Label Correlation in Deep Learning-Based Side-Channel Analysis

Wu, Lichao; Weissbart, Léo ; Krcek, Marina; Li, Huimin; Perin, Guilherme; Batina, Lejla; Picek, Stjepan

DOI
10.1109/TIFS.2023.3287728

Publication date
2023

Document Version
Final published version

Published in
IEEE Transactions on Information Forensics and Security

Label Correlation in Deep Learning-Based Side-Channel Analysis

Lichao Wu, Léo Weissbart, Marina Krček, Huimin Li, Guilherme Perin, Lejla Batina, *Senior Member, IEEE*, and Stjepan Picek, *Senior Member, IEEE*

***Abstract*— The efficiency of the profiling side-channel analysis can be significantly improved with machine learning techniques. Although powerful, a fundamental machine learning limitation of being data-hungry received little attention in the side-channel community. In practice, the maximum number of leakage traces that evaluators/attackers can obtain is constrained by the scheme requirements or the limited accessibility of the target. Even worse, various countermeasures in modern devices increase the conditions on the profiling size to break the target. This work demonstrates a practical approach to dealing with the lack of profiling traces. Instead of learning from a one-hot encoded label, transferring the labels to their distribution can significantly speed up the convergence of guessing entropy. By studying the relationship between all possible key candidates, we propose a new metric, denoted Label Correlation (LC), to evaluate the generalization ability of the profiling model. We validate LC with two common use cases: early stopping and network architecture search, and the results indicate its superior performance.**

***Index Terms*— Side-channel analysis, profiling analysis, deep learning, label distribution, profiling model fitting.**

I. Introduction

SIDE-CHANNEL analysis (SCA) is recognized as one of the most powerful attack methods on the implementations of cryptographic algorithms. Commonly, such attacks are divided into direct attacks like Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [16], and two-stage (profiling) attacks like template attack [7], stochastic models [34], and machine learning-based attacks [20], [23], [27]. The profiling attacks impose additional requirements as they assume an 'open' device (or a copy of it). Still, the actual key recovery might need only a few measurements or, in some cases, a single trace [17], [31].

In recent years, machine learning-based attacks positioned themselves as a strong alternative for more 'classical' SCA [4], [17], [42], which has become a standard evaluation approach for security evaluation and certification. The success of such methods relies on a sufficient number of training traces so that a machine learning classifier can accurately map the relationship between the traces and corresponding labels (intermediate data). In the worst-case attack scenario, an attacker can obtain unlimited training traces from the clone device for profiling attacks. However, an easy-to-ignore fact, especially in SCA research, is that it is not easy to acquire a large number of attack traces, even for a white-box evaluation in a security lab. We rarely see research focus on reducing the number of profiling traces number. Indeed, such a restriction mainly comes from three factors: time constraints, countermeasures, and the device's life cycle. We argue the importance of developing techniques that effectively decrease the required number of profiling traces while keeping a similar attack performance.

- Time constraints: An evaluation's time budget dramatically limits the number of traces one can obtain. For instance, according to [18], measuring one million profiling traces for a software RSA implementation with a 1 024-bit key could take more than a week. Additionally, in post-analysis tasks such as trace realignment, noise filtering, and leakage assessment, an evaluator may not have enough budget to measure sufficient traces to break the target. Therefore, reducing the required number of profiling traces would save time and enhance the evaluator's attack capability.
- Security Countermeasures: Unlike most deep learning applications, the SCA training data are most likely 'protected' - the SCA countermeasures represent a standard/default setting for the modern smart card/SoC's implementations. These protection mechanisms further increase the difficulties in learning the trace-label relationship, thus increasing the demand for the number of measurements. From a security developers' point of view, an increasing number of side-channel measurements to break the target implementation means higher security assurance of their product. If we can effectively reduce the required number of profiling traces, such vulnerabilities will be considered again.
- Application-level Protections: For a black/grey box evaluation, the available traces can drop to hundreds or thousands due to the upper limit of program counters such

Manuscript received 11 November 2022; revised 5 April 2023; accepted 11 May 2023. Date of publication 19 June 2023; date of current version 30 June 2023. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ulrich Rührmair. *(Corresponding author: Lichao Wu.)*

Lichao Wu, Marina Krček, and Huimin Li are with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 XE Delft, The Netherlands (e-mail: lichao.wu9@gmail.com).

Léo Weissbart, Guilherme Perin, and Stjepan Picek are with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 XE Delft, The Netherlands, and also with the Digital Security Group, Radboud University, 6525 EC Nijmegen, The Netherlands.

Lejla Batina is with the Digital Security Group, Radboud University, 6525 EC Nijmegen, The Netherlands.

Digital Object Identifier 10.1109/TIFS.2023.3287728

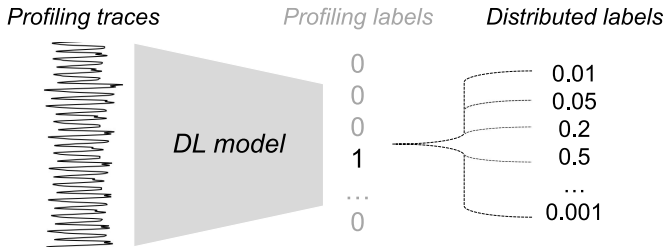


Fig. 1. Learning with distributed labels.

as Application Transaction Counter (ATC) or PIN Try Counter (PTC) [3], which is commonly insufficient when implementing an efficient profiling model. Building a profiling model with limited profiling traces would significantly increase the capability of exploiting the potential vulnerabilities protected by life-cycle-related counters.

There are limited evaluation metrics optimized for SCA. Evaluation metrics are essential in the training process: by actively monitoring the metric value, one can easily interpret the learning process, e.g., underfitting or overfitting. However, accuracy, a commonly used metric for deep learning, is less indicative for SCA in multi-trace attack scenarios [17]. The reasons can be explained from two aspects. First, side-channel leakages are more difficult to classify due to the noise/countermeasures in the traces. Second, accuracy does not represent the success of an attack well, as we commonly need to consider continuous attacks that are better evaluated with metrics capturing this continuity. Guessing entropy and success rate are the commonly used metrics for SCA. Unfortunately, using such evaluation metrics would significantly increase the training time due to their computation complexity. Moreover, guessing entropy *only* evaluates the rank of the correct key. Although effective, we argue that it can be less indicative as the internal relationships with other (wrong) key candidates are not considered. More discussion is available in Section V-B.

We put the above concerns forward as the motivations for this work. First, to reduce the required number of training traces, we transfer the one-hot encoded labels to their Gaussian distribution centering on the corresponding labels motivated by [10]. The proposed learning scheme is illustrated in Figure 1. A one-hot encoded label that belongs to class 4 has been transferred to the distributed label with the value of the fourth index with the highest probability. Based on our experiment, regardless of the used leakage model and deep learning architecture, if using our learning scheme, the profiling traces can be reduced more than five times compared with the number of profiling traces used in the literature.

One essential assumption of the distributed label is that the label closer to the correct label is more likely to be selected. Under the same assumption, we propose key distribution to measure the geometric distance between the most likely key (not necessarily the correct key) and all the other keys. From this method and guessing entropy estimation, we propose a novel profiling model fitting metric - Label Correlation (LC) that calculates the correlation between key distribution and the key guessing vector of all key guesses. As demonstrated with experiments on publicly available datasets, the proposed metric can indicate the generalization ability of a profiling model and thus serve as a reliable evaluation metric of early stopping and network architecture search. LC is more indicative than conventional metrics, such as validation (cross-entropy) loss, as it directly links with the attack performance. On the other hand, compared with GE, LC requires less computation effort as it does not rely on the averaging of multiple realizations of key ranks [40]. Thus, it can be a good metric to monitor the model training.

To summarize, the main contributions of this paper are:

1) We introduce a new efficient training scheme for profiling SCA when the number of profiling traces is limited. The attack performance is improved by learning from the distributed labels compared to conventional one-hot encoded labels.
2) We propose a novel method to calculate the distance between the target key and other keys called key distribution (KD).
3) Based on the guessing entropy, we introduce a new metric called Label Correlation (LC) that can effectively estimate how well the profiling model fits the data. To that end, we show that the proposed metric is reliable in reflecting the generality of the profiling model. We demonstrate two use cases for potential implementors: early stopping and network architecture search in various testing conditions. The results show that the LC metric performs better than other commonly used metrics.

