Ablation Analysis for Multi-device Deep Learning-based Physical Side-channel Analysis

Lichao Wu[1], Yoo-Seung Won[2], Dirmanto Jap[2], Guilherme Perin[3, 1], Shivam Bhasin[2], Stjepan Picek[3, 1]
[1]Delft University of Technology,
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands.
[2]Temasek Laboratories, Nanyang Technological University,
50 Nanyang Drive, Research Techno Plaza, BorderX Block, 9th Storey, Singapore 637553
[3]Faculty of Science, Radboud University,
Postbus 9010, 6500 GL Nijmegen, Netherlands.
Email: l.wu-4@tudelft.nl, yooseung.won@ntu.edu.sg djap@ntu.edu.sg, guilherme.perin@tudelft.nl,
sbhasin@ntu.edu.sg, stjepan.picek@ru.nl

***Abstract*—The use of deep learning-based side-channel analysis is an effective way of performing profiling attacks on power and electromagnetic leakages, even against targets protected with countermeasures. While many research papers have reported successful results, they typically focus on profiling and attacking a single device, assuming that leakages are similar between devices of the same type. However, this assumption is not always realistic due to variations in hardware and measurement setups, creating what is known as the portability problem. Profiling multiple devices has been proposed as a solution, but obtaining access to these devices may pose a challenge for attackers.**

This paper proposes a new approach to overcome the portability problem by introducing a neural network layer assessment methodology based on the ablation paradigm. This methodology evaluates the sensitivity and resilience of each layer, providing valuable knowledge to create a Multiple Device Model from Single Device (MDMSD). Specifically, it involves ablating a specific neural network section and performing recovery training. As a result, the profiling model, trained initially on a single device, can be generalized to leakage traces measured from various devices. By addressing the portability problem through a single device, practical side-channel attacks could be more accessible and effective for attackers.

***Index Terms*—Side-channel Analysis, Deep learning, Ablation, Portability.**

I. Introduction

The demand for certified products has grown due to the rising embedded device market and their ever-increasing security concerns and vulnerabilities. This leads to thousands of products undergoing strict security evaluations in evaluation laboratories worldwide on a daily basis [1]. Side-channel analysis (SCA [2]) is one such threat against which embedded devices are regularly evaluated. While a range of attacks can be mounted on an embedded device, profiling SCA [3] remains highly relevant as it provides the worst-case guarantees under a supervised learning paradigm.

In just a few years, deep learning has emerged as the preferred option for profiling side-channel analysis, as evidenced by numerous successful results, even against targets protected with state-of-the-art hiding and masking countermeasures [4]–[6]. Recently, researchers have continued this trend by finding smaller and shallower neural networks that perform well for specific datasets [7], [8]. However, all of these works focus on a single device, assuming that the leakage traces measured from different devices are identical or follow similar distributions. We argue that this assumption is unrealistic, as the leakage traces can vary significantly due to factors such as hardware manufacture, fabrication, and variation in the measurement setup. This variability presents a significant challenge for side-channel analysis, known as the "portability problem". While the Multiple Device Model (MDM) has been proposed as a practical solution to this problem [9] (i.e., training and validating on multiple copies of the training device rather than just one), the availability of multiple devices remains a significant constraint. The availability of multiple devices is a scoring criterion in common criteria evaluations [10]. A worst-case adversary assumes the availability of multiple copies of the device, which makes the MDM approach less practical in real-world scenarios.

This paper aims to address the portability issue in side-channel analysis without relying on multiple device assumptions while achieving comparable performance as Multiple Device Model (MDM). We aim to perform a worst-case analysis to mitigate potential risks. To achieve this goal, we propose a layer assessment methodology that provides an in-depth understanding of how neural networks function. Specifically, we introduce the ablation procedure that involves disabling specific parts of a neural network and observing the impact on its performance. By comparing the results before and after ablation, we can identify the sensitivity of layers where crucial side-channel analysis information processing occurs. After that, we perform recovery training on the ablated neural network so that the resilience of the ablated layer can also be assessed.

