryption using the estimated key) as Verify($\hat{\boldsymbol{k}}$), which returns true or false. If true, Algorithm 1 returns the estimated correct key. If we find no correct key, Algorithm 1 returns $\perp$ as an attack failure.

6 Experimental validation

6.1 Dataset and experimental setup

In this section, we demonstrate the validity and effectiveness of the improved collision DL-SCA through a comprehensive performance evaluation on a dataset created by the authors using the ASCAD AES implementation. ASCAD is a dataset of side-channel traces acquired from a first-order masked AES software implementation on an eight-bit microcontroller, and is one of the most popular public datasets for evaluating (profiled) DL-SCAs. Unfortunately, ASCAD basically consists of side-channel traces for a fixed key. Although there is a variable-key training dataset, it only consists of approximately 800 traces for each key value, which is too few to evaluate collision SCAs. In addition, the ASCAD

fixed-key dataset uses an identical secret key for both training and test datasets, which indicates that we can evaluate only a case of $k^* = k$; a success of such an attack does not necessarily validate the collision SCAs. Thus, it is difficult to evaluate the collision (DL-)SCAs using ASCAD. Although there are some other publicly open datasets for evaluating (DL-)SCA as listed in [46, Table 1], all of them are unsuitable for evaluating collision SCAs due to the same reason.

Therefore, we created a dataset by ourselves to evaluate collision (DL-)SCAs with more attack traces in estimating δ. We compiled the source code of the masked AES software provided by ANSSI[11] that is used in ASCAD [4] and implemented it on an Atmel Xmega128D4-AU eight-bit microcontroller. We acquired its side-channel traces for 16 different keys. The target device was mounted on a ChipWhisperer CW308 UFO baseboard. The ChipWhisperer CW1200 capture box generated the base clock, and the clock frequency was set to 50 MHz. The microcontroller is connected to a Keysight DSOX6004A oscilloscope with a sampling rate of 20 GSamples/s.

6.2 Evaluation targets

For the implementation of our method, we employed an NN architecture based on the one presented in [66] because we experimentally found that it achieved the highest performance for most cases among NN architectures proposed in TCHES (*e.g.*, [68]). Specifically, the last layer of the architecture was used as the classifier, while other layers were used as the feature extractor. Therefore, we used an architecture with 16 copies of the final layer in the architecture by Wouters *et al.*. We set the learning rate, batch size, and the number of epochs to 1e−3, 1,000, and 300, respectively. Among 300 epochs, the model parameters with the best attack performance were chosen using the method proposed by Ito *et al.* in [27].[12] We normalized the side-channel traces using "feature standardization" presented by Wouters *et al.* in [66].

As evaluation/comparison targets, we used the second-order CPA [7], MCC-DPA [41], and the original collision DL-SCA in [54] as major non-profiled (and collision) SCAs. For completeness, Section C.1 describes the higher-order CPA we performed. We also evaluate DDLA [58] and a state-of-the-art DDLA variant in [25]. We used the architecture proposed in [66] for all the comparative analysis. Here, we chose different hyperparameters for each method such that it works well (meaning that we searched for the best hyperparameters of each method experimentally as far as we could). For DDLAs [25, 58], we set the learning rate, batch size, and the number of epochs as 1e−3, 1,000, and 50, respectively. For the original collision DL-SCA, we set the learning rate, batch size, and the number of epochs as 5e−3, 50, and 50, respectively.

[11] `https://github.com/ANSSI-FR/secAES-ATmega8515`

[12] To reduce the computation costs, we used an evaluation method with the validation set, which would result in an increase in the number of traces for attacks. However, such an evaluation method is unnecessary in a real attack if the computational costs can be ignored because the attacker can perform the attacks with learned parameters for all epochs.

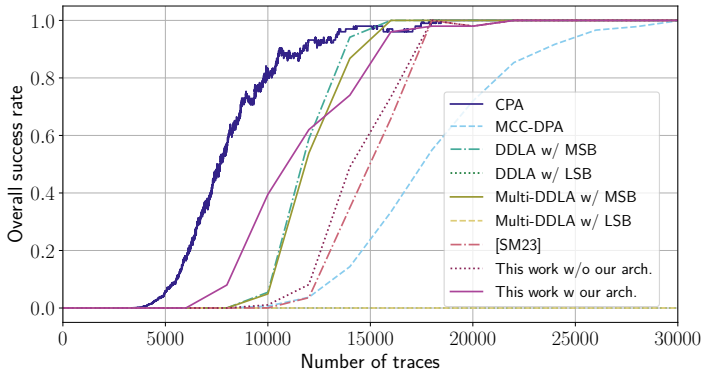

Fig. 3: Evaluation results of non-profiled SCAs.

In contrast, we did not evaluate other types of non-profiled collision SCAs (*e.g.*, [21]) and DL-SCA (*e.g.*, SCAUL [49]), because their extension to masked implementation (including ASCAD) is unknown[13].

6.3 Evaluation result

Figure 3 reports the full-key recovery SRs of the aforementioned non-profiled SCAs, where "Multi-DDLA" is the DDLA variant by Hoang *et al.* in [25]. In Figure 3, "This work w/o our arch." and "This work w/ our arch." mean the results of attack results using our distinguisher shown in Theorem 4 without and with the proposed architecture, respectively. For the collision DL-SCAs, we trained the NN to imitate $p_{Z|\boldsymbol{X},K}(Z \mid \boldsymbol{X}, k[j])$ using the same number of traces as attack traces, because the number of traces required for successful attacks is equivalent to the number of training traces in a real non-profiled scenario. The empirical SRs of CPA, and MCC-DPA were evaluated using 100 trials with the *bootstrap* method (*i.e.*, resampling from samples). We evaluated the attack SR of collision DL-SCAs according to the following procedure. Let j be the byte of the Sbox to be used to train the model, and let j' be another byte of the Sbox to be attacked. Let $\delta[j, j']$ be the key difference between these bytes. First, we estimated $\delta[j, j']$ using a model trained on the side-channel traces of the j-th byte. This was performed in the same way as conventional profiled DL-SCAs. This estimation was repeated 100 times in this paper to empirically calculate $\mathrm{SR}_{j,j'}$ for estimating the key difference $\delta[j, j']$. Next, we computed the probability $\mathrm{SR}_j = \prod_{j' \neq j} \mathrm{SR}_{j,j'}$ that we successfully estimate the key differences $\delta[j, j']$ for all $j' \neq j$. In Algorithm 1, the attack succeeds if we successfully estimate the key differences for all the bytes using the model trained for any one byte. Therefore, the overall SR is given by $1 - \prod_j (1 - \mathrm{SR}_j)$. In this study, we calculated the overall

[13] In fact, we applied SCAUL to ASCAD as it is, but we found that it did not work well.

