IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–24. DOI:XXXXXXXX

Weakly Profiling Side-channel Analysis

Lichao Wu[1], Guilherme Perin[2] and Stjepan Picek[3]

[1] Technical University of Darmstadt, Darmstadt, Germany
[2] Leiden University, Leiden, The Netherlands
[3] Radboud University, Nijmegen, The Netherlands

Abstract. Profiling side-channel analysis, recognized for its robust attack performance in worst-case scenarios, necessitates adversaries to have a cloned device for profiling measurements and secret information for data labeling. On the other hand, non-profiling attacks eschew these requirements by trying all key guesses. Although more suitable for real-world attack scenarios, they may suffer from mediocre attack performance due to the lack of leakage insight.

This paper introduces a novel weakly profiling side-channel analysis method that bridges classical profiling and non-profiling analyses. Our method operates within a profiling framework yet discards the necessity for a cloned device, which relies on the fact that there is (commonly) a bijective relationship between known information, such as plaintext and ciphertext, and secret information. This relationship allows an adversary to label leakage measurements using known information and then profile leakages directly on the attacked device. The empirical results show that the proposed approach reaches at least 3× better attack performance with negligible computational effort than existing non-profiling methods. Moreover, it can rival the performance of state-of-the-art profiling attacks.

Keywords: Weakly profiling side-channel analysis · Profiling side-channel analysis · Non-profiling side-channel analysis · Plaintext · Ciphertext.

1 Introduction

Side-channel analysis (SCA) on symmetric-key cryptography is commonly divided into profiling and non-profiling attacks, following different attack strategies. Profiling attacks assume the adversary has complete control of a device clone to be attacked. With that clone, the adversary characterizes the side-channel behavior and then uses this knowledge to reveal secret information from the device under attack. Non-profiling attacks work directly on the target device without this assumption. The adversary obtains secret information leakages and runs a statistical analysis to obtain the best guess. Under a restricted attack assumption, profiling SCA is often considered one of the strongest SCA methods [PPM+23]. Recent years have witnessed tremendous progress with machine (deep) learning-based profiling attacks (DLSCA) [MPP16, ZBHV19, PWP22]. Non-profiling SCA received less attention compared with its counterpart. A recent milestone in non-profiling DLSCA is the approach called Differential Deep Learning Analysis (DDLA) [Tim19]. While the author reported successful attacks on several targets, the attack approach is fundamentally the same as other classical non-profiling attacks such as differential power analysis (DPA) [KJJ99] and correlation power analysis (CPA) [BCO04]. Thus, it inherits their limitations and introduces new challenges, such as low attack performance, restricted leakage functions, and substantial computational complexity. In the last few years, there has been some progress to fix these issues, see, e.g., [DLH+22, HDD22]. However, the reported results are far from state-of-the-art profiling SCA.

Despite these limitations on attack performance, one should not ignore the unique advantage of non-profiling SCA. Its weaker attack assumption makes it more suitable for real-world attacks. For instance, CPA, a classical non-profiling SCA proposed two decades ago, remains the workhorse of practical side-channel evaluations [Ris23, Bri23]. Besides, recent work used CPA to recover the iPhone's hardware fused GID and UID keys, which is used to brute-force the iPhone PIN code offline [Tih22]. Profiling SCA is not feasible for this attack as the underlying hardware is not configurable by a user. Indeed, the availability of a fully controlled cloned device could be met with a white-box evaluation (e.g., in an evaluation lab) under the worst-case attack assumption. However, it would hardly meet a practical black-box attack with a closed device. Non-profiling SCA, although weaker than profiling SCA due to the lack of a profiling device, can be directly applied to the target device (device to be attacked).

The difference in attack assumptions and strategy between the profiling and non-profiling SCA makes them suitable for different attack scenarios. Naturally, an ideal SCA would have the attack capability of a profiling SCA while following the weaker attack assumption of a non-profiling SCA. Our work proposes a novel SCA that fulfills these two requirements. The proposed attack follows profiling SCA's two-step approach, namely profiling and attack. Since it profiles with known information but not key-related intermediate data that is only accessible from a cloned device, we refer to it as *weakly profiling SCA*[1]. In this study, we employ the term *weak* to emphasize the absence of dedicated profiling devices. Weakly profiling SCA leverages the bijective relationship between plaintexts/ciphertexts and the key-related intermediate data. Concretely, an adversary can directly profile the leakage measurements with plaintext/ciphertext on the target device and then leverage the key dependency of the plaintext probability vector output by the profiling model to recover the secret information. Weakly profiling SCA provides superior attack performance compared to classical and recent deep learning-based non-profiling attacks. Besides, it reaches comparable attack performance to the state-of-the-art profiling SCA.

Our main contributions are:

1. We propose a novel weakly profiling SCA approach distinct from classical profiling and non-profiling SCA, allowing an attacker to profile directly on the target device.[2]
2. Our novel approach can be easily used with different profiling approaches, which we showcase with template attack and deep learning (convolutional neural networks).
3. Through comprehensive theoretical and experimental analysis, we demonstrate the construction of our method and its superior performance compared to classical and state-of-the-art non-profiling SCAs. Besides, a comparison with classical and latest profiling SCAs reveals comparable performance.
4. We provide an experimental evaluation of several relevant factors for our attack: data augmentation, the number of measurements, and the number of training epochs. The results suggest that the proposed technique is robust to diverse settings, and data augmentation is crucial in mounting a powerful weakly profiling SCA.

The rest of this paper is organized as follows. In Section 2, we provide the necessary background information. Section 3 details the threat model, our labeling procedure, and the attack scheme. Section 4 compares the proposed attack with state-of-the-art and then discusses necessary preprocessing techniques. In Section 5, we provide experimental results. In Section 6, we conclude the paper and discuss potential future research directions.

[1] Although sharing a similar name, weakly profiling SCA is fundamentally distinct from weakly supervised learning — a machine learning paradigm in which machine learning models are trained using instances that are only partially annotated or labeled [Zho18].

[2] The source code is available in the Github https://github.com/lichao-wu9/Weakly-profiling-SCA.

2 Preliminaries

This section introduces the notation we follow. Afterward, we provide relevant information about side-channel analysis, leakage model, and evaluation metrics.

2.1 Notation

We use calligraphic letters like $\mathcal{X}$ to denote sets and the corresponding upper-case letters X to denote random variables and random vectors $\mathbf{X}$ over $\mathcal{X}$. The corresponding lower-case letters x and $\mathbf{x}$ denote realizations of X and $\mathbf{X}$, respectively. We use a sans serif font for functions (e.g., f).

The term k represents a key byte candidate that takes its value from the key space $\mathcal{K}$. k^* is the correct key byte or the key byte assumed to be correct by the adversary.[3]

A dataset $\mathbf{T}$ is a collection of traces $\mathbf{t}_i$, with each $\mathbf{t}_i$ associated with a label y_i. A complete set of labels with c classes is denoted by $\mathcal{Y} = \{y^1, y^2, \cdots, y^c\}$. In a dataset $\mathbf{T}$, each trace $\mathbf{t}_i$ is associated with a plaintext/ciphertext $\mathbf{d}_i \in \mathcal{D}$ and a key $\mathbf{k}_i$, or $k_{i,j}$ and $d_{i,j}$ when considering a partial key recovery on byte j. In this work, we consider attacking only a single key byte and, thus, omit the byte vector notation in equations. We divide the dataset $\mathbf{T}$ into the profiling set of size N and the attack set of size Q. $\boldsymbol{\theta}$ denotes the vector of parameters to be learned in a profiling model.

2.2 Side-channel Analysis

2.2.1 Profiling SCA

Profiling SCAs rely on building a profiling model to recover the secret. A profiling model is a predictive model constructed to characterize the relationship between the measurable side-channel information emitted by a cryptographic device and the device's internal state or key-dependent operations. Profiling side-channel attacks run in two phases:

1. **Profiling phase.** The adversely builds a profiling model $\mathsf{f}_{\boldsymbol{\theta}}^{M}$, parameterized by a leakage model M and a set of learning parameters $\boldsymbol{\theta}$, to map the input $\mathbf{t}_i$ to the output y_i on a set of N profiling traces. We use the notions $\mathsf{f}_{\boldsymbol{\theta}}^{M}$ and $\mathsf{f}_{\boldsymbol{\theta}}$ interchangeably.
2. **Attack phase.** The profiling model processes each attack trace $\mathbf{t}_i$, outputting a vector of probabilities representing the probability of the associated leakage value. Based on this vector of probabilities, the adversary decides on the best key candidate, as discussed in Section 2.4.

If a profiling model maps input traces to output data with high confidence, only a few measurements from the device under attack could suffice to retrieve the secret data. Examples of attacks are the template attack [CRR02], stochastic models [SLP05], and machine learning-based attacks [HGM+11, MPP16]. Typically, machine learning-based attacks follow the supervised learning scheme [ZBHV19, WAGP20, PCP20, RWPP21, WPP22b] that trains a machine learning model f to predict labels on previously unseen data. It follows the Empirical Risk Minimization (ERM) framework, where the machine learning model parameters $\boldsymbol{\theta}$ are obtained by solving the optimization problem with the loss function L:

$$\underset{\boldsymbol{\theta}}{\arg\min} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), y_i). \tag{1}$$

The loss function quantifies the difference between the predicted output of the machine learning model and the actual target data. During the training process, the objective is to minimize the value of the loss function. This process is achieved through optimization

[3] The subkey candidates can have any number of bits being guessed. Here, we assume the AES cipher scenario, but the concept is algorithm-independent.

techniques like gradient descent, where the model parameters (like weights in a neural network) are adjusted based on the gradient of the loss function.