We provide comprehensive experimental results on publicly available datasets to validate our claims. We also consider two commonly used deep learning architectures in SCA: multilayer perceptron and convolutional neural networks. The source code is available in the GitHub: https://github.com/AISyLab/Label-distribution-and-correlation.

The paper is organized as follows. Section II provides information about profiling SCA, commonly used evaluation metrics, and datasets used in this paper. The related works are discussed in Section III, followed by the proposal of the label distribution learning and novel SCA evaluation metric LC in Sections IV and V. Section VI validates the proposed methods experimentally with different datasets, attack models, and leakage models. Finally, Section VII concludes this paper and proposes possible future research directions. In Appendix A, we provide details about the neural network architectures we used.

II. Background

A. 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).

k represents a key byte candidate that takes its value from the keyspace $\mathcal{K}$. k^* is the correct key byte, and k^{ref} is the

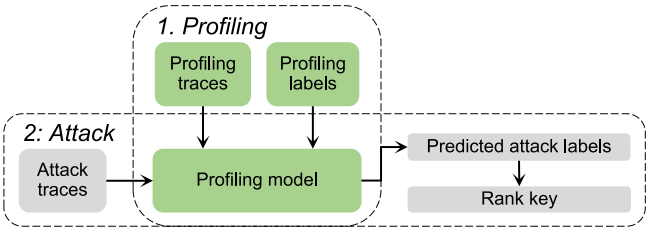


Fig. 2. Profiling side-channel analysis.

key byte assumed by an attacker to be correct (as the attacker does not know the correct key).[1]

A dataset $\mathbf{T}$ is defined as a collection of traces $\mathbf{t}_i$, where each trace $\mathbf{t}_i$ is associated with a key-related label (or the key itself) y_i. A complete set of labels with c classes is denoted by $\mathcal{Y} = \{y_1, y_2, \cdots, y_c\}$. The number of profiling traces equals N, the number of validation traces equals V, and the number of attack traces equals Q. Finally, $\boldsymbol{\theta}$ denotes the vector of parameters to be learned in a profiling model.

B. Profiling Side-Channel Analysis

As depicted in Figure 2, profiling side-channel attacks consist of two phases:

1) **Learning or profiling phase.** The profiling phase consists of building a profiling model $\mathsf{f}_M^{\boldsymbol{\theta}}$ to map the inputs (side-channel measurements) to the outputs (classes as obtained by evaluating the leakage model on the sensitive operation) on a set of N profiling traces. $\mathsf{f}_M^{\boldsymbol{\theta}}$ represents the profiling model trained for a given leakage model M and the set of learning parameters $\boldsymbol{\theta}$. This phase aims to fit the parameters of a function that maps the side-channel traces to the labels in the best way (minimizing the error function). It is common to use the validation set of size V to know when to stop the learning process.
2) **Test or attack phase.** The attack phase consists of obtaining label predictions for the traces from a different dataset of size Q to test the model. The trained model processes each attack trace and produces the attack's output as a vector of probabilities $\mathbf{p}_{i,j} \in \mathcal{K}$, where each index is the probability that a trace t_i is associated with the leakage value j. We can estimate the attack performance from this matrix of probabilities (as we have multiple vectors - one for each attack trace).

We consider two common profiling approaches:

- **Template Attack.** Template attack (TA) uses Bayes' theorem to obtain predictions, dealing with multivariate probability distributions as the leakage over consecutive time samples is not independent [7]. In the state-of-the-art, template attack relies mostly on a normal (Gaussian) distribution.
- **Deep Learning-based SCA (DL-SCA).** We consider supervised machine learning and the classification task as the side-channel attack's goal. Supervised learning deals with the task of learning a mapping f from a set of input variables from $\mathcal{X}$ to the set of output variables Y ($\mathsf{f}_M^{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y}$). For SCA, the profiling phase aims to learn the parameters $\boldsymbol{\theta}$, minimizing the empirical risk represented by a loss function on a dataset. In the attack phase, the goal is to predict the classes (more precisely, the probabilities that a certain class would be predicted) based on the previously unseen set of traces and the trained model $\mathsf{f}_M^{\boldsymbol{\theta}}$.

C. Evaluating the Attack Performance

An attack's output is the logarithmic sum of all Q probability vectors of single model predictions, where each index is associated with one key hypothesis. Sorting this vector by decreasing probabilities leads to a key guessing vector with increasing confidence predicted by a profiling model. The key rank denotes the position of the correct key. Then, one can use metrics such as guessing entropy to estimate the attacker's performance [35].

Definition 1 (Key Guessing Vector): The key guessing vector $\mathbf{g}$ is the vector of probabilities for all key candidates from the output of the profiling model's predictions:

$$\mathbf{g} = \mathsf{sort}\left(\sum_{i}^{Q} log\ \mathbf{P_r}(\mathbf{t}_i; \mathsf{f}_M^{\boldsymbol{\theta}})\right), \tag{1}$$

where $\mathbf{P_r}(\mathbf{t}_i; \mathsf{f}_M^{\boldsymbol{\theta}})$ is the prediction vector from the profiling model f_M^{θ} on a trace $\mathbf{t}_i$. sort is the function sorting array elements in order of decreasing values of their probabilities. Since the labels are key-related, the cumulative probabilities of labels can be easily mapped to their corresponding keys. From $\mathbf{g}$, the index of g represents 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.

Definition 2 (Key Rank): In a known-key setting, the key rank is the number of (most likely) keys an attacker needs to brute force until recovering the correct key. Among various key enumeration techniques [30], one of the more popular methods is to try every key given its probability after generating a key guessing vector. In this scenario, the key rank is the position of the correct key in the guessing vector.

Definition 3 (Guessing Entropy): The guessing entropy[2] represents the averaged rank of the correct key k^* in the key guessing vector $\mathbf{g}$:

$$GE = \mathbf{E}\left(\mathsf{rank}_{k^*}(\mathbf{g})\right), \tag{2}$$

where $\mathsf{rank}_k(\mathbf{g}) \in \{0, \ldots, |\mathcal{K}|-1\}$. $\mathbf{E}$ is the average of multiple realizations of key rank, which is commonly performed by attacking with a profiling model multiple times with randomly selected attack traces.

D. Datasets

1) ASCAD Dataset: The ASCAD dataset is generated by taking measurements from an ATMega8515 running a masked

[1] Note that the subkey candidates can have any number of bits that are being guessed and while here we assume the AES cipher scenario, the concept is algorithm-independent.

[2] As we attack only a single key byte, the proper term is partial guessing entropy. Nevertheless, we use the two terms interchangeably.

AES-128 implementation and is proposed as a benchmark dataset for SCA [1]. Side-channel traces represent the AES encryption, where the commonly attacked trace interval represents the processing of the third byte in the S-box operation (S-box is fixed and publicly known for AES) taking place in the first round (the third byte is the first masked one). The operation is masked, and we assume no knowledge about masks in the profiling phase. There are two versions of this dataset:

1) ASCAD_f: The first version consists of 50 000 profiling traces and 10 000 attack traces, where each trace consists of 700 features (pre-selected window around the leaking spot). The profiling and attacking sets use the same fixed key, and we denote this dataset as ASCAD_f.
2) ASCAD_r: The second version of the ASCAD dataset contains 200 000 traces for profiling with random keys and random plaintexts and 100 000 for the attack phase, with a fixed key and random plaintexts. A window of 1 400 points of interest is extracted around the leaking spot. We denote this dataset as ASCAD_r.

For both datasets, different numbers of profiling and attack traces are used in our experiments (see Section VI for details), and 5 000 traces are used for validation and attack. The datasets are provided at https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1.

2) CHES CTF Dataset: 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. In our experiments, we consider 45 000 traces for the training set, which contains a **fixed key**. The validation and attack sets consist of 5 000 traces. Each trace consists of 2 200 features. This dataset is available at https://chesctf.riscure.com/2018/news.

E. Leakage Models

The leakage model simulates the hypothetical power consumption to process one byte (as we attack the AES cipher that is byte-oriented). Our work considers two commonly used leakage models: the Hamming Weight (HW) and Identity (ID). For the HW leakage model, the attacker assumes the leakage is proportional to the sensitive variable's Hamming weight. This leakage model results in nine classes for a single intermediate byte for the AES cipher. In terms of the ID leakage model, an attacker considers the leakage in the form of an intermediate value of the cipher. This leakage model results in 256 classes for a single intermediate byte for the AES cipher.