SR 50 times, and averaged them for an accurate estimation of the overall SR. DDLAs were evaluated using 50 trials with 2 K, 4 K, 6 K, ..., and 30 K traces (*i.e.*, with increments of 2 K) due to its high computational cost. For DDLAs, we used the LSB and MSB as the leakage functions to investigate the impact of the leakage function selection on the attack performances. Since the MCC-DPA also requires a heavy computation, it was evaluated with 2 K, 4 K, 6 K, ..., and 30 K traces as well.

First, we compare our collision DL-SCAs with the original one. From Figure 3, we confirm that the performance of the collision DL-SCA with likelihood-based distinguisher in Theorem 4 is slightly better than that of the original collision DL-SCA [54]. This is because our distinguisher is optimal, as proven by Theorem 4, unlike the conventional sum-based distinguisher. In addition, Figure 3 also shows that the collision DL-SCA with our NN architecture outperforms that without our NN architecture, which indicates that the training of the NNs becomes more efficient by using the feature extractor shared by all bytes.

Next, we focus on the results of DDLAs. The performance improvement of the multi-DDLAs over DDLAs is negligibly small in Figure 3, which is not very surprising. Although the multi-DDLA trains 256 key candidates at a time in a single model, it does not provide any new information to the model; thus, it is natural that the performance is not improved. We also confirm that the performance of the DDLAs heavily depends on the leakage function (*i.e.*, LSB or MSB herein). The attacks succeeded when MSB was used as a leakage function, whereas we observed no attack success when LSB was used. This indicates that the adversary must choose the leakage function suitable for the target device. However, choosing the leakage function is not easy because Section 3.5 states there is no way to determine an appropriate leakage function without information on the correct key. Therefore, we confirm the instability of DDLA as a non-profiled SCA. In this aspect, collision DL-SCA is superior to DDLA, as its performance is independent of the choice of leakage function.

Finally, we focus on the non-DL-based methods: MCC-DPA and CPA. The MCC-DPA requires more traces than CPA and improved collision DL-SCA. Note that, in the evaluation of MCC-DPA, we employed an improved full-key recovery in a manner similar to the multiple distinguishers in the improved collision DL-SCA. So far, the MCC-DPA and CEPACA have been evaluated and utilized on hardware implementations (*e.g.*, [5, 38, 40, 41]), but there have been few studies for them on software implementations. Their (in)effectiveness in software implementation would pose an open problem. CPA had the highest performance among the methods in this experiment. However, to perform CPA, it is necessary to know in advance the countermeasures (*e.g.*, random delay and masking) taken on the target device. In contrast, DDLA and collision DL-SCA learn information about the target device by training a model, allowing them to perform attacks without detailed knowledge of the countermeasures of the device under attack. For example, successful CPA on an implementation with random delay countermeasures requires compensation for misalignment, while DDLA and collision DL-SCA do not. Note that although it has been pointed out

in previous studies of MIA [3, 61] that the appropriate attack method changes depending on the attacker's knowledge, this is the first study to clearly point out that the same is true for DDLA and collision DL-SCA.

We summarize our experimental results as follows:

1. Among various DL-based methods, collision DL-SCA has strict applicability conditions, as discussed in Section 4.5, while DDLA can attack various implementations. DDLA can also achieve high performance if an appropriate leakage function is chosen.
2. DDLA has some limitations discussed in Section 3, and its performance strongly depends on the leakage function, which leads to instability as a non-profiled SCA. Contrarily, collision DL-SCA does not require a choice of leakage function, and thus provides a stable performance.
3. CPA is a classic but still powerful attack. However, it is less flexible than non-profiled DL-SCAs and requires more information about the target device.

7 Concluding remarks

This paper formally analyzes non-profiled DL-SCAs, namely DDLA and collision DL-SCA. Owing to few formal studies, their theoretical validity and practical advantages were unclear. In this study, we first presented a formalization of DDLA, which reveals its relation to MIA. We then presented three theorems, which state some limitations and impossibility results of DDLA. Subsequently, we provided probability-theoretical analyses on a collision DL-SCA. Furthermore, we proposed a distinguisher slightly different from the existing one, and proved its optimality in terms of SR. Our analyses revealed some important conditions for the applicability and success of the attack. Accordingly, we propose an improvement of the collision DL-SCA by means of an optimal distinguished, an NN architecture dedicated to it, and a full-key recovery methodology using multiple neural distinguishers. For validation, we conducted an experimental evaluation of major non-profiled (DL-)SCAs using a newly created dataset using ASCAD implementation, which is sufficient to comprehensively evaluate collision SCAs. As a result, we confirmed the validity of our improved collision DL-SCA, instability of DDLA, and strength of CPA. The experimental result clarified the current situation of non-profiled SCAs.

Theorem 4 implies that the improved collision DL-SCA can achieve a success rate as optimal as the profiled DL-SCA, if we can use the attack traces as many as possible. Theorem 4 motivates the study on DL-SCA for a severe assessment of side-channel resistance in both profiled and non-profiled scenarios, for implementations to which the collision DL-SCA is applicable. Moreover, we will further improve the collision DL-SCA with the help of techniques to leverage DL on a small dataset, such as oversampling, data augmentation, pre-training (usage of pre-trained models), meta/transfer-learning, and fine-tuning.