2.2.2 Non-profiling SCA

Non-profiling side-channel analysis leverages the correlation between key-related intermediate values and leakage measurements. To perform such attacks, the adversary gathers a collection of traces through a series of encryptions/decryptions of different plaintexts. If the selected intermediate value is sufficiently correlated with leakage measurements, the adversary can use these measurements to verify guesses for a small part of the key. In particular, for each possible value of the relevant part of the key (or the key itself), the adversary will follow a "divide-and-conquer" strategy:

1. **Divide/Partition.** The adversary divides the traces into groups according to the intermediate value predicted by the current key guess.
2. **Attack.** If each group differs noticeably from the others (the definition of 'difference' is based on the attack methods), the current key guess is likely correct.

Non-profiling attacks assume less powerful adversaries with no access to a clone of a device to be attacked. Typical examples of such attacks are Correlation Power Analysis [BCO04], Mutual Information Analysis [GBTP08], Differential Deep Learning Analysis [Tim19], and Multi-output regression DLSCA [DLH+22].

Correlation Power Analysis (CPA) and *Mutual Information Analysis* (MIA) are two classical non-profiling SCA techniques. They are based on the Pearson correlation or mutual information measuring the relationships between the leakage features and hypothetical labels. Using CPA as an example, we calculate the correlation coefficient vector $\mathbf{r}$ by testing all key candidates in $\mathcal{K}$ with Eq. (2).

$$r_k = \frac{\sum_{i=1}^{N}(\mathbf{t}_i^k - \overline{\mathbf{t}^k})(y_i^k - \overline{y^k})}{\sqrt{\sum_{i=1}^{N}(\mathbf{t}_i^k - \overline{\mathbf{t}^k})^2 \sum_{i=1}^{N}(y_i^k - \overline{y^k})^2}},\ k \in \mathcal{K}, \tag{2}$$

where $\mathbf{t}_i^k$ and y_i^k are the leakage trace and the corresponding hypothetical label based on k, respectively. The averaged leakage traces and labels are represented by $\overline{\mathbf{t}^k}$ and $\overline{y^k}$. The most likely key k^* can be obtained by:

$$k^* = \mathsf{arg\,max}\ \mathbf{r}. \tag{3}$$

Differential Deep Learning Analysis (DDLA) can be considered as a deep learning (DL) version of CPA, as they both label the leakage traces based on different key guesses and then "correlate" these hypothetical labels with leakage traces. While the adversary uses Pearson correlation in CPA, DDLA relies on the empirical risk shown in Eq. (1). Using a non-bijective partition function, the adversary first repeats Eq. (1) for each key candidate k to estimate $\boldsymbol{\theta}_k$; y_i in Eq. (1) equals to y_i^k, which is calculated with the current key guess. Then, the most likely key (the key assumed to be correct) k^* can be distinguished by finding the one that generates the least empirical risk measured by the negative log-likelihood (NLL) loss:

$$k^* = \mathop{\mathsf{arg\,min}}_{k} -\frac{1}{N}\sum_{i}^{N}\log(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i)),\ k \in \mathcal{K}, \tag{4}$$

where $\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i)$ represents the conditional probability of y_i^k given an input $\mathbf{t}_i$ and DL model parameters $\boldsymbol{\theta}_k$, denoted as $\mathbf{p}(y_i^k|\mathbf{t}_i;\boldsymbol{\theta}_k)$. Given the same training effort, only $y_i^{k^*}$ can lead to fast convergence of the empirical risk due to its correlation with leakage traces. Compared to CPA, thanks to the employment of deep learning, DDLA can break SCA countermeasures

such as Boolean masking.[4] Meanwhile, it is more robust to noise introduced by, for instance, time jitters and random delay interrupts [Tim19].

Multi-output regression DLSCA (MOR) is introduced to reduce the computation efforts of DDLA. Instead of training 256 DL models (for the AES S-box case) with each model trying to correctly *classify* the hypothetical labels with probabilities, MOR employs the idea of multi-output regression, which aims to *regress* the prediction outputs to the actual label values. Specifically, a DL model is trained by mapping the input leakage traces to the actual values of all possible y_i^k. Like DDLA, the most likely key k^* is revealed by finding the smallest loss measured by mean squared error (MSE).

$$k^* = \arg\min_k \frac{1}{N} \sum_i^N (y_i^k - \mathsf{f}_{\boldsymbol{\theta}}^k(\mathbf{t}_i))^2, \ k \in \mathcal{K}, \tag{5}$$

where $\mathsf{f}_{\boldsymbol{\theta}}^k(\mathbf{t}_i)$ denotes the prediction value of DL model to approximate y_i^k. Compared with DDLA, the computation effort is reduced as the adversary only trains one model. Multiple attacks have been developed following this work, we refer interested readers to [KFYF21, APB+20, KHK22].

2.3 Leakage Models

The leakage model, also known as partition functions, simulates the hypothetical physical leakages to process one byte (as we attack the AES cipher that is byte-oriented). Different leakage models can be adopted in practice, and their results may vary depending on the target device. One simple option is to consider the most significant bit (MSB) or the least significant bit (LSB) of a byte (we use LSB in this work). This leakage model results in two classes. For the HW leakage model, the adversary assumes the leakage is proportional to the intermediate value's Hamming weight. This leakage model results in nine classes for a single intermediate byte for the AES cipher. Another type of leakage model results from the xor between two values. Commonly, the Hamming Weight of this xor is calculated and is referred to as the Hamming Distance. A typical approach for AES is to compute the xor (or HW of the xor, i.e., HD) between the final output and the Sbox input of the last round [KPH+19]. Like the HW leakage model, the HD leakage model results in nine classes for a single intermediate byte for the AES cipher. For the Identity (ID) leakage model, an adversary considers the leakage as an intermediate cipher value. This leakage model results in 256 classes for a single intermediate byte for the AES cipher.

2.4 Attack Performance Evaluation

Since the goal of an adversary is to guess the correct key k^*, the adversary calculates a key guessing vector $\mathbf{g}$, which is a vector representing the likelihood of each key candidate k:

$$\mathbf{g} = \mathsf{sort}\left(\mathcal{L}(k)\right), k \in \mathcal{K}, \tag{6}$$

where $\mathcal{L}(k)$ varies for each attack method. As mentioned in Section 2.2.2, CPA and MIA rely on the correlation coefficient and mutual information, respectively; MOR and DDLA use loss value; profiling attacks are based on likelihood. sort is the function sorting array elements in order of decreasing values of their probabilities. The elements in $\mathbf{g}$ represent the likelihood of the corresponding key candidate being the correct key candidate. g_0 and $g_{|\mathcal{K}|-1}$ are the first (best) and last (worst) element of $\mathbf{g}$, respectively.

In a known-key setting, the key rank is the number of (most likely) keys an adversary needs to brute force until recovering the correct key. Among various key enumeration

[4] With recombination of leakage features, high-order CPA can also break the Boolean masking countermeasure.

techniques [PSG16], one of the most straightforward methods is to try every key given its likelihood after generating a key guessing vector. In this scenario, the key rank is the position of the correct key in $\mathbf{g}$. If an attack method reaches the key rank of zero (meaning that the correct key ranks first), we calculate the required number of attack traces for this key rank.

3 Weakly Profiling Side-channel Analysis

This section presents the core idea of the paper. Instead of directly building the mapping between the input leakages and intermediate data (profiling SCA), we *weakly* learn this relationship via public knowledge, such as plaintext and ciphertext, then analyze the output of the learning model to extract the secret information. This section first discusses the threat model. Afterward, we provide information on plaintext/ciphertext labeling, plaintext distribution, and our attack scheme. Finally, a case study is performed.

3.1 Threat Model

Our threat model is the same as that of non-profiling SCA. An adversary has a device with a target cipher (white-box crypto implementation) and a fixed but unknown key. The adversary could send commands to perform encryption/decryption operations. We assume the adversary can only observe the used plaintext/ciphertext but cannot control their values. To launch attacks, the adversary measures multiple side-channel leakages with an oscilloscope and then analyzes leakage traces with plaintexts and/or ciphertexts. The target leakage measurement should be preprocessed to exclude plaintext/ciphertext leakage; a way to detect and remove them is discussed in Section 4.2.

3.2 From Intermediate Data to Plaintext and Ciphertext

In profiling SCA, the output variables are represented by sensitive operations, such as Sbox input or output of the AES cipher. We denote the model trained with intermediate data as *intermediate data-based model*. Recall that supervised learning learns a mapping between input data $\mathcal{X}$ and labels: $\mathsf{f} : \mathcal{X} \to \mathcal{Y}$.[5] If plaintexts or ciphertexts are also involved in such operations, for simplicity, we specify the profiling objective in Eq. (7) with supervised learning terms based on Eq. (1).