III. Related Works

In Chari et al.'s seminal work, the authors proposed the template attack (TA) and showed that it could break implementations secure against other forms of side-channel attacks [7]. This attack is the most powerful one from the information-theoretic point of view, but to reach its full potential, it requires an unbounded number of traces and noise following the Gaussian distribution [20]. Template attack is interesting as it is a generative technique, which means it will commonly overfit less as it allows the user to provide more information in the form of class conditionals.

While machine learning techniques have been widely used for several decades, the SCA community showed interest in such techniques only around a decade ago. In the beginning, the most attention was given to techniques like random forest [19], support vector machines [13], [15], [27], and multilayer perceptron [11] (commonly in the context of shallow learning as it had only a single hidden layer).

The rapid development of deep learning-based SCAs started in 2016 when Maghrebi et al. demonstrated the strong performance of several neural network types, most notably, convolutional neural networks [23].

In [1], an empirical evaluation for different hyperparameters is conducted for CNNs on the ASCAD database. In [42], the authors proposed a methodology to select hyperparameters related to the size (number of learnable parameters) of layers in CNNs. The methodology includes observations for the number of filters, kernel sizes, strides, and neurons in fully connected layers. Wouters et al. showed how to reach similar attack performance with data regularization and even smaller neural network architectures [37]. Perin et al. investigated deep learning model generalization and demonstrated how ensembles of random models could perform better than a single carefully tuned neural network model [26]. Wu et al. and Rijsdijk et al. explored different automatic hyperparameter tuning strategies, namely Bayesian optimization [38] and reinforcement learning [33] paradigms to find neural networks that perform well. While their approach requires a significant tuning effort (computational time), the authors improved state-of-the-art results. These works showed that deep learning models' good performance relies on an efficient selection of hyperparameters for specific datasets. If those hyperparameters are not selected properly, the attack will fail (or at least not work as well as possible).

It is intuitive that the number of measurements and input features also limits the performance of a profiling attack. Deep neural networks provide top-level performances in many domains when training data is sufficiently large. However, they could also perform excellently when the training data is reduced. In an effort to improve attack performance, already Choudary et al. investigated how adding noise to the input improves the performance of template attacks on different devices [5]. The same idea is applied to deep learning-based attacks, introduced by Kim et al. [17]. In the context of profiling side-channel attacks, Cagli et al. investigated how to create measurements that improve the attack performance synthetically [4]. Unlike the previous work where the authors developed a specialized data augmentation technique, Picek et al. showed that generic data augmentation techniques help in profiling SCA also [28]. Another work investigated whether limiting the number of traces can be beneficial both from the experimental setup and performance sides [29].

From the input features perspective, Bursztein introduced the usage of raw traces for profiling in an invited talk at CHES 2018 [2]. Lu et al. also worked in this direction, showing that better attack performance is achieved but with significantly

more complex neural networks (e.g., having around 50 layers) [21]. Perin et al. showed how simple re-sampling of raw traces could result in extremely powerful attacks (requiring only a single attack trace) while using simple neural networks with only a few hidden layers [31]. Finally, similarity learning was applied to pre-process the leakage traces and extract high-level features, leading to state-of-the-art attack performance with significantly reduced computation effort [39].

As already discussed, commonly, in machine learning, one estimates the behavior of a profiling model based on statistics of individual observations like accuracy, loss, or recall. Unfortunately, such metrics can be misleading in SCA, as one considers cumulative predictions. Picek et al. showed that standard machine learning metrics could suggest radically different performance than the SCA metrics [28]. Masure et al. connected the perceived information and negative log-likelihood, showing there can be common ground when using machine learning metrics in SCA [22]. Perin et al. discussed how mutual information could be a good metric to indicate when to stop the machine learning training process [25]. Finally, Rădulescu et al. compared efficient metrics (Massey's guessing entropy and empirical guessing entropy) in full-key recovery, which may help decide when to stop profiling [32].

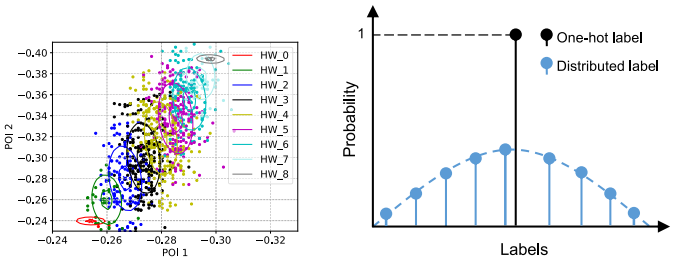
IV. Label Distribution

By asking *'how much does each label describe the instance?'*, Geng et al. first proposed Label Distribution Learning (LDL) by assigning a *description degree* (probability) of each possible label, leading to enhanced performance compared with hard (one-hot encoded) labels [10]. This method has been used in tasks such as age estimation [12] or personality recognition [41]. However, the application of LDL is restricted since one should have a reasonable estimation of the relation between labels, and such an estimation could be challenging in many tasks, e.g., image classification. Fortunately, SCA uses the leakage model to construct labels, which inherently leads to a clear relationship between labels. Indeed, two leakage traces with closer label (intermediate value) distance could be more similar. As a result, a combination of LDL and SCA could enhance attack performance.

Definition 4 (Description Degree): The description degree $d_x^{y_i}$ represents the degree of a label y_i to describe an input x. From the machine learning perspective, $d_x^{y_i}$ can be considered as the probability of the label y_i being selected. If a complete set of labels $\mathcal{Y}$ can fully describe the given input, then:

$$\sum_i d_x^{y_i} = 1,\ y_i \in \mathcal{Y}. \tag{3}$$

The conventional DL-based SCA represents a multi-class classification task that describes a measurement with a unique cluster/label. Using binary variables, the label is one-hot encoded to train a deep learning model (see Figure 3b). In an ideal case, the label y_i perfectly represents the leaking features within a measurement (i.e., the correlation between labels and leaking features equals one). However, the presence of noise/countermeasure increases the description degree of other labels to the corresponding leakage traces. For illustration, Figure 3a shows the Probability Density Function (PDF), and



(a) PDFs and POIs distribution for the correct key. (b) Comparison between one-hot and distributed labels.

Fig. 3. PDFs and a demonstration of distributed labels.

point-of-interests distributions (POI1 and POI2) from 1 000 measurements.[3] The color of each point is attributed based on its cluster label. Using the HW leakage model, nine PDFs representing nine HW clusters are built during the profiling phase. Each PDF is represented by two ellipses representing 0.5 (low) and 0.9 (high) of the maximum probabilities. Note that PDF is the basis of template attack, used to present the leakages' distributions and making label predictions [5], [6].

From Figure 3a, each PDF can be separated. However, the overlap between each PDF cannot be ignored. Although the traces in the middle between two PDFs have deterministic (single) labels representing the targeted intermediate data, they are also geometrically close(r) to their neighboring clusters leakage-wise. Here, we denote the squared Euclidean distance between two label values as *label distance*. Indeed, one can observe a natural measure of description degree that associates the labels with the traces. An accurate description of these traces should involve the 'incorrect' labels. Since their similarity to each cluster is inversely correlated with their label distance, as demonstrated in Figure 3b, the one-hot label and the highest distributed label should be on the same abscissa; the distribution degree of other labels is assigned with reduced probability based on label distance. We denote this label representation as *distributed labels*. The description degree of each label is sorted based on their actual values. Distributed labels more precisely describe the leakage features, thus helping relax the conditions on the required number of training traces to achieve a robust performance than training with one-hot encoded labels.

It is worth noting the links between template attacks and distributed labels. The multivariate normal distribution, parameterized by the mean vector μ and covariance matrix Σ, can be represented using the following equation

$$f(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{k}{2}} \cdot |\Sigma|^{\frac{1}{2}}} \cdot \exp^{-\frac{1}{2}(x-\mu)^T \cdot \Sigma^{-1} \cdot (x-\mu)}, \tag{4}$$

where x represents the random vector from the multivariate normal distribution with k dimensions. Indeed, both methods calculate the mean and variation of the variable to precisely describe the leakage features. The main difference is that the

[3]ChipWhisperer dataset [24] is used as it represents measurements obtained from a physical device, where two point-of-interests are selected based on the signal-to-noise ratio to represent the traces. Note that this dataset is not noiseless, but obtaining less noisy measurements without resorting to simulations is challenging.

template attack directly characterizes the leakage features with μ and Σ, while our method focuses on enhancing leakage features' label representation.