References

1. Alipour, A., Papadimitriou, A., Beroulle, V., Aerabi, E., Hély, D.: On the performance of non-profiled differential deep learning attacks against an AES encryption algorithm protected using a correlated noise generation based hiding countermeasure. In: Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 614–617 (2020). https://doi.org/10.23919/DATE48585.2020.9116387
2. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.X., Strub, P.Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Advances in Cryptology—Eurocrypt 2017. LNCS, vol. 10210, pp. 535–556 (2017)
3. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.X., Veyrat-Charvillon, N.: Mutual information analysis: a comprehensive study. J. Cryptology **24**, 269–291 (2011). https://doi.org/10.1007/s00145-010-9084-8
4. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. Journal of Cryptographic Engineering **10**(2), 163–188 (2020), https://doi.org/10.1007/s13389-019-00220-8
5. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Trade-offs for threshold implementations illustrated on AES. IEEE Transactions on Computer-Aided Design of Integrated and Systems **34**(7), 1188–1200 (2015)
6. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006)
7. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 16–29. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
8. Bruneau, N., Carlet, C., Guilley, S., Heuser, A., Prouff, E., Rioul, O.: Stochastic collision attack. IEEE Transactions on Information Forensics and Security **12**(9), 2090–2104 (2017). https://doi.org/10.1109/TIFS.2017.2697401
9. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Cryptographic Hardware and Embedded Systems – CHES 2017. Lecture Notes in Computer Science, vol. 10529, pp. 45–68. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_3
10. Carbone, M., Conin, V., Cornélie, M.A., Dassance, F., Dufresne, G., Dumas, C., Prouff, E., Venelli, A.: Deep learning to evaluate secure RSA implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(2), 132–161 (2019)
11. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 13–28. LNCS (2002)
12. de Chérisey, E., Guilley, S., Rioul, O., Piantanida, P.: Best information is most successful. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019, Issue 2**, 49–79 (2019). https://doi.org/10.13154/tches.v2019.i2.49-79, https://tches.iacr.org/index.php/TCHES/article/view/7385
13. Cristiani, V., Lecomte, M., Maurine, P.: Revisiting mutual information analysis: Multidimensionality, neural estimation and optimality proofs. Journal of Cryptology **36**(4), 38 (2023). https://doi.org/10.1007/s00145-023-09476-0, https://doi.org/10.1007/s00145-023-09476-0
14. De Cnudde, T., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., Rijmen, V.: Masking AES with $d+1$ shares in hardware. In: International Conference on Cryptographic

Hardware and Embedded Systems. LNCS, vol. 9813, pp. 194–212. Springer Berlin Heidelberg (2016)
15. De Meyer, L.: Recovering the CTR_DRBG state in 256 traces. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(1), 37–65 (2019), https://tches.iacr.org/index.php/TCHES/article/view/8392
16. Dinu, D., Kizhvatov, I.: EM analysis in the IoT context: Lessons learned from an attack on thread. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(1), 73–97 (2018). https://doi.org/10.13154/tches.v2018.i1.73-97, https://tches.iacr.org/index.php/TCHES/article/view/833
17. Do, N.T., Hoang, V.P., Doan, V.S.: A novel non-profiled side channel attack based on multi-output regression neural network. Journal of Cryptographic Engineering (2023). https://doi.org/10.1007/s13389-023-00314-4
18. Do, N.T., Le, P.C., Hoang, V.P., Doan, V.S., Nguyen, H.G., Pham, C.K.: MO-DLSCA: Deep learning based non-profiled side channel analysis using multi-output neural networks. In: International Conference on Advanced Technologies for Communications (ATC). pp. 245–250 (2022). https://doi.org/10.1109/ATC55345.2022.9943024
19. Dubrova, E., Ngo, K., Gärtner, J.: How we broke a fifth-order masked Kyber implementation by copy-paste. Real World Cryptography (RWC) (2023), https://iacr.org/submit/files/slides/2023/rwc/rwc2023/19/slides.pdf
20. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis: A generic side-channel distinguisher. In: International Conference on Cryptographic Hardware and Embedded Systems. Lecture Notes in Computer Science, vol. 5154 (2008)
21. Glowacz, C., Grosso, V.: Optimal collision side-channel attacks. In: Belaïd, S., Güneysu, T. (eds.) Smart Card Research and Advanced Applications (CARDIS). pp. 126–140. Springer International Publishing, Cham (2020)
22. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
23. Hettwer, B., Horn, T., Gehrer, S., Güneysu, T.: Encoding power traces as images for efficient side-channel analysis. In: 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 46–56 (2020). https://doi.org/10.1109/HOST45689.2020.9300289
24. Heuser, A., Rioul, O., Guilley, S.: Good is not good enough: Deriving optimal distinguishers from communication theory. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 55–74 (2014)
25. Hoang, V.P., Do, N.T., Doan, V.S.: Efficient non-profiled side channel attack using multi-output classification neural network. IEEE Embedded Systems Letters (2022). https://doi.org/10.1109/LES.2022.3213443, early access
26. Ito, A., Ueno, R., Homma, N.: Toward optimal deep-learning based side-channel attacks: Probability concentration inequality loss and its usage. Cryptology ePrint Archive, Report 2021/1216 (2021), https://ia.cr/2021/1216
27. Ito, A., Ueno, R., Homma, N.: Perceived information revisited: New metrics to evaluate success rate of side-channel attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(4) (2022)
28. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO' 99. Lecture Notes in Computer Science, vol. 1666, pp. 388–397 (1999)
29. Könighofer, R.: A fast and cache-timing resistant implementation of the AES. In: Malkin, T. (ed.) Topics in Cryptology—CT-RSA 2008. pp. 187–202. Springer Berlin Heidelberg (2008)