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{l}(k_i, d_i)), \tag{7}$$

where l denotes the labeling function that returns the intermediate value according to a known key candidate k_i and a plaintext/ciphertext d_i. Then, the adversary calculates the probability vector $\mathbf{p}(\mathsf{l}(k, d_i)|\mathbf{t}_i; \boldsymbol{\theta})$ given an attack trace $\mathbf{t}_i$. Since d_i and l are known, k can be easily retrieved by picking the label value with the highest probability [WPP22a, PWP22, ZBHV19].

However, in non-profiling SCA, since the adversary does not know the key being used, the adversary cannot use $\mathsf{l}(k_i, d_i)$ to estimate $\boldsymbol{\theta}$. Fortunately, the key is fixed (we denote it as k) for all leakage traces following our threat model. Then, the label $\mathsf{l}(k, d_i)$ and d_i would satisfy:

$$d_i \longmapsto \mathsf{l}(k, d_i). \tag{8}$$

Eq. 8 shows that a plaintext or ciphertext d_i can uniquely identify the intermediate value $\mathsf{l}(k, d_i)$. However, this bijectivity depends on selecting l. For instance, Sbox output is

[5] In practice, f outputs a probability vector representing the probability of all possible labels in $\mathcal{Y}$.

a common labeling function (and intermediate data) for AES attacks [ZBHV19, WPP22a]. d_i and $\mathsf{Sbox}(d_i \oplus k_i)$ are bijective given a fixed key k_i. If Eq. (8) holds, d_i and $\mathsf{l}(k, d_i)$ can be mapped to each other with mapping functions map parameterized by k.

$$\mathsf{l}(k, d_i) = \mathsf{map}_k(d_i). \tag{9}$$

Since the adversary targets a known cipher, the mapping functions are known or can be easily calculated with the (unknown) correct key k^*. Then, we can rewrite Eq. (7) as:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{map}_{k^*}(d_i)). \tag{10}$$

Indeed, due to the bijectivity between labels $\mathsf{l}(k, d_i)$ and d_i, the estimation of $\mathbf{p}(\mathsf{l}(k, d_i)|\mathbf{t}_i; \boldsymbol{\theta})$ is equivalent to an estimation of $\mathbf{p}(d_i|\mathbf{t}_i; \boldsymbol{\theta})$. From the perspective of model profiling, although map_{k^*} is unknown to the adversary, it is a deterministic function and thus does not influence the optimization of $\boldsymbol{\theta}$. Therefore, we can remove the mapping function in Eq. (10) and train a profiling model with dataset $\mathbf{T}$ labeled with plaintexts/ciphertexts:

$$\boldsymbol{\theta}_d = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), d_i). \tag{11}$$

We denote the profiling model built with plaintext/ciphertext labels as a plaintext (or ciphertext) -based model. For simplicity, we denote it as *plaintext-based model*. Following the threat model, since there is no leakage from d_i in leakage traces, both $\mathsf{f}_{\boldsymbol{\theta}_d}$ and $\mathsf{f}_{\boldsymbol{\theta}}$ should learn on the same leakage features that correspond to the processing of $\mathsf{l}(k, d_i)$. If $\mathsf{f}_{\boldsymbol{\theta}_d}$ and $\mathsf{f}_{\boldsymbol{\theta}}$ generalize equally well on these features, they can be converted to each other with a similar mapping function map'_k, where k equals k^*:

$$\mathsf{f}_{\boldsymbol{\theta}_d}(\cdot) = \mathsf{map}'_k(\mathsf{f}_{\boldsymbol{\theta}}(\cdot)), \tag{12}$$

where map'_k converts the probability of the input $\mathsf{l}(k, d_i)$ to its corresponding output d_i. If the adversary correctly guesses the k as k^*, map'_{k^*} can be easily calculated with the knowledge of cipher implementation.

Eq. (12) represents the core idea of the proposed method, which is also visualized in Figure 1. The goal of an adversary is to find a map'_k that maps $\mathsf{f}_{\boldsymbol{\theta}}$ to $\mathsf{f}_{\boldsymbol{\theta}_d}$. If $\mathsf{f}_{\boldsymbol{\theta}}$ is known, k^* can be brute-forced by trying all possible map'_k and finding the best match to $\mathsf{f}_{\boldsymbol{\theta}_d}$. Although the adversary cannot learn $\mathsf{f}_{\boldsymbol{\theta}}$ in a non-profiling setting due to the lack of key-related intermediate data knowledge, in the next section, we propose an estimation of $\mathsf{map}'_k(\mathsf{f}_{\boldsymbol{\theta}}(\cdot))$ by calculating the plaintext or ciphertext distribution for all possible key candidates. Knowing this, an adversary can find the best key by brute-forcing the key-related distributions.

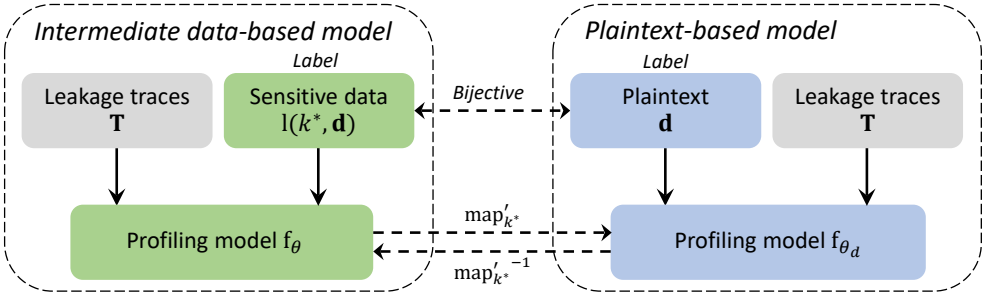


Figure 1: The relationship between intermediate data-based and plaintext-based models.

3.3 Plaintext/Ciphertext Distribution

In profiling SCA, as introduced in Section 2.2, a profiling model $\mathsf{f}_{\boldsymbol{\theta}}$ represents the relation between the input leakage measurement and output intermediate data processed by a leakage model M (discussed in Section 2.3). When inputting an attack trace to $\mathsf{f}_{\boldsymbol{\theta}}$, it outputs a probability vector for all possible intermediate data:

$$\mathbf{Pr}_i(y) = \mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \tag{13}$$

where $y = \mathsf{M}(\mathsf{I}(k, d_i)),\ k \in \mathcal{K}$. An adversary aims for the highest probability of the correct intermediate data y^*, while the rest of the values' probabilities are commonly considered non-critical (e.g., NLL loss). However, we argue that these probabilities are far from arbitrary. If a profiling model can map the input traces to the output labels with high confidence, the prediction probability of the incorrect value is *closely linked* to y^*.

Following a common side-channel assumption [BCO04], let us consider a leaking device with the physical leakage $\mathbf{t_i}$ represented by the real (unknown) leakage function ψ and some additive noise $Z \sim \mathcal{N}(0, \sigma^2)$.

$$\mathbf{t}_i = \psi(y_i) + Z,\ y_i \in \mathcal{Y}. \tag{14}$$

Then, the conditional probability of a label y_j being selected given a correct label y^* can be represented by a probability density function:

$$\mathbf{p}(y_i|y^*) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{a\left\|\psi(y^*) - \psi(y_i)\right\|}{\sigma^2}\right)},\ y_i \in \mathcal{Y}, \tag{15}$$

where $\|\cdot\|$ represents the squared Euclidean distance[6] between two variables, we denote it as label distance; a denotes the linear correlation coefficient. A closer distance between $\psi(y_i)$ and $\psi(y^*)$ indicates that their physical leakage observations are more likely to be similar. When y_i equals y^*, $\mathbf{p}(y_i|y^*)$ reaches maximum. In Eq. (15), since ψ is unknown, an adversary estimates the actual value of $\psi(y_i)$ and $\psi(y^*)$ with leakage assumption M: $\mathsf{M}(y_i)$ and $\mathsf{M}(y^*)$. This paper considers the commonly-used Hamming weight (HW) and Identity (ID) leakage models. If the leakage model is ID, $\mathsf{M}(y_i) = y_i$. Since y_i is parameterized by d_i and k, Eq. (15) can be easily updated to calculate a probability vector $\mathbf{p}(d_i|d^*)$ given a correct key k^*:

$$\mathbf{p}(d_i|d^*) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{a\left\|\mathsf{M}(\mathsf{I}(k^*, d^*)) - \mathsf{M}(\mathsf{I}(k^*, d_i))\right\|}{\sigma^2}\right)}, d_i \in \mathcal{D}. \tag{16}$$

In a non-profiling context, an adversary cannot estimate σ as k^* is unknown. Here, we assume it is a constant as it only depends on the leakage measurements. Now, since the label distance is the only variable of $\mathbf{p}(d_i|d^*)$, we simplify Eq. (16) to Eq. (17) by only keeping the label distance and the negative sign:

$$\mathsf{PD}_k^{\mathsf{M}}(d^*) = -\left\|\mathsf{M}(\mathsf{I}(k, d^*)) - \mathsf{M}(\mathsf{I}(k, d_i))\right\|,\ d_i \in \mathcal{D},\ k \in \mathcal{K}. \tag{17}$$

We denote the output vector of Eq. (17) as *plaintext distribution* (could also be called *ciphertext distribution*, depending on the intermediate data), representing the likelihood of $d_i \in \mathcal{D}$ being selected given a known d^*. Note that when it is clear from the context, we use the notations $\mathsf{PD}_k^{\mathsf{M}}(\cdot)$ and PD_k interchangeably. Intuitively, given a key guess, the plaintext distribution (prior) estimates the physical leakage likelihood of key-dependent data with different plaintexts. If the key guess is correct, a high physical leakage likelihood is expected with the same key-dependent data. Let us consider a plaintext-based model $\mathsf{f}_{\boldsymbol{\theta}_d}$.