Notice, our learning method fundamentally differs from linear regression attack (LRA) [8]. Although LRA would also lead to smooth labels by estimating weight parameters to each binary decomposition of the target value, these labels are constructed during the regression process. On the other hand, our method considers SCA as a classification task. The goal of using distributed labels is to reach more efficient classification.

One should notice the relation between label smoothing [36] and label distribution. As a regularization technique, label smoothing improves accuracy by computing cross-entropy not with the 'hard' (i.e., one-hot encoded) labels from the dataset but with a weighted mixture of all possible labels with the noise (i.e., uniform) distribution. Label distribution further preserves relations between different labels to describe leakage features. From the model training perspective, the model is less penalized by the loss value caused by the inconsistency between predictions and real labels (e.g., one-hot labels), thus speeding up the learning process. The performance benchmark between these two techniques can be found in Section VI-A. A natural choice to form distributed labels is a normal distribution. Indeed, the construction of distributed labels should align with the actual distribution of leakages. This paper assumes the leakage follows a Gaussian distribution commonly observed in practice. Moreover, Gaussian distribution inherently fits the distribution of the environmental noise, a main factor that increases the description degree of labels.

Definition 5 (Label Distribution Learning): Given a training set with trace-label pairs (x, y) sampled from $\mathbf{T}$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the goal is to learn a function f_M^{θ}, so that the predicted output $\hat{y}$, representing the probability of all possible labels given an input x, has a similar distribution to the distributed label $D(y)$:

$$D(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-y'}{\sigma}\right)^2\right), y' \in \mathcal{Y}, \quad (5)$$

where $D(y)$ denotes the distributed label for the input label y; the variance of is denoted by σ.[4]

An essential assumption of label distribution learning is that the label y should be pre-determined by an attacker. Then, the attacker can calculate the distributed label $D(y)$ with Eq. (5). Indeed, the only adjustable parameter σ depends on the data property (specifically, the dataset's noise). In Section VI, we systematically analyze the influence of σ with different datasets (including their noisy versions) and leakage models and then give suggestions on the value selection.

Next, to optimize the learning parameter θ, instead of using conventional loss functions such as categorical cross-entropy or mean squared error, following [10], Kullback-Leibler (KL) divergence is used as the loss function to measure the similarity between the predicted and ground truth distribution:

$$L = \sum_i D(y_i) \ln\left(\frac{D(y_i)}{\hat{y}_i}\right), \quad y_i \in \mathcal{Y}, \quad (6)$$

where $\hat{y}_i$ and y_i denote the predicted probability for label i and the true label, respectively.

Stochastic gradient descent is used to minimize the loss function L. Once a network is trained, given a random input x with an unknown label from the attack dataset, the model f outputs a predicted label distribution $\hat{y}$. The predicted label is the one in $\hat{y}$ with the highest probability.

$$i^* = \underset{i}{\text{argmax}}\, \hat{y}_i. \quad (7)$$

V. Label Correlation Metric

A. Key Distribution

Following Eq. (5), the probability of a label y being selected as the correct label depends on its label distance to the true label y^*. Since these labels are key-related, we can also calculate the differences between key candidates, denoted as *key distribution* (KD), based on the label distance.

$$KD(k^{ref}, k) = \left\| \mathsf{f}(d, k^{ref}) - \mathsf{f}(d, k) \right\|^2, k \in \mathcal{K}. \quad (8)$$

where f is the leakage model function (described in Section II-E) that returns the leakage value (labels) according to a key candidate k and data value d. Similar to Eq. (5), KD is based on the squared Euclidean distance between the leakage distribution of all key hypotheses $k \in \mathcal{K}$ and the reference key candidate k^{ref}. We form a KD vector sorted by the KD value for each k (so k^{ref} always ranks the first).[5] Note that when it is clear from the context, we use the notations $KD(k^{ref}, k)$ and KD interchangeably.

KD gives a unique distribution of all key candidates k based on their difference to the reference key k^{ref}. Therefore, k^{ref} determines the KD value for each key candidate. Typically, k^{ref} has a distribution difference equal to zero with itself, and the lower the distribution difference, the more similar the key candidate is to the reference key. The reference key can be set to the k^* (correct key). When k^* is unknown (black-box), k^{ref} should be the most likely key.

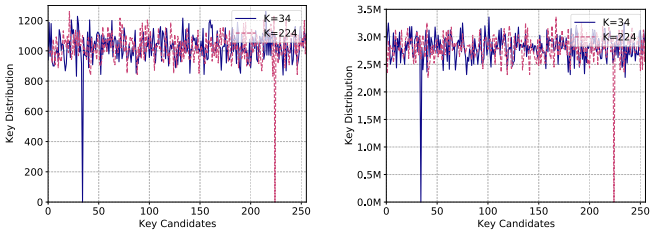
From an attack perspective, for a model built in a successful profiling attack (the correct key k^* is the best guess), suppose KD is large between a specific key $k \in \mathcal{K}$ and k^*. Then, k will likely be ranked low (i.e., with guessing entropy close to $2^b - 1$) as it has a negligible probability of being selected. Consequently, KD can be considered an ideal key rank[6] metric indicating the best possible scenario where the correct key is maximally separated from all the other keys.

The KD definition can be extended to any leakage model, i.e., the Hamming distance or the Least Significant Bit.

[4] We assume the leakage follows a Gaussian distribution. If this assumption does not hold for the given leakage traces, the calculation of the distributed label should be adjusted accordingly.

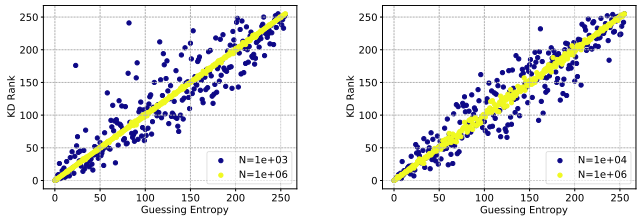
[5] We also investigated the Manhattan distance and found the results to be in line but with smaller discriminate power. Besides, since KD is a list of labels associated with the given key that does not follow any known distribution, f-divergence functions (i.e., KL-divergence, Hellinger distance) are not considered.

[6] Here, 'ideal' means the perfect fit between an attack model and the leakage. Under this circumstance, the resulting key rank is equivalent to KD as discussed in Section V-B.



(a) HW leakage model. (b) ID leakage model.

Fig. 4. Illustration for the Key Distribution for the HW/ID leakage models and correct keys 34 and 224.



(a) HW leakage model. (b) ID leakage model.

Fig. 5. 'Perfectly' fitted profiling model with template attack, considering the HW and ID leakage models and simulated traces with an increasing number of profiling traces N. KD ranks (Y-axis) stands for a sorted KD.

Figure 4 illustrates the summed KD (for all key candidates) with the HW and ID leakage models for the key candidates $k^{ref} = 34$ (correct third subkey for the ASCAD_r) and $k^{ref} = 224$ (correct third subkey for the ASCAD_f). Although all KD values except for $k^{ref} = k^*$ act as 'noisy' values, the similarity difference between k^{ref} and other keys can be observed. One should note that a precise estimation of KD relies on the chosen leakage model. An incorrect leakage model would not only degrade the attack performance, but KD's effectiveness will also drop. Since the publicly available datasets leak mostly in the HW leakage model, we calculate KD with the HW leakage model throughout the paper.

B. Label Correlation - LC

Key distribution defines the distance between k^{ref} and other key candidates. Following this, we define a *profiling model fitting metric* by correlating KD with the predicted probability for all $k \in \mathcal{K}$, denoted as *Label Correlation* (LC), as a function of KD and the key guessing vector $\mathbf{g}$ (defined in Definition 3):

$$LC = \mathsf{corr}(KD, \mathbf{g}). \tag{9}$$

Eq. (9) defines how well a profiling model fits the data concerning a key candidate k^{ref} for a chosen leakage model. The notation corr represents the Spearman correlation [14] that evaluates the monotonic relationship with two inputs. We also considered the Pearson correlation, but the results are less optimal due to the significant differences in key distribution between the correct key and other keys (Figure 4).