30. Kuroda, K., Fukuda, Y., Yoshida, K., Fujino, T.: Practical aspects on non-profiled deep-learning side-channel attacks against AES software implementation with two types of masking countermeasures including RSM. Journal of Cryptographic Engineering (2023). https://doi.org/10.1007/s13389-023-00312-6
31. Kwon, D., Hong, S., Kim, H.: Optimizing implementations of non-profiled deep learning-based side-channel attacks. IEEE Access **10**, 5957–5967 (2022). https://doi.org/10.1109/ACCESS.2022.3140446
32. Kwon, D., Kim, H., Hong, S.: Improving non-profiled side-channel attacks using autoencoder based preprocessing. Cryptology ePrint Archive, Paper 2020/396 (2020), https://eprint.iacr.org/2020/396, https://eprint.iacr.org/2020/396
33. Kwon, D., Kim, H., Hong, S.: Non-profiled deep learning-based side-channel preprocessing with autoencoders. IEEE Access **9**, 57692–57703 (2021). https://doi.org/10.1109/ACCESS.2021.3072653
34. Lei, Q., Li, C., Qiao, K., Ma, Z., Yang, B.: VGG-based side channel attack on RSA implementation. In: IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 1157–1161 (2020)
35. Lu, X., Zhang, C., Cao, P., Gu, D., Lu, H.: Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(3), 235–274 (2021)
36. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. Security, Privacy, and Applied Cryptography Engineering (SPACE) **10076**, 3–26 (2016). https://doi.org/10.1007/978-3-319-49445-6_1
37. Martinasek, Z., Hajny, J., Malina, L.: Optimization of power analysis using neural network. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications. pp. 94–107. Springer International Publishing, Cham (2014)
38. Moradi, A., Mischke, O.: On the simplicity of converting leakages from multivariate to univariate. In: Bertoni, G., Coron, J.S. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2013. pp. 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
39. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.X. (eds.) International Workshop on Cryptographic Hardware and Embedded Systems. pp. 125–139. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
40. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: A very compact and a threshold implementation of AES. In: Advances in Cryptology—EUROCRYPT 2011. LNCS, vol. 6632, pp. 59–88. Springer (2011)
41. Moradi, A., Standaert, F.X.: Moments-correlating DPA. In: ACM Workshop on Theory of Implementation Security. pp. 5–15 (2016)
42. Ngo, K., Dubrova, E., Guo, Q., Johasson, T.: A side-channel attack on a masked ind-cca secure saber kem implementation. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 676–707 (2021)
43. Ngo, K., Wang, R., Dubrova, E., Paulsrud, N.: Side-channel attacks on lattice-based KEMs are not prevented by higher-order masking. In: International Symposium on Multiple-Valued Logic (ISMVL) (2023)
44. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. Journal of Cryptology **24**(2), 292–321 (2011)
45. Oswald, D., Paar, C.: Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world. In: International Workshop on Cryptographic Hardware and Embedded Systems. LNCS, vol. 6917, pp. 207–222 (2011)

46. Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: SoK: Deep learning-based physical side-channel analysis. ACM Computing Surveys **55**(11), 1–35 (2023)
47. Prouff, E., Rivain, M.: Theoretical and practical aspects of mutual information based side channel analysis. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) Applied Cryptography and Network Security. pp. 499–518. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
48. Ramezanpour, K., Ampadu, P., Diehl, W.: SCARL: Side-channel analysis with reinforcement learning on the ascon authenticated cipher. arXiv:2006.03995 (2020), https://doi.org/10.48550/arXiv.2006.03995
49. Ramezanpour, K., Ampadu, P., Diehl, W.: SCAUL: Power side-channel analysis with unsupervised learning. IEEE Transactions on Computers **69**(11), 1626–1638 (2020). https://doi.org/10.1109/TC.2020.3013196
50. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems (2021)
51. Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R., Rohatgi, P.: Efficient Rijndael encryption implementation with composite field arithmetic. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 171–184. Springer Berlin Heidelberg (2001)
52. Saito, K., Ito, A., Ueno, R., Homma, N.: One truth prevails: A deep-learning based single-trace power analysis on RSA-CRT with windowed exponentiation. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(4), 490–526 (2022)
53. Schwave, P., Stoffelen, K.: All the AES you need on Cortex-M3 and M4. In: Selected Areas in Cryptography—SAC 2016. LNCS, vol. 10532, pp. 180–194 (2016)
54. Staib, M., Moradi, A.: Deep learning side-channel collision attack. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(3), 422–444 (2023)
55. Standeart, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Advances in Cryptology—Eurocrypt 2009. Lecture Notes in Computer Science, vol. 5479, pp. 443–461 (2009)
56. Sugawara, T.: 3-share threshold implementation of AES S-box without fresh randomness. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(1), 123–145 (2019)
57. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N.: Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(3) (2023)
58. Timon, B.: Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(2), 107–131 (2019)
59. Ueno, R., Homma, N., Aoki, T.: Toward more efficient DPA-resistant AES hardware architecture based on threshold implementation. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. Lecture Notes in Computer Science, vol. 10348, pp. 50–64 (2017)
60. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems **1**, 296–322 (2022)
61. Veyrat-Charvillon, N., Standaert, F.X.: Mutual information analysis: How, when and why? In: Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5747, pp. 429–443. Springer

(2009). https://doi.org/10.1007/978-3-642-04138-9_30, https://www.iacr.org/archive/ches2009/57470430/57470430.pdf
62. Vijayakanthi, G., Mohanty, J.P., Swain, A.K., Mahapatra, K.: Differential metric based deep learning methodology for non-profiled side channel analysis. In: IEEE International Symposium on Smart Electronic Systems (iSES). pp. 200–203 (2021). https://doi.org/10.1109/iSES52644.2021.00054
63. Weissbart, L., Chmielewski, Ł., Picek, S., Batina, L.: Systematic side-channel analysis of curve25519 with machine learning. Journal of Hardware and System Security (4), 314–328 (2020)
64. Weissbart, L., Picek, S., Batina, L.: One trace is all it takes: Machine learning-based side-channel attack on EdDSA. In: International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE). pp. 86–105. LNTCS (2019)
65. Won, Y.S., Han, D.G., Jap, D., Bhasin, S., Park, J.Y.: Non-profiled side-channel attack based on deep learning using picture trace. IEEE Access **9**, 22480–22492 (2021). https://doi.org/10.1109/ACCESS.2021.3055833
66. Wouters, L., Arribas, V., Gierlichs, B., Praneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3), 147–168 (2020)
67. Wouters, L., Van den Herrewegen, J., Garcia, F.D., Oswald, D., Gierlichs, B., Preneel, B.: Dismantling DST80-based immobiliser systems. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(2), 99–127 (2020)
68. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(1), 1–36 (2019), https://tches.iacr.org/index.php/TCHES/article/view/8391
69. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Efficiency through diversity in ensemble models applied to side-channel attacks – a case study on public-key algorithms –. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) (3), 60–96 (2021)