[6] There are multiple methods to measure the label distance. Based on [WWK+23], squared Euclidean distance offers the best attack performance.

When classification is accurate, the highest probability is allocated to the corresponding known plaintext byte value d^*. Although this information remains inconsequential to an adversary, the subsequent highest probability typically corresponds to the intermediate byte value that produces physical leakages akin to the known plaintext. An adversary can exploit this secondary information to guess the secret key, a process detailed in the next section.

3.4 Attack Scheme

The plaintext-based model $f_{\boldsymbol{\theta}_d}(\mathbf{T})$ outputs the probability vector of all possible plaintext, including the correct one. Consider an ideal case where the leakage is only associated with the intermediate data; the conditional probability $\mathbf{p}(d_i|d^*)$ would be Eq. (17). In other words, even in an ideal case, $\mathbf{p}(d_i|d^*),\ d \neq d^*$ would be a non-zero value. Given a plaintext-based model $\mathsf{f}_{\boldsymbol{\theta}_d}$, the output probability vector $\mathbf{Pr}(d_i)$ and PD_k with k equal to k^* would have a higher correlation value compared to other key candidates. To retrieve the correct key k^*, the adversary calculates PD_k with all possible keys; the one leading to the highest correlation with $\mathbf{Pr}(d_i)$ would be the most likely key (and assumed to be the correct key):

$$k^* = \arg\max_k \mathsf{corr}(\mathsf{PD}_k^{\mathsf{M}}(\mathbf{d}),\ \mathsf{f}_{\boldsymbol{\theta}_d}(\mathbf{T})),\ k \in \mathcal{K}, \tag{18}$$

where corr represents the Spearman correlation [HK11] that evaluates the monotonic relationship with two inputs. The Spearman correlation offers more numerical stability than the Pearson correlation, as it has a high tolerance when the labels and leakage features are not linearly correlated. Since the adversary knows the plaintext $\mathbf{d}$ for all side-channel measurements $\mathbf{T}$, the brute force effort equals the key space size. For instance, if attacking a subkey (a single byte), the adversary must calculate PD_k 256 times to find the correct key.

Eq. (18) represents the basis of the proposed attack method. Several practical considerations must be made when performing actual attacks. First, similar to other SCA methods, the preprocessing of leakage measurements is mandatory for an efficient attack [WAGP20]. Besides normalizing the data, data augmentation [SK19], a technique to increase the diversity and quantity of training data by modifying the existing data, is the key part that makes the proposed attack successful. Indeed, data augmentation, as a regularization technique, can prevent the profiling model from focusing on specific features and better focus on global features. In the SCA context, since the data leakages only exist in a few features, such techniques can prevent the model from overfitting on non-relevant features. We realize it by randomly shifting the leakage measurement with an augmentation threshold γ. An investigation on data augmentation is presented in Section 5.4.1. An alternative could be to measure and train with more leakage traces. In the same section, we tune the number of training traces and observe its effect on the attack performance.

Second, our attack methodology utilizes plaintext/ciphertext pairs as labels during the training phase. This approach is equivalent to learning the Identity (ID) leakage of key-related intermediate data. However, selecting an appropriate leakage model for labeling the intermediate data is crucial. This ensures that the PD_k can accurately estimate the probability vector $\mathbf{Pr}(d_i)$ generated by a plaintext-based model. For example, in cases where an implementation is prone to leaking Hamming Weight (HW), intermediate data with identical HW values will produce similar physical leakages. A classifier designed to profile based on ID will differentiate between intermediate data only when their HW values vary. Consequently, intermediate data with the same HW values will yield analogous probability estimates, as indicated by PD_k, when using a leakage model M set to HW. In Section 5, multiple leakage models are tested, and one can observe significant differences in attack performance. Considering the above discussions, we formulate the proposed attack scheme in Algorithm 1.

Algorithm 1 Weakly profiling side-channel analysis.

Input: traces $\mathbf{T}$, plain/ciphertext bytes $\mathbf{d}$, leakage model M, augmentation threshold γ
Output: most-likely key k^*

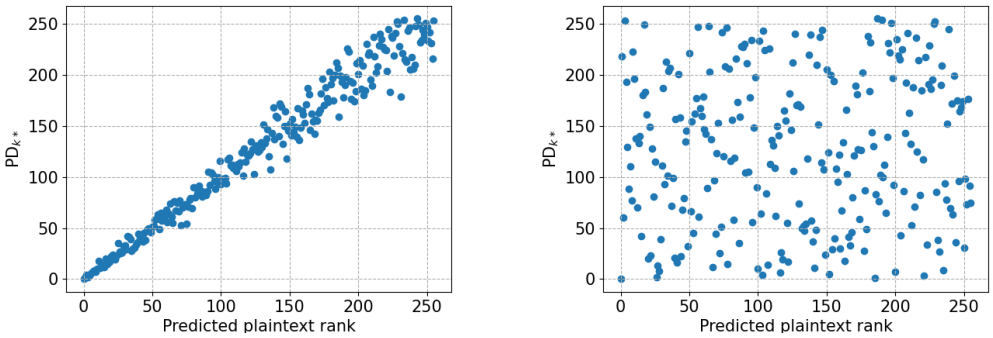
1: $\mathsf{f}_{\boldsymbol{\theta}_d} = \mathsf{train}(\mathbf{T}, \mathbf{d}, \gamma)$
2: $\mathbf{Pr} = \mathsf{f}_{\boldsymbol{\theta}_d}(\mathbf{T})$
3: **for** k **in** $\mathcal{K}$ **do**
4: $\quad \mathbf{corr}_k = \mathsf{corr}(\mathbf{Pr}, \mathsf{PD}_k^{\mathsf{M}}(\mathbf{d}))$
5: **end for**
6: $k^* = \arg\max \mathbf{corr}$

3.5 Case Study

To demonstrate the effectiveness of the weakly profiling SCA, we present results when attacking a simulated dataset comprising leakages from Sbox outputs. Each trace denoted as $\mathbf{t}_i$ has 32 features, with two features containing leakage information at *fixed* locations.

$$\mathbf{t}_i[j] = \mathsf{Sbox}(d_i \oplus k^*), \; d_i \in \mathcal{D}, \tag{19}$$

where d_i and k^* denote random plaintexts and a fixed key, respectively; j denotes the index of the leaking feature. All 32 features are infused with Gaussian noise with a mean of zero and a variance of 0.01. A total of 30 000 traces were simulated for each dataset. Weakly profiling SCA is compatible with all types of classifiers. We use the Gaussian template due to its simplicity and interpretability in this case study.



(a) With correct key assumption. (b) With wrong key assumption.

Figure 2: Predicted plaintext rank vs. PD_k with different key assumptions.

The comparison between the predicted plaintext rank provided by a plaintext-based model and the estimate produced by PD_k^{ID}, using different key assumptions, is given in Figure 2. When the correct key is used to compute PD_k, the plaintext rank aligns perfectly with the predicted key rank. On the contrary, no correlation can be seen when an incorrect key is used. The minor divergence between the predicted plaintext probability vs. PD_{k*} can be attributed to the noise present in the traces. This deviation is particularly noticeable in plaintexts with a high rank (low probability to be selected, see top-right of the figure), implying they are less likely to be chosen. If the number of attack traces would be further increased, one could anticipate a further reduction in this difference. When the attack traces were increased tenfold from 30 000 to 300 000, the correlation between the predicted plaintext rank and PD_{k*} increased from 0.985 to 0.999, signifying a closer match.

4 Discussion

This section initially delves into state-of-the-art attacks, providing a comparative analysis with weakly profiling SCA. Subsequently, we propose leakage preprocessing techniques tailored to various attack settings, with the primary objective of mitigating plaintext/ciphertext leakages.

4.1 Comparison with the State-of-the-art

Figure 3 demonstrates the attack procedure employed in non-profiling SCA, weakly profiling SCA, and profiling SCA. Positioned between non-profiling and profiling SCAs, the weakly profiling SCA integrates an intermediary layer akin to that of profiling SCAs. This layer is represented by a classifier, f_{θ_d}, that processes leakage traces. Including this classifier bolsters the robustness of weakly profiling SCA against leakage noise and potential countermeasures. On the other hand, a unique characteristic of weakly profiling SCA distinguishes it from profiling attacks: it profiles and targets the same trace sets, a feature associated with non-profiling SCA. Benefit from it, weakly profiling SCA immune from portability problem, one of the main challenges of profiling SCA [BCH+20, WWJ+23]. Considering the close connection between the weakly profiling SCA and different types of SCA, we briefly introduce the related and state-of-the-art attacks and discuss the similarities and differences with our work.