Following Eq. (9), if the profiling model outputs the correct key as the most likely key, one could expect a stronger correlation between KD and $\mathbf{g}$. Conversely, if the profiling model fails to fit the data, the outputted random (but still) most likely key would lead to a low correlation between KD and $\mathbf{g}$. As a demonstration, Figure 5 depicts the 'almost' perfectly fitted profiling model for the HW and ID leakage models. We use simulated measurements with strong HW and ID leakages and a controlled Gaussian noise level, normal-distributed with a variance of 0.01 around a mean of zero. The simulated traces have two features that hold the leakage, which is proportional to $HW(S_{box}(d \oplus k))$ and $S_{box}(d \oplus k)$, to simulate the ideal HW and ID leakages, respectively. The profiling set has plaintexts d and keys k chosen from a uniformly random distribution. The attack set's plaintexts are selected uniformly at random, while the attack key is the same for the whole dataset. We use the template attack and consider the increasing number of profiling traces N. In both figures, LC increases w.r.t. the number of profiling traces, reaching 0.999 and 0.998 for the HW and ID leakage models. The results confirm that the correlation between KD and $\mathbf{g}$ tends to increase with better (fitter) models (since we use template attack, better models are those that are trained with more traces).

Definition 6 (Perfectly Fitted Profiling Model): A perfectly fitted profiling model reaches $LC = 1$ in the attack phase for any set of Q attack traces.[7]

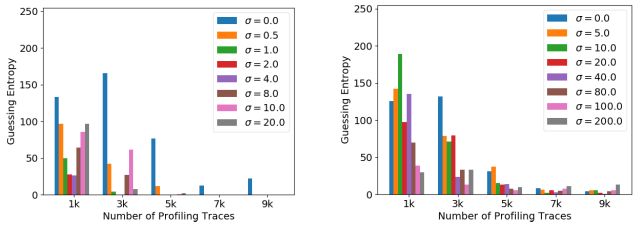
It is worth mentioning that KD can also be used to calculate the confusion covariance metric ($\mathbb{E}(KD)$) [9], a metric designed initially to measure the DPA resistance of S-boxes. A low expectation of KD indicates less distinctive intermediate data, which could lead to reduced data leakage. On top of that, the LC metric suggests that the variance of KD should also be low to secure the target. Indeed, a low KD variance indicates high similarity between different keys, which will lead to a less deterministic order of $\mathbf{g}$. Since the LC value is more likely to be low in this case, one can expect more effort in obtaining the security assets via SCA. Another way of perturbing $\mathbf{g}$ is by introducing additional noise or countermeasures, a common implementation in modern products.

VI. Experimental Results

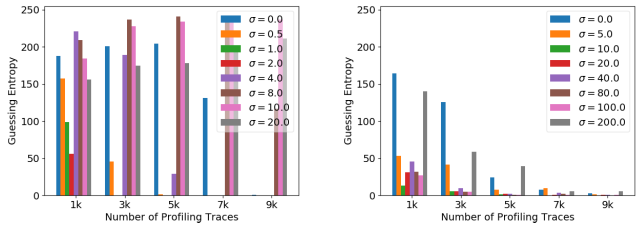
A. Profiling With Distributed Labels

In Section IV, we argue that the distributed label enlarges the description degrees of labels to the leakage traces and can lead to more efficient learning even with fewer profiling examples. We validate this assumption with machine learning models by training the state-of-the-art CNNs [33] and MLPs [38] with a different number of profiling traces. The models' hyperparameters are listed in Appendix . We use a diverse selection of DL architectures to ensure the generalization of the results. Note that we select MLP and CNNs due to their wide applications in SCA. Still, we expect other supervised learning methods to benefit from distributed labels thanks to their higher description degree of the leakage features. Besides, we tune the σ value of the distributed label to find the optimal value for different training settings. The distributed labels are pre-computed before the training starts. To obtain the most representative performance, the

[7]We assume there are many possible ways to select Q traces from the available traces.

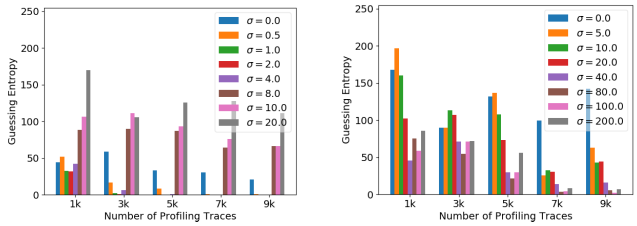


(a) MLP with the HW leakage model. (b) MLP with the ID leakage model.

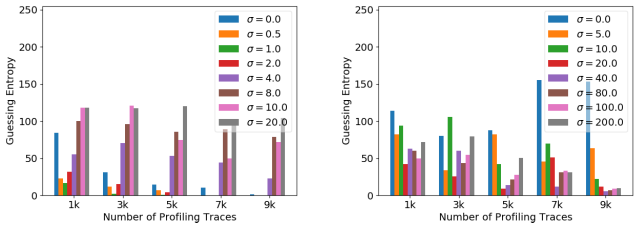


(c) CNN with the HW leakage model. (d) CNN with the ID leakage model.

Fig. 6. Label distribution learning on the ASCAD_f dataset.



(a) MLP with the HW leakage model. (b) MLP with the ID leakage model.



(c) CNN with the HW leakage model. (d) CNN with the ID leakage model.

Fig. 7. Label distribution learning on the ASCAD_r dataset.

attack results of each training setting (σ and profiling traces number) are the median value from 20 independent training (and attacks) with random weight initialization following recommendations from [40].

Figures 6, 7, and 8 show the results for the ASCAD_f, ASCAD_r, and CHES_CTF datasets, respectively. The conventional training method (one-hot encoded label) is represented with $\sigma = 0.0$ (black bar). We used the categorical cross-entropy (CCE) loss when training with the conventional method. CCE is a standard loss function for classification tasks widely used in DL-SCA. When learning from the label distribution, the KL divergence loss measures the distribution difference between the true and predicted label distributions.

For the ASCAD_f dataset, as shown in Figure 6, by distributing HW-based labels, GE equal to zero can be reached with less than 3 000 profiling traces for both MLP and CNN within the given number of attack traces, which is more than ten times less than the number of the profiling traces commonly used in literature (50 000). At the same time, more than 10 000 profiling traces are insufficient when considering the conventional training method ($\sigma = 0.0$). Using the ID leakage model, although one-hot encoded labels lead to better performance in some cases (discussed in later paragraphs), one can confirm the advantages of using the distributed label in low profiling settings.

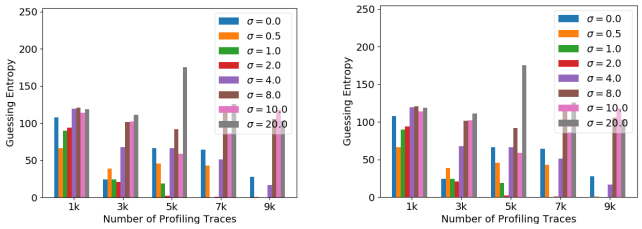
When looking at the influence of the label distribution variation σ (Figure 6), although different numbers of profiling traces, leakage models, and attack models are considered, the optimal settings show consistency: for the HW leakage model, σ ranges from 1 to 2 can lead to the best attack performance. This value increases to 20-80 for the ID leakage model. We have also tested the traces with Gaussian noise levels 2 and 4. While the optimal value of σ defined in the paper still holds for most settings, we expect the best σ to be larger since the leakage traces become more difficult to classify correctly.

Although ASCAD_r is considered more difficult to break than ASCAD_f [38], as shown in Figure 7, the distributed label boosts the attack performance significantly. For the HW leakage model, around 6 000 profiling traces are sufficient for MLP and CNN models to reach GE of zero, which is around ten times less than the related works ($\geq$50 000 profiling traces). For the ID leakage model, aligned with the attack on the ASCAD_f dataset, although none of the training settings can retrieve the secret information with 5 000 attack traces, label distribution learning halves the GE value compared with its one-hot encoded counterpart, indicating a faster GE convergence with our learning scheme.