A Optimal distinguisher

SCA can be formulated using a distinguisher, which is denoted by a function $d\colon \mathcal{X}^m \times \mathcal{T}^m \to \mathcal{K}$. A distinguisher calculates the rank of each key candidate using a score function from side-channel traces, and estimates the correct key as a candidate with the highest score. For example, correlation power analysis (CPA) uses Pearson's correlation coefficient as the score. DL-SCA described in Section 2.2 uses the NLL. A distinguisher is called optimal if it maximizes the success rate (SR). Studies on optimal distinguishers are crucial from both theoretical and practical perspectives, as a common goal of studies on SCAs is to improve/maximize the SR. The concept of optimal distinguishers was initially presented by Heuser *et al.* in [24]. As the formal definition of the optimal distinguisher is not explicitly described in [24], in this paper, the optimal distinguisher is formally defined as follows:

Definition 2 (Optimal distinguisher). *For attack traces* $\boldsymbol{X}^m = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m)$ *and inputs* $T^m = (T_1, T_2, \ldots, T_m)$*, the success rate of a distinguisher* $d : \mathcal{X}^m \times \mathcal{T}^m \to \mathcal{K}$ *is defined as* $\mathrm{SR}_m(d) = \Pr(K = d(\boldsymbol{X}^m, T^m))$*, where* m *is the number of attack traces. A distinguisher* d_{opt} *is optimal if we have* $\mathrm{SR}_m(d_{\mathrm{opt}}) = \sup_d \mathrm{SR}_m(d)$*.*

According to [24, Theorem 1], an optimal distinguisher is given by

$$d_{\mathrm{opt}}(\boldsymbol{X}^m, T^m) = \arg\max_k \Pr(K = k \mid \boldsymbol{X}^m, T^m).$$

Heuser *et al.* also proved that the optimal distinguisher is given as

$$d_{\mathrm{opt}}(\boldsymbol{X}^m, T^m) = \arg\max_k \sum_i \log p_{\boldsymbol{X}|T,K}(\boldsymbol{X}_i \mid T_i, k), \tag{11}$$

where $p_{\boldsymbol{X}|T,K}$ is the true probability distribution to be estimated in the template attack [11]. Therefore, the optimal distinguisher in Equation (11) validates the template attack. Ito *et al.* further proved in [26,27] that optimal distinguisher is equivalent to

$$d_{\mathrm{opt}}(\boldsymbol{X}^m, T^m) = \arg\max_k \sum_i \log p_{Z|\boldsymbol{X}}(Z_i^{(k)} \mid \boldsymbol{X}_i) = \arg\min_k L^{(k)}, \tag{12}$$

where $L^{(k)}$ is the NLL calculated using $p_{Z|\boldsymbol{X}}(Z^{(k)} \mid \boldsymbol{X})$ for k. Equation (12) validates the DL-SCA using NLL in Equation (3), as $p_{Z|\boldsymbol{X}}$ is the probability distribution to be modeled in DL-SCA.

B Collision SCA

Collision SCA estimates the partial key difference [39]. Let $k[j]$ denote the j-th partial key byte ($j \in \{1, 2, \ldots, 16\}$). The partial key difference between the j-th and j'-th byte is defined as $\delta[j, j'] = k[j] \oplus k[j']$. The collision SCAs estimate

$\delta[j, j']$ instead of $k[j]$, which enables the non-profiling attacker to leverage the power of profiled SCAs. To the best of the authors' knowledge, there are two types of collision SCAs.

Let g be a leakage function, such as the HW. Recall that the AES SubBytes consist of 16 parallel Sboxes. Let $Z[j]^{(k[j])}$ be the j-th byte Sbox output. The first type of collision SCA estimates the partial key difference by finding a difference δ such that $g(Z[j]^{(k[j])}) = g(Z[j']^{(k[j']\oplus\delta[j,j'])})$ [8]. Namely, given a leakage function g, the attacker uses the hypothetical power consumption of $g(Z[j]^{(k[j])})$ and $g(Z[j']^{(k[j'])})$. Although the attacker does not know $k[j]$ and $k[j']$, attacker guesses $\delta[j, j']$ alternatively such that $\mathrm{Sbox}(T[j] \oplus k[j]) = \mathrm{Sbox}(T[j'] \oplus k[j'] \oplus \delta[j, j'])$ from the leakage. In [21], Glowacz and Grosso constructed a collision SCA of this type with an optimality proof under some assumptions on the leakage function. However, the performance of such collision SCAs heavily depends on the quality of the leakage function g, related to $p_{\boldsymbol{X}|Z}$, which indicates that this collision SCA still requires profiling for the leakage function or specific assumption on the leakage in a non-profiling scenario. In addition, its extensions to higher-order attacks still remain as an open problem [21].

Another type is called (tweaked) CEPACA [39], which is followed by MCC-DPA in [41]. In the MCC-DPA, the attacker determines a profiling byte. The MCC-DPA creates a power model using the profiling byte to attack other bytes in a similar manner to CPA. Namely, for the profiling byte, the attacker computes a d-th order centerized moment of a leakage (for univariate analysis) or an expectation of the product of d leakages (for d-variate analysis) for each plaintext value. Then, the attacker estimates the partial key difference between the profiling and target bytes in a manner like CPA using the aforementioned moments/expectations as a power model. In this paper, we consider the MCC-DPA as a major state-of-the-art counterpart of non-profiled higher-order collision SCA, as its advantage over CEPACA has been shown in [41]. This type of collision SCA has been extended to a DL-based variant, namely collision DL-SCA [54], as described and discussed in Section 4. See Section C for the details and formal description of CEPACA and MCC-DPA.