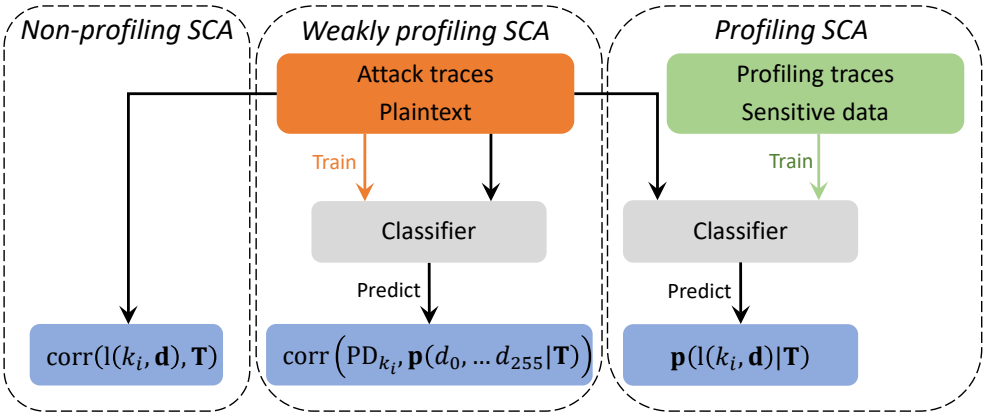


Figure 3: A demonstration of non-profiling SCA, weakly profiling SCA, and profiling SCA.

- **Regression-based attacks**, also known as stochastic attacks, use a parametric model to estimate physical leakage. These were first introduced by Schindler *et al.* in 2005 [SLP05] and later demonstrated in Doget *et al.*'s 2011 study [DPRS11]. Various improvements have been made, such as using stepwise regression [WOS14], lasso regression [WYL+15], ridge regression [WYL+15], to enhance attack robustness, especially in high-noise conditions. However, these attacks face the challenge of learning 256 estimators for each key byte. With millions of traces and a huge number of features within one trace, these methods could become computationally intensive than, e.g., attacks that only need a single estimator. Besides, their adaptability could be challenged when dealing with protected implementation with noisy side-channel leakages. The latest expectation-maximization (EM) attack [BCGR22] requires the point-of-interest (POI) selection, but no POI selection method is given. Knowing that POI selection is challenging in a non-profiling context (except with simple power analysis), its practical application is questionable. Recently proposed EVIL Machine attack addresses the computation complexity issue by reducing the number

of estimations to just one [CLM23a], achieved by combining a neural network with a mutual information neural estimator. Nonetheless, this method's reliance on the simplest leakage model representation for leakage estimation could potentially become problematic in the presence of noisy physical leakages, such as those from hardware crypto engines. Moreover, these attacks might be ineffective against masked implementations without certain techniques (i.e., joint moment estimation) to expose higher-order leakages. On the other hand, the weakly profiling SCA has less constraints: one model is enough to recover the key, leakages are extracted automatically.

Despite these differences, weakly profiling SCA emerges as a versatile SCA methodology compatible with diverse estimators. Specifically, regression models can be applied within the framework of weakly profiling SCA, utilizing plaintext as a label. The inherent bijectivity between plaintext and intermediate data facilitates the regression model's convergence to the physical leakages of intermediate data. While the experimental study of these methods and their adaptability to weakly profiling SCA remains unexplored in the current study, we perceive it as a compelling avenue for future research.

- **Leakage model-based attacks** use models to estimate the physical leakage of targeted data. Prime examples of this approach are CPA, DDLA, and MOR. These attacks are connected to weakly profiling SCA, which also depends on a pre-defined leakage model. Bevan *et al.*'s work notably improved CPA efficiency by leveraging insights from incorrect key guesses [BK02], a concept similar to this study. Weakly profiling SCA stands out by combining the advantages of powerful leakage extraction and limited computation overhead. Another well-known leakage model-based attack is Mutual Information Analysis (MIA) [GBTP08, CLM23b]. Compared to weakly profiling SCA, the key limitation of MIA is that it only works effectively with more straightforward, non-injective leakage models. This often necessitates weakening the model, such as by disregarding certain bits. Weakly profiling SCA does not have such restrictions. Besides, unlike MIA, weakly profiling SCA is compatible with all types of estimators, allowing a better characterization of the leakage data.

- **Non-profiled attacks on asymmetric crypto.** The idea of profiling and attacking on the same device can also applied in attacking asymmetric crypto. For instance, online template attacks can be applied to recover the private key of asymmetric cryptos, such as ECC [BCP+19]. Concretely, with the same device, an attacker obtains template traces with chosen public input and then performs pattern matching on the attack trace to recover each bit separately. Besides focusing on different crypto targets, the main difference with weakly profiling SCA is that the online template attack is based on (segmented) pattern matching (more similar to the side-channel collision attack [SWP03]), while this work directly maps the leakages with labels.

4.2 Leakage Preprocessing

As mentioned in Section 3.4, leakage preprocessing is critical for weakly profiling SCA. In weakly profiling SCA, identifying target cryptographic operations and preprocessing the associated leakages are not the only prerequisites. Recall that the efficacy of weakly profiling SCA is based on the bijective relationship between the plaintext/ciphertext and the key-related intermediate data. If plaintext/ciphertext leakages persist, they can reduce the performance of the plaintext-based model since it is primarily trained using plaintext/ciphertext. This diversion compromises the plaintext-based model's efficacy in drawing a clear link between plaintext and intermediate data. Hence, as emphasized in Section 3.1, a fundamental assumption underpinning the weakly profiling SCA is the

absence of leakages in side-channel traces originating from plaintext/ciphertext. If the assumption does not hold, leakage preprocessing is necessary to guarantee this assumption.

Recall that we assume an adversary knows the plaintexts/ciphertexts (refer to Section 3.1). If the plaintext is masked, we expect no plaintext leakages on the microcontrollers. Simple power/EM analysis (SPA/SEMA) would be required to remove the leakages corresponding to the plaintext loading, ensuring that the plaintext-based model does not combine plaintext shares.. For the more naive implementations, e.g., low-security assurance devices with direct plaintext loading, a simple yet effective solution might be to conduct a preliminary leakage analysis on the plaintext and eliminate or mask the relevant leaking features.

If the crypto algorithm is mask-protected. When performing leakage analysis, the detected leakages would only indicate the plaintext processing (if it is unmasked), as intermediate data, such as Sbox output, would not have first-order leakages. Note that most of the datasets considered in this paper are mask-protected.

If the crypto algorithm is not protected (rarely seen in real world). An adversary would first perform SPA/SEMA to identify the cryptographic operations (e.g., 10 AES rounds for AES-128), then perform leakage segmentation and side-channel attacks only on the identified part.

If SPA/SEMA does not work due to, for instance, hardware crypto implementations or leakage traces being too noisy. Then, since the data loading operations would happen before the actual cryptographic operations, an adversary performs leakage analysis and removes the first detected leakages.

In summary, an adversary would perform plaintext leakage analysis and SPA/SEMA to determine where the plaintext leakage is and where the actual cryptographic operation happens. Then, the adversary can determine how to remove the plaintext leakage based on the scenarios discussed above.

5 Experimental Results

In this section, we benchmark the attack performance of different attack methods. Besides comparing with various non-profiling SCA methods, including Correlation Power Analysis (CPA), Mutual Information Analysis (MIA), Multi-output DLSCA (MOR), and Differential Deep Learning Analysis (DDLA), we also benchmark with template attack and DLSCA to demonstrate the attack capability of weakly profiling SCA. Since non-profiling and profiling attacks follow different attack strategies, the benchmarks are performed with two different settings. In Section 5.4 (hyperparameter evaluation of weakly profiling SCA), only the first setting is considered.

Comparison with non-profiling attacks. The *same* leakage traces are used for training/-partitioning and attack (they are just drawn from the same set). The number of traces for each dataset is detailed in Table 2.

Comparison with profiling attack. The profiling uses a set of leakage traces; the numbers follow Table 2. Then, a *new* set of traces is used for the key recovery (attack); the corresponding data sizes are detailed in Table 1. Note that weakly profiling SCA can attack different data with the same secret key.

The deep learning (DL) model and hyperparameters are identical for all DL-based attack methods, i.e., DDLA, MOR, weakly profiling DLSCA, and profiling DLSCA. We acknowledge that tuning DL models for each dataset and method would potentially lead to better attack performance. However, it also introduces more variables, such as DL

Table 1: The number of attack traces for the profiling setting.