Finally, similar results can be obtained when attacking the CHES_CTF dataset. Since this dataset leaks limited ID leakage according to literature [33], [38], we attack with the HW leakage model only. With the MLP and CNN models, 5 000 profiling traces are needed to break the target, nine times less than the traces used in the literature (45 000 traces). It is important to note that the optimal σ setting shows similarity for all three tested datasets. From the experimental results on three datasets, good prior knowledge about the leakage model is necessary to construct a meaningful label distribution. Still, when attacking leakages from other devices, one could start with low σ and monitor the attack performance until it reaches optimal behavior.

Indeed, there are various techniques available when the profiling traces are limited. To better illustrate the pros and cons of label distribution learning compared to these methods, we benchmark the attack performance of previously used state-of-the-art (SotA) MLPs and CNNs with multiple profiling settings. We consider several commonly-used techniques to counter the limitation of the training data.

- 10 000 profiling traces with and without techniques such as label distribution, label smoothing, L2 regularization, and dropout.



(a) MLP with the HW leakage model. (b) CNN with the HW leakage model.

Fig. 8. Label distribution learning on the CHES_CTF dataset.

TABLE I

BENCHMARK THE ATTACK PERFORMANCE (T_{GE0}) WITH SOTA MLP. ATTACK RESULTS FOR THE HW AND ID LEAKAGE MODELS ARE SEPARATED BY '/'

Traces	Label	ASCAD_f	ASCAD_r	CHES_CTF
10 000	One-hot	-/-	-/-	-
	Smoothed	3 484/**-**	**-/-**	**-**
	Distributed	**1 618/4 964**	**3 623/4 892**	**2 337**
10 000	One-hot	-/-	-/-	3 728
(L2)	Distributed	-/-	-/-	**1 930**
10 000	One-hot	-/-	-/-	4 156
(Dropout)	Distributed	**2 264**/-	-/-	**2 493**
50 000	One-hot	**1 219/182**	970/**2 625**	**567**
	Distributed	1 421/3 530	**919**/-	905
50 000	One-hot	1 588/-	-/-	-
(Augmented)	Distributed	**1 095/4 728**	**2 784**/-	**2 735**
100 000	One-hot	**1 254**/-	-/-	-
(Augmented)	Distributed	1 447/**4 895**	**2 998**/-	**2 793**

- 50 000 profiling traces, obtained directly or generated with Gaussian noise-based data augmentation.
- 100 000 profiling traces generated from 10 000 traces with Gaussian noise-based data augmentation.

The dropout rate and regularization factor are tuned to 5e-2 and 1e-4. For data augmentation, four augmentation levels (0.25, 0.5, 0.75, 1.0) are selected following [17], and the one with the best performance is presented in the benchmark. The label smoothing factor is set to be optimal based on the various search options.[8] Each profiling setting is tested with two label formats: one-hot encoded and distributed labels. For label distributed learning, σ is set to 1/2 and 40/80 for HW and ID leakage models. Recall that the number of attack traces is set to 10 000. The attack performance is evaluated by calculating the required number of attack traces to reach GE of zero, denoted as T_{GE0}. The results are the median T_{GE0} from 20 independently trained models. If an attack setting fails to reach GE zero with a given number of attack traces, the results are marked with "-".

The benchmark results are shown in Table I and Table II. The best results for each profiling setting are marked in **bold**. With limited (10 000) profiling traces, distributed labels bring a significant performance boost with various attack settings.

[8]The possible label smoothing factors are 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 5, and 10.

TABLE II

BENCHMARK THE ATTACK PERFORMANCE (T_{GE0}) WITH SOTA CNN. ATTACK RESULTS FOR THE HW AND ID LEAKAGE MODELS ARE SEPARATED BY '/'

Traces	Label	ASCAD_f	ASCAD_r	CHES_CTF
10 000	One-hot	2940/-	-/-	-
	Smoothed	**2 994**/-	**-/-**	**-**
	Distributed	**1 252/4 050**	**1 939/3 753**	**2 182**
10 000	One-hot	2 217/**3 779**	-/-	-
(L2)	Distributed	**1 096**/-	**2 034/4 892**	**1 458**
10 000	One-hot	1 913/-	-/-	-
(Dropout)	Distributed	**1 338/4 219**	-/-	**1 868**
50 000	One-hot	**544/87**	650/**487**	455
	Distributed	779/-	**553**/3 684	**450**
50 000	One-hot	2 829/**4 061**	-/-	-
(Augmented)	Distributed	**1 201**/-	**2 190**/-	**1 724**
100 000	One-hot	2 278/**1 621**	-/-	-
(Augmented)	Distributed	**1 218**/-	**2 298**/-	**2 105**

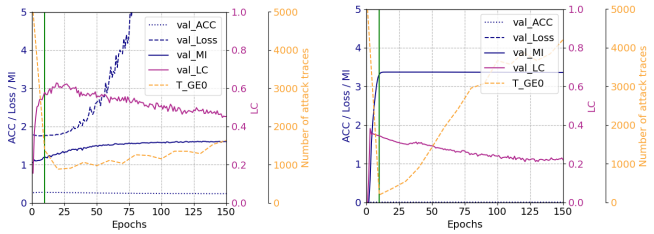
The considered regularization techniques are helpful in some attack settings, improving significantly when combined with distributed labels. Similarly, data augmentation helps obtain better performance in some cases; the combination with distributed labels makes it even better. In practice, due to the limitation of controlled devices and time budget, attackers would likely use smaller networks, more regularization, and more data augmentation to run their attacks in lower-data settings. However, as shown in the table, label distribution is the best technique considering the additional efforts to tune hyperparameters and their performance.

Note that one-hot encoded labels often lead to comparable or superior results when training with 50 000 profiling traces. Indeed, one-hot encoded labels are more precise in discriminating the correct labels than distributed labels. On the other hand, increasing the number of profiling traces amplifies the side-effect of distributed labels, leading to high estimation variance and reduced predictive performance. Finally, although data augmentation could also generate more profiling traces, the difficulty of setting a proper augmentation level makes the generated traces less helpful in the profiling phase.

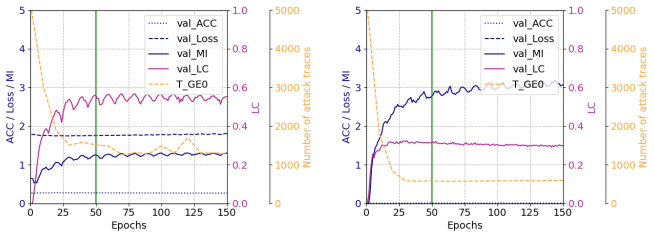
B. Use Cases of Label Correlation

In this section, we investigate the effectiveness of the LC metric for different use cases. Specifically, we consider network architecture search (NAS) and overfitting prevention as they significantly influence the attack performance with DL-based SCA. Indeed, adjusting the profiling model size will directly influence its learning capacity. On the other hand, a correctly set training epoch number could improve the model's fitness to the dataset. Since these two aspects rely on well-performing evaluation metrics [26], [33], we show the performance of LC in various settings and benchmark it with other standard metrics.

1) Early Stopping: As an evaluation metric, LC can be used as early stopping regularization or as an indicator of when to save the best model. For illustration, we evaluate state-of-the-art models by training with different training epochs

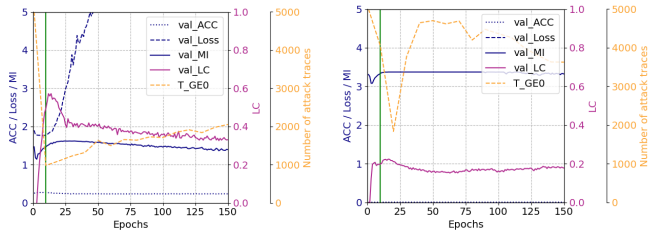


(a) MLP with the HW leakage model. (b) MLP with the ID leakage model.

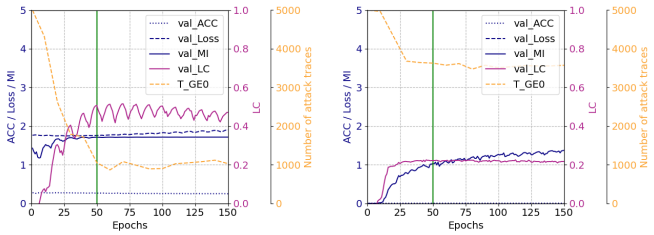


(c) CNN with the HW leakage model. (d) CNN with the ID leakage model.

Fig. 9. Metrics performance on the ASCAD_f dataset.



(a) MLP with the HW leakage model. (b) MLP with the ID leakage model.