C Conventional higher-order non-profiled SCAs

C.1 Correlation power analysis (CPA)

Consider that, for a univariate analysis (*i.e.*, in attacking hardware implementation), side-channel trace consists of one leakage point, while, for a d-variate analysis (*i.e.*, in attacking masked software implementation like the experiment in this paper), trace consists of d leakage points. If the attacker does not know the point-of-interest, the attacker should perform the following procedure for all possible (d-combination of) sample points. For univariate analysis on a $(d-1)$-st order masked hardware, let L be the leakage at a point-of-interest included in the side-channel trace, and let L_i be its i-th measurement. For d-variate analysis on a $(d-1)$-st order masked implementation, let $L^{(e)}$ (and $L_i^{(e)}$) be the leakage

at the e-th point-of-interest (*i.e.*, leakage of e-th share) as well. The d-th order univariate attecker then converts L_i to $S_i = (L_i - \mathbb{E}L)^d$ for each i, while the d-th order d-variate attacker converts $(L_i^{(1)}, L_i^{(2)}, \ldots, L_i^{(d)})$ to $S_i = \prod_{e=1}^{d}(L_i^{(e)} - \mathbb{E}L^{(e)})$ as well. Let $Z^{(k)} = \phi(T, k)$ be the secret variable for k, where ϕ is the selection function such as $\mathrm{Sbox}(T \oplus k)$. Let g be a leakage function such as the Hamming weight. Then, the attacker estimates k using the empirical Pearson's correlation coefficient $\hat{\rho}$ as

$$\begin{aligned}\hat{k} &= \arg\max_k \left|\hat{\rho}\left(g(Z^{(k)}), S\right)\right| \\ &= \arg\max_k \left| \frac{\sum_i \left(g(z_i^{(k)}) - \mathbb{E}g(Z^{(k)})\right)\left(s_i - \mathbb{E}S\right)}{\sqrt{\sum_{z^{(k)}} \left(g(z^{(k)}) - \mathbb{E}g(Z^{(k)})\right)^2}\sqrt{\sum_i \left(s_i - \mathbb{E}S\right)^2}} \right|.\end{aligned}$$

C.2 Correlation-enhanced power analysis collision attacks (CEPACA)

Suppose that side-channel trace consists of one leakage point for a univariate analysis (*i.e.*, in attacking hardware implementation) while trace consists of d leakage points for a d-variate analysis (*i.e.*, in attacking masked software implementation)[14]. In the CEPACA, the attacker has side-channel leakage for each Sbox computation. For univariate analysis on a $(d-1)$-st order masked hardware, let $L[j]$ be the leakage at a point-of-interest from j-th Sbox included in the side-channel trace, and $L_i[j]$ be its i-th measurement. For d-variate analysis on a $(d-1)$-st order masked implementation, let $L^{(e)}[j]$ (and $L_i^{(e)}[j]$) be the leakage at the e-th point-of-interest (*i.e.*, leakage of e-th share) as well[15]. For each j, the attacker creates 256 sets of side-channel traces according to the value of plaintext. Then, for each set and for each j, the d-th order univariate attacker first computes the d-th order centerized moment of each sample point as $\mathbb{E}\left[(L[j] - \mathbb{E}L[j])^d\right]$; the d-th order d-variate attacker computes the expectation of a product of d leakages as $\mathbb{E}\left[\prod_{e=1}^{d}(L^{(e)}[j] - \mathbb{E}L^{(e)}[j])\right]$. Let $\hat{\mu}_j^d(t)$ be its empirical computation result for the j-th Sbox and for plaintext $t \in \{0,1\}^8$ in a d-th order attack. Let $\delta[j, j']$ denote the difference between j-th and j'-th partial

[14] In practical attacks, an attacker should select the leakage point from side-channel traces. Here, for the ease of explanation, we assume here that the attacker has already identified the point(s)-of-interest.

[15] Herein, we consider that a standard $(d-1)$-st order masked software uses d shares and processes them in a serial manner, although there are some different one such as [2]

keys. The attacker estimates $\delta[j, j']$ using Pearson's correlation coefficient ρ as[16]

$$\begin{aligned}\hat{\delta}[j, j'] &= \arg\max_{\delta} \left|\rho\left(\hat{\mu}_j^d(T \oplus \delta), \hat{\mu}_{j'}^d(T)\right)\right| \\ &= \arg\max_{\delta} \left| \frac{\sum_t \left(\hat{\mu}_j^d(t \oplus \delta) - \mathbb{E}_T\left[\hat{\mu}_j^d(T)\right]\right)\left(\hat{\mu}_{j'}^d(t) - \mathbb{E}_T\left[\hat{\mu}_{j'}^d(T)\right]\right)}{\sqrt{\sum_t \left(\hat{\mu}_j^d(t) - \mathbb{E}_T\left[\hat{\mu}_j^d(T)\right]\right)^2}\sqrt{\sum_t \left(\hat{\mu}_{j'}^d(t) - \mathbb{E}_T\left[\hat{\mu}_{j'}^d(T)\right]\right)^2}} \right|.\end{aligned}$$

C.3 Moments-correlating collision DPA (MCC-DPA)