	ASCAD_F	ASCAD_R	CHES_CTF	AES_RD	AES_HD
Trace num.	30 000	30 000	5 000	20 000	20 000

complexity and training effort, making our benchmark extremely complex. Additionally, benchmarking with customized DL could introduce more uncertainty, as one cannot guarantee that it is optimal for a specific setting. Consequently, we employ a convolution neural network (CNN) from [PWP22] due to its excellent performance in various attack settings.[7] The network consists of a convolution block with a convolution layer (kernel number: 4; size: 40; stride: 20), an average pooling layer (size: 2; stride: 2), and a batch normalization layer, followed by two dense layers with 400 neurons and an output layer with 256 neurons. $Selu$ is used for the layer activation except for the last layer that uses $Softmax$; the batch size is 800. Regarding training epochs (the number of iterations that allow the model to learn and adjust its parameters based on the data), the DL model for DDLA has trained for 50 epochs for each key guess [Tim19][8]; the rest of the DL models are trained for 250 epochs. An evaluation of training epochs can be found in Section 5.4.3. Data augmentation is applied to all *DL-based* attack methods for a fair comparison, which is realized by adding a layer, right after the input, that randomly shifts the leakage measurement within a pre-defined augmentation threshold γ equal to 10. A study of data augmentation is given in Section 5.4.1. In terms of template attack, we use Principle Component Analysis [WEG87] to reduce the trace dimension to 20: the best parameter ranging from 5 to 50 with a step of 5.

The rest of the attack settings follow Table 2. To reduce the effect of random factors (e.g., random weight initialization) on the attack performance, each DL-based attack method is executed ten times independently and the averaged results are presented[9]. Indeed, compared with a single attack, the averaged results from multiple attacks better represent the general attack performance of an attack method.

5.1 Datasets

Our experiments consider five datasets. Four are software targets, and one is a hardware target. All software targets are protected - three with masking and one with a hiding countermeasure. The detailed attack settings of these datasets are presented in Table 2. The last columns list the targeted intermediate data. Based on our preliminary correlation analysis, all tested datasets have none or very limited first-order leakage on the key-dependent intermediate data (<0.03) and plaintexts/ciphertexts (<0.04).

Table 2: Summary of the tested datasets.

Dataset	Traces/Samples	Protection	Target intermediate value
ASCAD_F	30 000/1 400	First-order boolean masking	$\mathsf{Sbox}(p_2 \oplus k_2)$
ASCAD_R	30 000/5 000	First-order boolean masking	$\mathsf{Sbox}(p_2 \oplus k_2)$
CHES_CTF	40 000/2 200	First-order boolean masking	$\mathsf{Sbox}(p_0 \oplus k_0)$
AES_RD	30 000/3 500	Random delay	$\mathsf{Sbox}(p_0 \oplus k_0)$
AES_HD	30 000/1 250	None	$\mathsf{Sbox}^{-1}(c_7 \oplus k_7) \oplus c_{11}$

[7]The DL models were implemented in Python version 3.6, using TensorFlow library version 2.6.0. The model training algorithms were run on an Nvidia GTX 1080TI graphics processing unit (GPU), managed by Slurm workload manager version 19.05.4.

[8]We have also tried with 100 epochs but noticed that it performs worse.

[9]Success rate is also a widely used metric to represent the attack performance. However, if both attacks reach a 100% success rate, the attack performance difference is indistinguishable. Therefore, this paper does not employ the success rate to interpret results.

ASCAD_F. The ASCAD datasets contain the measurements from an 8-bit AVR microcontroller running a masked AES-128 implementation [BPS+20].

ASCAD_R. The trace pattern of ASCAD_F is different (more noisy) from ASCAD_F due to different measurement configurations[10].. Besides, ASCAD_R also provides traces with random keys. We only use the leakage traces with a fixed key (thus, the dataset part that is commonly used for testing with DLSCA).

AES_RD. The target smartcard is an 8-bit Atmel AVR microcontroller. The protection uses random delay countermeasures described by Coron and Kizhvatov [CK09]. Adding random delays to the normal operation of a cryptographic algorithm affects the misalignment of important features, making the attack more difficult to conduct.

CHES_CTF. This dataset refers to the CHES Capture-the-flag (CTF) AES-128 measurements released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces consist of masked AES-128 encryption running on a 32-bit STM microcontroller.[11]

AES_HD. This dataset is first introduced in [KPH+19], targeting an unprotected hardware implementation of AES-128 written in VHDL in a round-based architecture. Side-channel traces were measured using a high sensitivity near-field EM probe, placed over a decoupling capacitor on the power line on Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board.[12].

Note that for the AES_HD dataset, we attack the last round of AES with the knowledge of ciphertexts $\mathbf{c}$; for the rest, we attack the first round of AES with plaintexts $\mathbf{d}$. As mentioned, both plaintexts and ciphertexts are valid targets for weakly profiling SCA.

5.2 Attack Performance

In this section, we evaluate the attack performance of different attack methods. The benchmark results are shown in Table 3 and Table 4. The results for different leakage models are separated by '/'. The HD leakage model is only used for the AES_HD dataset; other datasets use the HW leakage model. Besides, since MIA and DDLA do not support a bijective leakage model such as ID, LSB is used as a replacement, following the original papers. Key rank (see Section 2.4) is used to assess the attack performance of each method. As mentioned, all traces (the numbers are detailed in Table 2) are used for attacks. Aligned with other non-profiling SCAs, weakly profiling SCA uses the same traces for training an attack. The key recovery speed represents the attack performance given the same number of attack traces, measured by the required number of attack traces to reach a certain key rank, e.g., zero. If an attack cannot break the target with the given number of attack traces, its performance is measured by its final key rank value (such as KR10, meaning the key rank of the correct key is 10). Our method is benchmarked with non-profiling and profiling attacks. The best results obtained in the non-profiling settings are marked in **bold**. For profiling settings, they are marked in ***bold italic***.

Weakly profiling SCA performs significantly better in all test cases when compared to non-profiling attack methods, as shown in Table 3. For instance, when attacking the AES_RD dataset with the HW leakage model, weakly profiling SCA only requires a single trace to reveal the key, while the second best, DDLA, requires more than 2 500 traces. Although DDLA may perform better by optimizing DL model hyperparameters and training settings, it always suffers from significant training effort (more than four hours

[10] https://github.com/ANSSI-FR/ASCAD/issues/13
[11] https://chesctf.riscure.com/2018/news
[12] http://aisylabdatasets.ewi.tudelft.nl/aes_hd.h5

per leakage model). CPA reaches a low key rank with the AES_RD dataset. However, DL is more resilient than CPA when facing the random delay countermeasure. MOR is considered an improved version of DDLA regarding computation complexity, as an adversary only needs to train one DL model instead of 256 for a subkey byte. However, from the attack performance perspective, MOR improves marginally compared to DDLA.

Table 3: Performance benchmark with non-profiling attacks. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID or LSB (MIA and DDLA only).

Dataset	CPA	MIA	MOR	DDLA	This work
ASCAD_F	KR158/KR46	17 085/3 521	1 957/638	KR7/309	**8/111**
ASCAD_R	KR64/KR7	KR192/4 616	KR28/KR9	27 266/KR48	**20/19**
CHES_CTF	KR138/KR220	KR133/KR241	KR6/KR31	KR54/KR85	**6 121/KR2**
AES_RD	KR11/KR45	KR3/9 320	KR33/3 112	2 541/KR2	**1/57**
AES_HD	KR18/KR144	KR203/KR156	5 593/KR10	KR26/KR20	**60/KR6**

From Table 3, first-order CPA is not working on masked datasets, while a DL model helps recombine physical leakages of mask shares, thus leading to successful key recovery. The performance of MIA aligns with the observation report in [EST+22] that certain second-order moments in the ASCAD dataset can be captured with classical attacks, but more attack traces are required to recover the secret. Note that high-order CPA and MIA might lead to faster key retrieval. However, given our specific threat model, where the adversary is limited to observing only the cipher input and output, a substantial investment of effort would be required to construct second-order features for all leakage features with potential information loss [BGP+11]. For instance, given leakage traces with n features, second-order CPA requires n^2 times correlation analysis. This renders the attack practically infeasible within the constraints of our threat model.

In comparisons with profiling attacks, as presented in Table 4, weakly profiling SCA (more specifically, *weakly profiling DLSCA*), template attack, and profiling DLSCA exhibit impressive results. Profiling DLSCA performs better than the template attack, indicating its strong capability in handling and combining complex leakage features. As mentioned, weakly profiling SCA is now working in the profiling setting: the profiling and attack traces are different. Still, our method surpasses the performance of the profiling DLSCA in seven out of ten test scenarios. One may be surprised that weakly profiling SCA can surpass the effectiveness of traditional profiling SCA despite having weaker attack assumptions. This phenomenon can be attributed to two key factors. First, unlike profiling SCA, which typically concentrates on a single-label hypothesis, weakly profiling SCA engages with probability vectors across all label hypotheses. This approach significantly enhances attack performance in methods like [BK02] and profiling DLSCA [WWK+23]. Second, if the plaintext-based model $\mathsf{f}_{\boldsymbol{\theta}_d}$ demonstrates superior generalization capabilities over intermediate data-based model $\mathsf{f}_{\boldsymbol{\theta}}$ — possibly owing to advanced trace preprocessing or refined hyperparameter optimization — the weakly profiling attack can indeed outperform its profiling counterpart. However, it is critical to acknowledge that, in contrast to profiling SCA, weakly profiling SCA cannot directly learn key-related intermediate data under a weaker attack assumption. Assuming an extensive collection of training traces and optimal adjustments to the profiling model, one would anticipate a more robust attack efficacy from profiling SCA than its weakly profiling counterpart. As an example, the best profiling attack can break some of the test datasets with a single trace [PWP22].