(c) CNN with the HW leakage model. (d) CNN with the ID leakage model.

Fig. 10. Metrics performance on the ASCAD_r dataset.

ranging from 1 to 150 in steps of 10, representing the number of iterations to train a profiling model. Aligned with previous sections, the attack performance is assessed by T_{GE0}. Besides, four metrics, accuracy, loss, mutual information (MI) [25], and LC, are calculated per epoch with 5 000 validation traces.[9] One may argue that T_{GE0} can be used as an evaluation metric. However, T_{GE0} can only be calculated when GE equals zero. For a model that cannot break the target with a given number of attack traces, T_{GE0} is not indicative. Similarly, the key rank metric is only meaningful when GE is larger than zero: when the key rank stays zero, one cannot know if the model is still learning or starting overfitting. Since all selected models reached the key rank of zero quickly and never changed, we omit the key rank metric as it is less indicative in the training process.

The results for three datasets and two leakage models are shown in Figures 9, 10, and 11. Since the metrics and T_{GE0} have different scales, multiple Y-axes are used to scale the results data. The optimal training epoch proposed in the literature is marked by green vertical lines (10 for MLPs and 50 for CNNs). Aligned with the previous section, all the presented results are the median from 20 independent pieces of training.

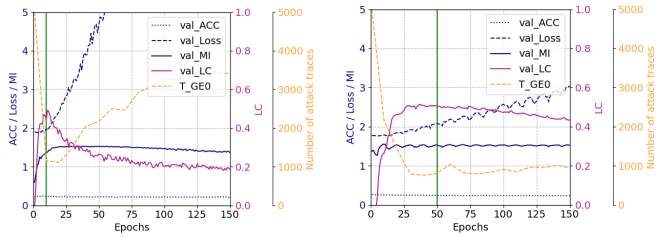
Regarding ASCAD_f, LC perfectly reflects the generalization variation of the profiling model with different training epochs when using T_{GE0} as a reference. From both figures, LC indicates the overfitting effect accurately, or even before the attack performance degrades. Indeed, LC evaluates the order of the key candidates, and the order closer to KD is more likely to be perturbed when overfitting starts. When the overfitting effect accumulates to a certain level, the "disorder" of the key candidate propagates to the correct key, finally captured by GE-related metrics. Due to LC's sensitivity, one can decide on optimal epochs with more patience (i.e., the number of epochs to wait before an early stop if there is no progress on the validation set) without suffering from overfitting.

Regarding other metrics, the MI metric is somewhat misleading as it keeps increasing (e.g., Figures 9a, 9d, and 10d) or does not change much (Figures 10b and 11b) even when T_{GE0} suggests performance degradation. The loss value is only useful in limited cases (e.g., Figure 9a), which confirms the conclusion from Picek et al. [28] that it is commonly not considered a good evaluation metric for SCA, and accuracy remains mostly stable with different training epochs, indicating mediocre performance. Lastly, the literature's optimal training epochs are not optimal for Figures 9a and 9d. On the other hand, LC consistently indicates the epoch that achieves the best attack performance.

Attacks on ASCAD_r and CHES_CTF show consistent results with ASCAD_f. LC performs the best among all evaluated metrics, alarming the overfitting effect precisely. As an evaluation metric, LC combines the advantages of key rank and T_{GE0} with limited computation overhead, thus becoming a reliable metric for the applications such as early stopping.

2) Network Architecture Search: Network architecture search (NAS) is essential in DL-SCA. A smartly designed neural network can break the target and reduce the training complexity as well [33], [42]. To better illustrate the advantage of the LC metric, we use CNN listed in Table III with a tunable α parameter to control the size of the deep learning model. Specifically, α determines the number of filters in convolutional layers and neurons in the fully connected layers. We use α (range from 1 to 64) to estimate the complexity of a profiling model. Note, for the CNN_best from [1], α equals 64. The training epoch is set to be optimal (75) based on [1], represented by the green vertical line in the plot. This section

[9] If GE is greater than zero, T_{GE0}=5 000.



(a) MLP with the HW leakage model. (b) MLP with the ID leakage model.

Fig. 11. Metrics performance on the CHES_CTF dataset.

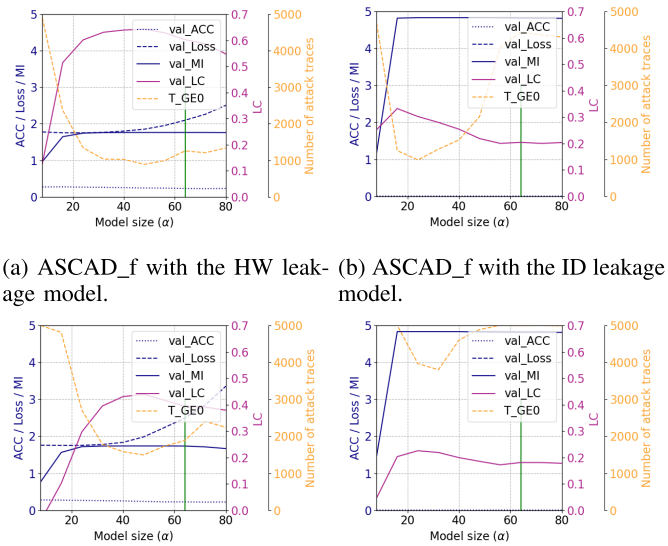
TABLE III
CNN ARCHITECTURE USED FOR THE ATTACK

Layer Types	Filter Size	# of Filters	Pooling Stride	# of Neurons
Conv block	11	a*1	2	-
Conv block	11	a*2	2	-
Conv block	11	a*4	2	-
Conv block	11	a*8	2	-
Flatten	-	-	-	-
Fully connected (2×)	-	-	-	a*64

presents the results for the ASCAD_f and ASCAD_r datasets only. Since CHES_CTF produce similar results, we omit them from this section.

The results are shown in Figure 12. Aligned with the previous section, accuracy, loss, MI, and LC are used as evaluation metrics. As a reference, T_{GE0} represents the attack performance. Among the three considered metrics, LC best represents the attack performance. For instance, in Figure 12a, T_{GE0} reaches minimum when α equals around 50. Further increase of the profiling model size degrades the attack performance, meaning the fitness reduction for a dataset. The LC metric perfectly represents this tendency, as it reaches the maximum when α is around the same model size, then decreases gradually. Regarding other metrics, the validation accuracy has limited changes regardless of the variation of α. Validation loss, in contrast, is more indicative than its counterpart. However, it is challenging to judge when to stop the training. For instance, the loss value in Figure 12c suggests that the profiling should end after training with around 35 epochs, but the best performance is reached 15 epochs later. MI keeps on increasing with the HW leakage model. However, it does not correctly reflect the attack performance. Finally, the training epoch suggested in the literature is still sub-optimal when looking at the results (i.e., Figure 12b). Using LC as an evaluation metric can help monitor the attack performance in various settings.

In addition, we have also tested the influence of the noise on the considered metrics by adding Gaussian noise to the traces with incremental variations ranging from 0 to 10 in a step of 0.5. The results show that the LC metric can correctly and precisely reflect the negative influence introduced by the noise. Since the results align with the conclusions from the previous sections, the results are omitted.



(a) ASCAD_f with the HW leakage model. (b) ASCAD_f with the ID leakage model.

(c) ASCAD_r with the HW leakage model. (d) ASCAD_r with the ID leakage model.

Fig. 12. Metrics performance with different model sizes.

In conclusion, the LC metric reliably reflects the generality of the profiling model in various training conditions. Compared to other metrics, the evaluation of the keys' order helps in increasing the sensitivity of the LC metric in measuring the model's performance. Indeed, in almost all of the experimental results, LC is the first metric that indicates the overfitting effect. Additionally, due to its computation simplicity, we believe LC is an ideal candidate as an evaluation metric.

VII. CONCLUSION AND FUTURE WORK

In the profiling side-channel analysis, one commonly uses intermediate data to form a one-hot encoded label for the profiling. Additionally, it is common to use guessing entropy to estimate the attack performance. This paper introduces distributed labels as a new learning approach that can effectively reduce the required number of profiling traces. Then, based on the relationship between each key candidate, we define the Key distribution (KD) metric and use it to form a novel LC metric. Our results show that the LC metric can be a reliable candidate for evaluating the generality of a model, which has been validated with two use cases: early stopping and network architecture search. Our findings are confirmed for several experiments considering various usage cases, attack methods, leakage models, and datasets.