In the MCC-DPA, the attacker computes $\hat{\mu}_j^d(t)$ for each t using the j-th Sbox leakage, which is used as a kind of power model. The attacker first determines a profiling byte index j, and computes $\hat{\mu}_j^d(t)$ as the d-th order centerized moment for each t and for the determined j in the same manner as CEPACA. For each $j' \neq j$, the d-th order univariate attacker then converts $L_i[j']$ to $S_i[j'] = (L_i[j'] - \mathbb{E}L[j'])^d$ for each i, while the d-th order d-variate attacker converts $(L_i^{(1)}[j'], L_i^{(2)}[j'], \ldots, L_i^{(d)}[j'])$ to $S_i[j'] = \prod_{e=1}^{d}(L_i^{(e)}[j'] - \mathbb{E}L^{(e)}[j'])$ as well. In the attack phase, the MCC-DPA attacker estimates $\delta[j, j']$ using the empirical Pearson's correlation coefficient $\hat{\rho}$ as

$$\begin{aligned}\hat{\delta}[j, j'] &= \arg\max_{\delta} \left|\hat{\rho}\left(\hat{\mu}_j^d(T[j] \oplus \delta), S[j']\right)\right| \\ &= \arg\max_{\delta} \left| \frac{\sum_i \left(\hat{\mu}_j^d(t_i[j] \oplus \delta) - \mathbb{E}_T\left[\hat{\mu}_j^d(T)\right]\right)\left(s_i[j'] - \mathbb{E}S[j']\right)}{\sqrt{\sum_t \left(\hat{\mu}_j^d(t) - \mathbb{E}_T\left[\hat{\mu}_j^d(T)\right]\right)^2}\sqrt{\sum_i \left(s_i[j'] - \mathbb{E}S[j']\right)^2}} \right|.\end{aligned}$$

D Proofs

D.1 Proof of Proposition 1

Let $\mathcal{D}_{k_1} = \{\, \delta_{k_1,k_2} \mid k_2 \in \mathcal{K} \,\}$ be a set of key differences for a partial key k_1. First, we prove $|\mathcal{D}_{k_1}| = |\mathcal{K}|$. From the definition, we obviously have $|\mathcal{D}_{k_1}| \leq |\mathcal{K}|$; thus, it suffices to check whether $|\mathcal{D}_{k_1}| \geq |\mathcal{K}|$ holds. For any different partial keys k_2 and k_3, it does not hold that $\phi_{k_1}^{-1} \circ \phi_{k_2} = \phi_{k_1}^{-1} \circ \phi_{k_3}$. This is because, if $\phi_{k_1}^{-1} \circ \phi_{k_2} = \phi_{k_1}^{-1} \circ \phi_{k_3}$ holds, then we have $\phi_{k_2} = \phi_{k_1} \Leftrightarrow k_2 = k_1$, which is a contradiction. Therefore, there exists an injective function $\mathcal{K} \to \mathcal{D}_{k_1}$, and we have $|\mathcal{D}_{k_1}| \geq |\mathcal{K}|$. Thus, it holds that $|\mathcal{D}_{k_1}| = |\mathcal{K}|$.

Next, we prove that $\mathcal{D}_{k_1} = \mathcal{D}_{k_2}$ holds for any pair of k_1 and k_2. We consider a case where $k_1 \neq k_2$ because it obviously holds in the case of $k_1 = k_2$. First, we prove that $\mathcal{D}_{k_1} \subset \mathcal{D}_{k_2}$. For any key difference $\phi_{k_1}^{-1} \circ \phi_k \in \mathcal{D}_{k_1}$, a partial key k'

[16] In literature, δ is added to secret key k, but not plaintext t. We add δ to t for the ease of explanation, as they are mathematically equivalent in the case of AES.

exists such that $\phi_{k_1}^{-1} \circ \phi_k = \phi_{k_2}^{-1} \circ \phi_{k'} \in \mathcal{D}_{k_2}$ from Assumption 1. Therefore, it holds that $\mathcal{D}_{k_1} \subset \mathcal{D}_{k_2}$. Similarly, we can have $\mathcal{D}_{k_2} \subset \mathcal{D}_{k_1}$, and thus $\mathcal{D}_{k_1} = \mathcal{D}_{k_2}$ holds.

For a fixed key k, we have $\{\, \delta_{k_1,k_2} \mid k_1, k_2 \in \mathcal{K} \,\} = \bigcup_{k_1} \mathcal{D}_{k_1} = \mathcal{D}_k$, which indicates that $|\{\, \delta_{k_1,k_2} \mid k_1, k_2 \in \mathcal{K} \,\}| = |\mathcal{K}|$, as required.

D.2 Proof of Theorem 1

Let k_1 and k_2 be any two different partial keys. We assume that g is a bijective function because g can be regarded as a bijective function by restricting its codomain to its image. We have $Y^{(k_1)} = g(\phi_{k_1}(T))$ and $Y^{(k_2)} = g(\phi_{k_2}(T))$, which is followed by

$$Y^{(k_1)} = g \circ \phi_{k_1} \circ \phi_{k_2}^{-1} \circ g^{-1}(Y^{(k_2)}).$$

Thus, we have $I(Y^{(k_1)}; \boldsymbol{X}) = I(Y^{(k_2)}; \boldsymbol{X})$ because the function $g \circ \phi_{k_1} \circ \phi_{k_2}^{-1} \circ g^{-1}$ is a bijection. Since this holds for any keys k_1 and k_2, the mutual information $I(Y^{(k)}; \boldsymbol{X})$ is a constant independent of the value of k.