Based on the above results, weakly profiling DLSCA can overcome the protection schemes in the tested datasets with only raw traces as inputs. However, higher-order masking might compromise its efficacy, a limitation shared with other profiling and non-

Table 4: Performance benchmark with Profiling DLSCA. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID.

Dataset	Template attack	Prof. DLSCA	This work
ASCAD_F	300/712	464/147	***8*** / ***44***
ASCAD_R	244/888	458/62	***30*** / ***214***
CHES_CTF	KR22/KR72	1 943/***KR25***	***230***/KR90
AES_RD	KR36/KR96	530/163	***22*** / ***136***
AES_HD	15 195/KR131	7 077/KR21	***3 173*** / ***KR4***

profiling methods. We expect that specific leakage pre-processing techniques, such as leakage recombination or access to design information such as mask shares, may become essential. This need arises particularly when the classifier, such as a DL model, lacks the capability to integrate the physical leakages originating from multiple mask shares.

5.3 Robustness to Desynchronization

Desynchronization, a.k.a. misalignment, is a frequent issue in side-channel traces, disrupting the timing and alignment of key features. This misalignment causes informative points in the trace to be out of place, complicating the process of extracting useful leakage features. This section benchmarks the robustness of different attack methods under the influence of desynchronization. Two desynchronization levels, 50 and 100, are considered to simulate the time-jitter effect, realized by randomly shifting the traces within the desynchronization level. Our preliminary analysis shows that CPA, MIA, and template attacks do not recover the key when attacking the desynchronized traces. Therefore, the corresponding results are discarded from the following tables. On the other hand, DL models, especially convolutional neural networks (CNNs), are more resilient to masking and desynchronization due to their shift-invariance property [KWPP23]. Consequently, DL-based attacks, including weakly profiling DLSCA, would also benefit from this characteristic. The results are shown in Table 5 and Table 6. Again, the best results obtained for the non-profiling and profiling settings are marked in **bold** and ***bold italic***, respectively.

Table 5: Performance benchmark with desynchronization 50. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID or LSB (DDLA only).

Dataset	MOR	DDLA	This work	Prof. DLSCA	This work
ASCAD_F	2 910/KR34	KR152/KR118	**10**/**112**	985/***280***	***14***/531
ASCAD_R	KR12/KR62	KR69/KR156	**54**/**17**	KR9/***223***	***22***/942
CHES_CTF	KR5/KR120	KR124/KR123	**KR4**/**KR116**	KR41/***KR56***	***KR14***/KR135
AES_RD	KR21/3 200	2 517/KR7	**1**/**555**	429/97	***21*** / ***95***
AES_HD	6 098/**KR8**	KR94/KR116	**951**/KR14	***14 127***/KR27	KR2/***KR13***

Our method performs significantly better than its counterparts in non-profiling settings. It performs the best in nine of ten attack scenarios; for the cases where the key rank reaches zero (the correct key ranks the first), it performs at least 6× better than other methods. MOR outperforms our method in one test case. Compared with the attack performance with no noise (Table 3), MOR is slightly influenced by added noise when attacking AES_HD, while weakly profiling SCA experiences a considerable performance reduction. Indeed, when targeting the ciphertext with weakly profiling SCA, the classifier has to find the leakages related to the Sbox^{-1} output. However, AES_HD has limited

leakage on this intermediate data, thus increasing the learning difficulties for our method. MOR, on the other hand, utilizes the Hamming distance labeling directly. The stronger leakage could lead to robust performance.

Moving to the profiling setting, profiling DLSCA achieves a similar attack performance to our method. Out of ten test cases, profiling DLSCA is better in four cases, and weakly profiling DLSCA is better in the rest. Compared with the results in Table 3, profiling DLSCA performs better in more test cases. Indeed, the introduction of time randomness reduces the capability of both $\mathsf{f}_{\boldsymbol{\theta}_d}$ and $\mathsf{f}_{\boldsymbol{\theta}}$, which could potentially reduce the capability difference between these two DL models. Additionally, since our method is based on the correlation between PD_k and $\mathsf{f}_{\boldsymbol{\theta}_d}$, a reduced $\mathsf{f}_{\boldsymbol{\theta}_d}$ would require more leakage traces to compensate for the performance loss. We acknowledge that some attacks/profiling models could perform better than presented results, i.e., attacking different intermediate data [GJS19] or using fine-tuned mode [PWP22]. However, weakly profiling SCA could also benefit from these approaches.

Similarly, our method performs the best in most scenarios in non-profiling settings when increasing the desynchronization level from 50 to 100. As shown in Table 6, with a single DL model, weakly profiling SCA reaches a key rank of zero in seven out of ten cases, and the attack performance is superior to its counterparts. Compared with profiling DLSCA, there are more cases (five) where profiling DLSCA performs better. This observation is aligned with our underlying assumption between profiling DLSCA and our method. It suggests that the potential of profiling DLSCA may be even greater than current results, provided it undergoes further optimization and fine-tuning.

Table 6: Performance benchmark with desynchronization 100. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID or LSB (DDLA only).

Dataset	MOR	DDLA	This work	Prof. DLSCA	This work
ASCAD_F	KR5/KR34	KR153/KR149	**8/3 373**	2 139/1 695	***17/1 408***
ASCAD_R	KR12/KR62	KR68/KR130	**154/867**	KR24/***1 055***	***30***/2 245
CHES_CTF	KR13/KR137	KR125/173	**KR12/KR99**	KR149/***KR59***	***KR9***/KR91
AES_RD	KR54/4 521	4876/KR4	**1/561**	732/***198***	***22***/KR2
AES_HD	6 077/**KR9**	KR131/KR142	**166**/KR10	***10 691/KR8***	19 146/KR16

5.4 Hyperparameter Evaluation

This section explores the influence of various hyperparameters connected with the training process on the weakly profiling SCA's attack performance. Indeed, since we are using a neural network architecture from a related work [PWP22], it remains unclear how factors not connected with the architecture influence the performance of our attack.

5.4.1 Data Augmentation

The processing of leakage measurements is crucial for executing an effective attack. In this study, data augmentation significantly contributes to the success of the proposed attack. Data augmentation is a regularization technique that prevents the profiling model from becoming overly focused on specific features, enabling it to concentrate on global features. In side-channel attacks, where data leakages are found in a limited number of features, such techniques impede the profiling model from overfitting irrelevant features.[13]

[13]Various strategies can also be adopted to counteract overfitting. For instance, early stopping techniques may be utilized, which terminate model training if a tracked metric does not improve after a specified number of epochs.

Table 7: Data augmentation (DA) analysis. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID.

Dataset	DA-0	DA-5	DA-10	DA-20
ASCAD_F	KR10/KR46	8/118	8/111	8/429
ASCAD_R	KR64/KR143	19/13	20/19	2/12
CHES_CTF	8 573/KR74	8 506/KR10	6 121/KR2	11 742/KR53
AES_RD	KR50/KR131	1/22 085	1/57	1/54
AES_HD	KR35/KR66	11 916/KR15	60/KR6	256/KR8

Table 7 shows the augmentation threshold's influence on the attack performance. When setting the data augmentation threshold to zero, weakly profiling SCA could only recover the key in one test case. Then, one can observe a performance boost when introducing random shifts to datasets. The ranges between DA-5 and DA-10 are optimal for most test cases. When the augmentation threshold reaches 20, the attack performance worsens in several settings. We can conclude the necessity of data augmentation for weakly profiling SCA. Recall that one of the fundamental assumptions for weakly profiling SCAis that the side-channel traces do not have plaintext leakage. The random shifting of the leakage traces would reduce the potential plaintext leakages (if any) and help the plaintext-based model better focus on the intermediate data. Still, a too-large data augmentation threshold would reduce the attack performance, as it would increase the difficulties of the DL model fitting the leakage, necessitating longer training and larger DL models.

5.4.2 Dataset Size

DL-based methods are known to be data-hungry [A+18]. In the SCA context, more leakage traces would be helpful to compensate for noise and reveal the underlying distribution of leakage features. When looking at Table 8, as expected, more training traces lead to better attack performance for weakly profiling SCA.

Table 8: Study on the influence of the data size. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID.