In future work, we plan to extend the application of label distribution for high-order masked implementations. In terms of the LC metric, since the key distribution relies on the hypothetical distance between key candidates, the distance depends on the algorithm and hardware implementation. Following this, we plan to investigate if the method can be easily adapted to a new implementation or a new algorithm. Moreover, prior knowledge about the leakage model plays a significant role in the proposed label distribution, so we plan to explore LC in the context of leakage assessment for the black-box devices without this knowledge. Finally, applying our results to the non-profiling SCA would be an exciting research direction.

TABLE IV
CNN ARCHITECTURE USED FOR THE ATTACK [33]

Dataset	Leakage Model	Architectures	Learning Rate	Batch Size
ASCAD_f	HW	C(2,25,1), P(4,4), FLAT, FC(15, 10, 4), SM(9)	5e-3	50
	ID	C(128,25,1), P(25,25), FLAT, FC(20, 15), SM(256)	5e-3	50
ASCAD_r	HW	C(4,50,1), P(25,25), FLAT, FC(30, 30, 30), SM(9)	5e-3	128
	ID	C(128,3,1), P(75,75), FLAT, FC(30, 2), SM(256)	5e-3	128
CHES_CTF	HW	C(2,2,1), P(7,7), FLAT, FC(10), SM(9)	5e-3	128

TABLE V
MLP ARCHITECTURE USED FOR THE ATTACK [38]

Dataset	Leakage Model	Architectures	Learning Rate	Batch Size
ASCAD_f	HW	FC(496, 496, 136, 288, 552, 408, 232, 856), SM(9)	5e-4	32
	ID	FC(160, 160, 624, 776, 328, 968), SM(256)	1e-4	32
ASCAD_r	HW	FC(200, 200, 304, 832, 176, 872, 608, 512), SM(9)	5e-4	32
	ID	FC(256, 256, 296, 840, 280, 568, 672), SM(256)	5e-4	32
CHES_CTF	HW	FC(192, 192, 616, 248, 440), SM(9)	1e-3	32

APPENDIX

The used state-of-the-art models are listed in Tables IV and V. All of the non-listed hyperparameter settings are aligned with the original papers [33], [38]. The convolution layer is denoted by C; averaging pooling layer is denoted by P. FLAT and FC denote the flatten layer and fully connected layer, respectively. Finally, SM denotes the output layer with the $softmax$ activation function.

REFERENCES

[1] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 163–188, Jun. 2020.

[2] E. Bursztein, "Leveraging deep-learning to perform SCA attacks against AES implementations," in *Proc. Cryptograph. Hardw. Embedded Syst. (CHES)*, 2018.

[3] IC Card. (Nov. 2021). *EMV Integrated Circuit Card Specifications for Payment Systems, Book 3 Application Specification*. [Online]. Available: https://www.emvco.com/wp-content/uploads/2017/04/EMVv4.3Book3ApplicationSpecification20120607062110791.pdf

[4] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.* W. Fischer and N. Homma, Eds. Cham, Switzerland: Springer, 2017, pp. 45–68.

[5] O. Choudary and M. G. Kuhn, "Template attacks on different devices," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Paris, France: Springer, Apr. 2014, pp. 179–198.

[6] M. O. Choudary and M. G. Kuhn, "Efficient, portable template attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 2, pp. 490–501, Feb. 2018.

[7] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 2523. Cham, Switzerland: Springer, Aug. 2002, pp. 13–28.

[8] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert, "Univariate side channel attacks and leakage modeling," *J. Cryptograph. Eng.*, vol. 1, no. 2, pp. 123–144, Aug. 2011.

[9] Y. Fei, Q. Luo, and A. A. Ding, "A statistical model for DPA with novel algorithmic confusion analysis," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2012, pp. 233–250.

[10] X. Geng, "Label distribution learning," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1734–1748, Jul. 2016.

[11] R. Gilmore, N. Hanley, and M. O'Neill, "Neural network based attack on a masked implementation of AES," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 106–111.

[12] X. Geng, Q. Wang, and Y. Xia, "Facial age estimation by adaptive label distribution learning," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Aug. 2014, pp. 4465–4470.

[13] G. Hospodar, B. Gierlichs, E. D. Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: A first study," *J. Cryptograph. Eng.*, vol. 1, no. 4, pp. 293–302, Dec. 2011.

[14] J. Hauke and T. Kossowski, "Comparison of values of Pearson's and spearman's correlation coefficients on the same sets of data," *Quaestiones Geographicae*, vol. 30, no. 2, p. 87, 2011.

[15] A. Heuser and M. Zohner, "Intelligent machine homicide–breaking cryptographic devices using support vector machines," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design* W. Schindler and S. A. Huss, Eds., vol. 7275. Cham, Switzerland: Springer, 2012, pp. 249–264.

[16] P. C. Kocher, J. Jaffe, and B. Jun., "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 1666, M. J. Wiener, Ed. Santa Barbara, CA, USA: Springer, 1999, pp. 388–397.

[17] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, 2019.

[18] Y. K. Kumar and R. M. Shafi, "An efficient and secure data storage in cloud computing using modified RSA public key cryptosystem," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 10, no. 1, p. 530, Feb. 2020.

[19] L. Lerman, S. F. Medeiros, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," in *Proc. CARDIS*, in Lecture Notes in Computer Science. Berlin, Germany: Springer, Nov. 2013, pp. 123–139.

[20] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert, "Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis)," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Cham, Switzerland: Springer, 2015, pp. 20–33.

[21] X. Lu, C. Zhang, P. Cao, D. Gu, and H. Lu, "Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 235–274, Jul. 2021.

[22] L. Masure, C. Dumas, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 348–375, Nov. 2019.

[23] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Proc. Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng.* Cham, Switzerland: Springer, 2016, pp. 3–26.

[24] C. ObFlynn and Z. D. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*. Cham, Switzerland: Springer, 2014, pp. 243–260.

[25] G. Perin, I. Buhan, and S. Picek, "Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*, vol. 12910. Cham, Switzerland: Springer, 2021, pp. 53–81.

[26] G. Perin, Ł. Chmielewski, and S. Picek, "Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 337–364, Aug. 2020.

[27] S. Picek et al., "Side-channel analysis and machine learning: A practical perspective," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 4095–4102.

[28] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, pp. 209–237, Nov. 2018.

[29] S. Picek, A. Heuser, G. Perin, and S. Guilley, "Profiled side-channel analysis in the efficient attacker framework," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Cham, Switzerland: Springer, 2022, pp. 44–63.

[30] R. Poussier, F.-X. Standaert, and V. Grosso, "Simple key enumeration (and rank estimation) using histograms: An integrated approach," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2016, pp. 61–81.

[31] G. Perin, L. Wu, and S. Picek, "Exploring feature selection scenarios for deep learning-based side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2022, pp. 828–861, Aug. 2022.

[32] A. Rădulescu, P. G. Popescu, and M. O. Choudary, "GE vs GM: Efficient side-channel security evaluations on full cryptographic keys," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2022, pp. 886–905, Aug. 2022.

[33] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 677–707, Jul. 2021.

[34] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* J. R. Rao and B. Sunar, Eds. Berlin, Germany: Springer, 2005, pp. 30–46.

[35] F.-X. Standaert, G. Tal Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, A. Joux, Ed. Berlin, Germany: Springer, 2009, pp. 443–461.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[37] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 147–168, Jun. 2020.

[38] L. Wu, G. Perin, and S. Picek, "I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis," *IEEE Trans. Emerg. Topics Comput.*, early access, Nov. 7, 2022, doi: 10.1109/TETC.2022.3218372.

[39] L. Wu, G. Perin, and S. Picek, "The best of two worlds: Deep learning-assisted template attack," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2022, pp. 413–437, Jun. 2022.

[40] L. Wu, G. Perin, and S. Picek, "On the evaluation of deep learning-based side-channel analysis," in *Proc. International Workshop Constructive Side-Channel Anal. Secure Design*, vol. 13211. Leuven, Belgium: Springer, p. 49.

[41] D. Xue et al., "Personality recognition on social media with label distribution learning," *IEEE Access*, vol. 5, pp. 13478–13488, 2017.

[42] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 1–36, Nov. 2019.