D.3 Proof of Theorem 2

First, we prove that, for any leakage function g_1, there exists a function g_1' such that $h_{g_1}(k_1) = h_{g_1'}(k_2)$ holds. Let $g = g_1 \circ \phi_{k_1} \circ \phi_{k_2}^{-1}$. Here, we have $I(g(Z^{(k_2)}); \boldsymbol{X}) = I(g_1 \circ \phi_{k_1} \circ \phi_{k_2}^{-1}(Z^{(k_2)}); \boldsymbol{X}) = I(Y^{(k_1)}; \boldsymbol{X})$. For any $k \neq k_2$, it holds that

$$I(g(Z^{(k)}); \boldsymbol{X}) = I(g_1 \circ \phi_{k_1} \circ \phi_{k_2}^{-1}(Z^{(k)}); \boldsymbol{X}).$$

From Assumption 1, there exists $k_3 \in \mathcal{K}$ such that $\phi_{k_1} \circ \phi_{k_2}^{-1} = \phi_{k_3} \circ \phi_k^{-1}$. Thus, we have $I(g(Z^{(k)}); \boldsymbol{X}) = I(g_1(Z^{(k_3)}); \boldsymbol{X})$. As k_3 takes all key candidate values except for k_1 according to the value of k, we have $\max_{k' \neq k_2} I(g(Z^{(k')}); \boldsymbol{X}) = \max_{k' \neq k_1} I(g_2(Z^{(k')}); \boldsymbol{X})$. From the above, we have

$$\begin{aligned} h_g(k_2) &= I(g(Z^{(k)}); \boldsymbol{X}) - \max_{k' \neq k_2} I(g(Z^{(k')}); \boldsymbol{X}) \\ &= I(g_1(Z^{(k_1)}); \boldsymbol{X}) - \max_{k' \neq k_1} I(g_1(Z^{(k')}); \boldsymbol{X}) \\ &= h_{g_1}(k_1). \end{aligned}$$

Thus, we conclude $h_{g_1}(k_1) = h_{g_1'}(k_2)$ by letting $g_1' = g_1 \circ \phi_{k_1} \circ \phi_{k_2}^{-1}$.

Next, we prove that there exist two functions $g_1 = \arg\max_g h_g(k_1)$ and $g_2 = \arg\max_g h_g(k_2)$ such that $h_{g_1}(k_1) = h_{g_2}(k_2)$. As aforementioned, for g_1, there exists a function g_1' such that $h_{g_1}(k_1) = h_{g_1'}(k_2)$. Similarly, for g_2, there exists a function g_2' such that $h_{g_2}(k_2) = h_{g_2'}(k_1)$. From the definition of g_1 and g_2, we have

$$\begin{cases} h_{g_2'}(k_1) \leq h_{g_1}(k_1) = h_{g_1'}(k_2), \\ h_{g_1'}(k_2) \leq h_{g_2}(k_2) = h_{g_2'}(k_1). \end{cases}$$

Therefore, since $h_{g_2'}(k_1) \leq h_{g_1}(k_1) \leq h_{g_2'}(k_1)$ holds, we conclude that $h_{g_1}(k_1) = h_{g_2'}(k_1) = h_{g_2}(k_2)$, as required.

D.4 Proof of Theorem 3

Let $\operatorname{Im} g := g(\mathcal{Z})$ be the image of the functions g. Because g is an injective function from $\mathcal{Z}$ to $\mathcal{Y}$ from the assumption, the function $g' : \mathcal{Z} \ni z \mapsto g(z) \in \operatorname{Im} g$ whose codomain is restricted in the image $\operatorname{Im} g$ is a bijection function over this image. Therefore, the equality $I(\boldsymbol{X}; Y^{(k^*)}) = I(\boldsymbol{X}; Z^{(k^*)})$ holds.

D.5 Proof of Theorem 4

Fixing any partial key k', a map $k \mapsto \phi_{k'}^{-1} \circ \phi_k = \delta$ is a bijection (that is, k and δ are one-to-one for a fixed k') from Assumption 1. We have a Markov chain $(\Delta, T^m) \to (K, T^m) \to (Z^m, T^m) \to (\boldsymbol{X}^m, T^m) \to \hat{K} \to \hat{\Delta}$, where $\hat{\Delta} = \phi_{k'}^{-1} \circ \phi_{\hat{K}}$ is an estimation of $\Delta = \phi_{k'}^{-1} \circ \phi_K$. Therefore, according to [24, Theorem 1], a distinguisher

$$d(\boldsymbol{X}^m, T^m) = \arg\max_{\delta} \sum_i \log p_{\Delta|\boldsymbol{X}^m, T^m}(\delta \mid \boldsymbol{X}^m, T^m)$$

is optimal in estimating the difference Δ. Let f be a function $k' \mapsto \phi_{k'}$. We then have

$$\begin{aligned}
d(\boldsymbol{X}^m, T^m) &= \arg\max_{\delta} p_{\Delta|\boldsymbol{X}^m, T^m}(\delta \mid \boldsymbol{X}^m, T^m) \\
&= \phi_{k'}^{-1} \circ f \left(\arg\max_{k} p_{\phi_{k'}^{-1} \circ \phi_K | \boldsymbol{X}^m, T^m}(\phi_{k'}^{-1} \circ \phi_k \mid \boldsymbol{X}^m, T^m) \right) \\
&= \phi_{k'}^{-1} \circ f \left(\arg\max_{k} p_{\phi_K | \boldsymbol{X}^m, T^m}(\phi_k \mid \boldsymbol{X}^m, T^m) \right) \\
&= \phi_{k'}^{-1} \circ f \left(\arg\max_{k} p_{K | \boldsymbol{X}^m, T^m}(k \mid \boldsymbol{X}^m, T^m) \right) \\
&= \phi_{k'}^{-1} \circ f \left(\arg\max_{k} \sum_i \log p_{Z|\boldsymbol{X}}(\phi_k(T_i) \mid \boldsymbol{X}_i) \right).
\end{aligned}$$

Here, $(Z, \boldsymbol{X})$ is independent of K if T is not given. Thus, the optimal distinguisher is equivalent to

$$\begin{aligned}
d(\boldsymbol{X}^m, T^m) &= \phi_{k'}^{-1} \circ f \left(\arg\max_{k} \sum_i \log p_{Z|\boldsymbol{X},K}(\phi_k(T_i) \mid \boldsymbol{X}_i, k') \right) \\
&= \phi_{k'}^{-1} \circ f \left(\arg\max_{k} \sum_i \log p_{T|\boldsymbol{X},K}(\phi_{k'}^{-1} \circ \phi_k(T_i) \mid \boldsymbol{X}_i, k') \right) \\
&= \arg\max_{\delta} \sum_i \log p_{T|\boldsymbol{X},K}(\delta(T_i) \mid \boldsymbol{X}_i, k),
\end{aligned}$$

as required.