Dataset	5 000	10 000	20 000	30 000
ASCAD_F	KR3/KR46	202/KR37	7/848	8/111
ASCAD_R	KR4/KR40	56/KR2	45/60	20/19
CHES_CTF	KR212/KR187	KR72/KR121	KR36/KR114	KR10/KR52
AES_RD	707/KR50	1/KR14	1/2 332	1/57
AES_HD	KR18/KR48	KR11/KR25	4 709/KR4	60/KR6

One could observe that CHES_CTF and AES_HD require more traces than the other datasets. For AES_HD, as discussed before, the possible cause would be the limited leakage on the Sbox^{-1} output. On the other hand, the performance for CHES_CTF could be explained by its leakage type. According to the literature [WPP22b, RWPP21], the CHES_CTF dataset mainly contains HW leakages, while when attacking with the ID leakage model, it is less likely to reveal the key with the same number of attack traces. Since the proposed method is trained with plaintext/ciphertext values, the method's efficiency intrinsically relies on the intermediate data ID leakages (HW of the intermediate data is injective to plaintexts/ciphertext). When a dataset mainly has HW leakages, the plaintext-based model $\mathsf{f}_{\boldsymbol{\theta}_d}$ would struggle in mapping the plaintext to the (HW) leakage features, finally leading to a reduced attack performance.

5.4.3 Training Epochs

The number of training epochs is not inherently tied to enhancing the capability of a DL model to map inputs to outputs. In fact, excessive training epochs can diminish the DL model's generalization capacity when presented with previously unseen data, a phenomenon known as *overfitting*. However, in the context of weakly profiling SCA, since an adversary typically employs the same leakage traces for both profiling and attack phases, concerns regarding deep learning generalization and overfitting become less significant.

Table 9: Study on the influence of the training epoch. The results for different leakage models are separated by '/'. Before '/': HW or HD (AES_HD only); after '/': ID.

Dataset	50	100	150	200
ASCAD_F	34/451	8/237	8/112	8/81
ASCAD_R	KR3/KR6	54/19	18/13	19/14
CHES_CTF	KR22/KR162	KR9/KR160	28 065/KR73	6 855/KR17
AES_RD	1/KR36	1/KR2	1/54	1/64
AES_HD	77/KR6	74/KR6	19/KR5	58/KR4

Table 9 shows the performance variation of weakly profiling SCA when training with different numbers of epochs. Training with 50 epochs is insufficient for most settings; with an extra 100 epochs of training (150), eight out of ten attacks lead to successful key recovery. For the CHES_CTF dataset, one could observe a steady decrease in key rank value, indicating that the DL model is gradually transferring the HW-related feature and learning to connect with plaintext labels. This observation confirms our assumption in Section 5.4.2 about CHES_CTF performance and the limitation of weakly profiling SCA. Simultaneously, within the range of tested epochs, weakly profiling SCA is robust to the overfitting effect.

6 Conclusions and Future Works

This paper introduces a novel weakly profiling SCA leveraging the bijectivity between plaintext/ciphertext and key-related intermediate data. We define the plaintext distribution (PD_k) to approximate the likelihood of each plaintext being selected as the correct plaintext given a key guess k, then use this approximation to correlate with the prediction output of a plaintext-based model trained with plaintext/ciphertext to retrieve the correct key. Thanks to this bijectivity, the weakly profiling SCA is performed in a profiling way, thus bypassing a common prerequisite of non-profiling SCA: non-injectivity of the intermediate data or leakage model. Our method shows outstanding performance compared with state-of-the-art non-profiling methods. Besides, the attack performance of weakly profiling SCA with a DL classifier is comparable with profiling DLSCA, which relies on a more restricted attack assumption: the availability of a cloned device.

Several directions can be investigated following this work. First, knowing that PD_k is imperfect, one could investigate a better approximation of $\mathsf{f}_{\boldsymbol{\theta}_d}$ with, for instance, stochastic models. Second, since the proposed method relies on a priori leakage model, finding a solution to select/estimate the leakage model will be valuable. Third, it would be interesting to compare our method against other attack methods. Finally, exploring the usage of weakly profiling SCA in other ciphers would be an exciting topic.

References

[A+18] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10(978):3, 2018.

[APB+20] Amir Alipour, Athanasios Papadimitriou, Vincent Beroulle, Ehsan Aerabi, and David Hély. 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 *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*, DATE '20, page 614–617, San Jose, CA, USA, 2020. EDA Consortium.

[BCGR22] Julien Béguinot, Wei Cheng, Sylvain Guilley, and Olivier Rioul. Side-channel expectation-maximization attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):774–799, 2022.

[BCH+20] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan. Mind the portability: A warriors guide through realistic profiled side-channel analysis. In *NDSS 2020-Network and Distributed System Security Symposium*, pages 1–14, 2020.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.

[BCP+19] Lejla Batina, Łukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. *Journal of Cryptographic Engineering*, 9:21–36, 2019.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *Journal of Cryptology*, 24(2):269–291, 2011.

[BK02] Régis Bevan and Erik Knudsen. Ways to enhance differential power analysis. In *International Conference on Information Security and Cryptology*, pages 327–342. Springer, 2002.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptographic Engineering*, 10(2):163–188, 2020.

[Bri23] SGS Brightsight. Brightsight security lab in a box (bslb), 2023.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In *Cryptographic Hardware and Embedded Systems-CHES 2009: 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*, pages 156–170. Springer, 2009.

[CLM23a] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. The evil machine: Encode, visualize and interpret the leakage. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 1566–1575, 2023.

[CLM23b] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. Revisiting mutual information analysis: Multidimensionality, neural estimation and optimality proofs. *Journal of Cryptology*, 36(4):38, 2023.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[DLH+22] Ngoc-Tuan Do, Phu-Cuong Le, Van-Phuc Hoang, Van-Sang Doan, Hoai Giang Nguyen, and Cong-Kha Pham. Mo-dlsca: Deep learning based non-profiled side channel analysis using multi-output neural networks. In *2022 International Conference on Advanced Technologies for Communications (ATC)*, pages 245–250, 2022.

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *Journal of Cryptographic Engineering*, 1:123–144, 2011.

[EST+22] Maximilian Egger, Thomas Schamberger, Lars Tebelmann, Florian Lippert, and Georg Sigl. A second look at the ascad databases. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 75–99. Springer, 2022.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 426–442. Springer, 2008.

[GJS19] Aron Gohr, Sven Jacob, and Werner Schindler. Ches 2018 side channel contest ctf-solution of the aes challenges. *Cryptology ePrint Archive*, 2019.

[HDD22] Van-Phuc Hoang, Ngoc-Tuan Do, and Van Sang Doan. Efficient non-profiled side channel attack using multi-output classification neural network. *IEEE Embedded Systems Letters*, pages 1–1, 2022.

[HGM+11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.*, 1(4):293–302, 2011.

[HK11] Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, 30(2):87, 2011.

[KFYF21] Kunihiro Kuroda, Yuta Fukuda, Kota Yoshida, and Takeshi Fujino. Practical aspects on non-profiled deep-learning side-channel attacks against aes software implementation with two types of masking countermeasures including rsm. In *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, ASHES '21, page 29–40, New York, NY, USA, 2021. Association for Computing Machinery.

[KHK22] Donggeun Kwon, Seokhie Hong, and Heeseok Kim. Optimizing implementations of non-profiled deep learning-based side-channel attacks. *IEEE Access*, 10:5957–5967, 2022.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[KWPP23] Marina Krček, Lichao Wu, Guilherme Perin, and Stjepan Picek. Shift-invariance robustness of convolutional neural networks in side-channel analysis. *Cryptology ePrint Archive*, 2023.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[PCP20] Guilherme Perin, Lukasz Chmielewski, and Stjepan Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):337–364, Aug. 2020.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Computing Surveys*, 55(11):1–35, 2023.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 61–81. Springer, 2016.

[PWP22] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):828–861, Aug. 2022.

[Ris23] Riscure. Jlsca: Side-channel toolkit in julia, 2023.

[RWPP21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):677–707, Jul. 2021.

[SK19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 30–46. Springer Berlin Heidelberg, 2005.

[SWP03] Kai Schramm, Thomas Wollinger, and Christof Paar. A new class of collision attacks and its application to des. In *Fast Software Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers 10*, pages 206–222. Springer, 2003.

[Tih22] Tihmstar. Using a magic wand to break the iphone's last security barrier, 2022.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 107–131, 2019.

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, Jun. 2020.

[WEG87] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[WOS14] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic dpa... and the magic of learning. In *Topics in Cryptology–CT-RSA 2014: The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, pages 183–205. Springer, 2014.

[WPP22a] Lichao Wu, Guilherme Perin, and Stjepan Picek. The best of two worlds: Deep learning-assisted template attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(3):413–437, Jun. 2022.

[WPP22b] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IEEE Transactions on Emerging Topics in Computing*, 2022.

[WWJ+23] Lichao Wu, Yoo-Seung Won, Dirmanto Jap, Guilherme Perin, Shivam Bhasin, and Stjepan Picek. Ablation analysis for multi-device deep learning-based physical side-channel analysis. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[WWK+23] Lichao Wu, Léo Weissbart, Marina Krček, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. Label correlation in deep learning-based side-channel analysis. *IEEE Transactions on Information Forensics and Security*, 2023.

[WYL+15] Weijia Wang, Yu Yu, Junrong Liu, Zheng Guo, François-Xavier Standaert, Dawu Gu, Sen Xu, and Rong Fu. Evaluation and improvement of generic-emulating dpa attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*, pages 416–432. Springer, 2015.

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov. 2019.

[Zho18] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.