Using the insight of layers-wise behavior, we propose the *Multiple Device Model from Single Device* (MDMSD) approach. First, we partially "damage" a selected layer of the model (defined during layer assessment) trained on the original device with ablation. This step reduces the model's overfitting to the original device while retaining most of its predictive capability. We then hypothesize that portability can be seen as additive Gaussian noise [9], and we conduct recovery training on the ablated model with perturbed leakages (with Gaussian noise) from the original device to simulate the portability effect. The resulting model can generalize to various devices without relying on multi-device assumptions. By adopting this methodology, we aim to eliminate the portability problem and make side-channel analysis more accessible and effective in practical scenarios. Our main contributions are:

1) We propose a new methodology for conducting ablation analysis in profiling SCA that allows us to assess the importance of each layer in the neural network.
2) We introduce two new layer assessment criteria: sensitivity and resilience. Layer sensitivity measures the importance of a particular layer for achieving high attack performance. Layer resilience represents the necessity of a specific layer for the overall neural network.
3) We demonstrate that even smaller neural networks than those commonly used in SCA can achieve top performance, indicating room for further improvement in network design methodologies.
4) We apply our layer assessment methodology to address the portability problem in SCA, where the training device and the device under attack differ. Our proposed approach, Multiple Device Model from Single Device (MDMSD), shows promising results in overcoming this challenge.

To demonstrate the effectiveness of our approach, we provide extensive experimental analysis with two neural network types and four datasets. The source code is available in Github.[1]

II. Background

A. Notation

Calligraphic letters like $\mathcal{X}$ denote sets, and the corresponding upper-case letters X 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. k represents a key byte candidate taking its value from the keyspace $\mathcal{K}$, and k^* represents the correct key byte.

A dataset is a collection of traces (side-channel measurements) $\mathbf{T}$, where each trace $\mathbf{t}_i$ is associated with an input value (plaintext or ciphertext) $\mathbf{d}_i$ and a key $\mathbf{k}_i$. The vector of parameters to be learned in a profiling model (e.g., the weights in neural networks) is denoted by $\boldsymbol{\theta}$.

B. Deep Learning and Profiling SCA

Deep learning represents machine learning methods based on artificial neural networks with representation learning. Supervised machine (deep) learning involves learning a function f that maps an input to the output ($f : \mathcal{X} \rightarrow Y$) based on examples of input-output pairs. The function f is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$, where n denotes the number of trainable parameters.

Supervised learning happens in two phases: training and testing. This corresponds to profiling SCA, executed in profiling and attack phases. In the rest of this paper, we use the terms profiling/training and attack/test interchangeably.

1) The goal of the training phase is to learn the parameters $\boldsymbol{\theta}$ that minimize the empirical risk represented by a loss function on a dataset $\mathbf{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ of size N. As common in profiling SCA, we consider the c classification task, where c denotes the number of classes depending on the used leakage model. Thus, the classifier f is a function mapping input features to label space ($f : \mathcal{X} \rightarrow \mathbb{R}^c$). In the rest of this paper, the function f is a deep neural network with the *Softmax* output layer.
2) In the attack phase, the goal is to make predictions about the classes
$$y(\mathbf{x}_1, k^*), \ldots, y(\mathbf{x}_Q, k^*),$$
where k^* represents the secret (unknown) key on the device under the attack. The outcome of prediction with a model f on the attack set is a two-dimensional matrix P with dimensions equal to $Q \times c$. The cumulative sum $S(k)$ for any key byte candidate k is a common SCA distinguisher (tool to distinguish among different hypotheses):
$$S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,y}). \tag{1}$$
The value $\mathbf{p}_{i,y}$ denotes the probability that a predicted input $\mathbf{x}_i$ is represented by class y. The class y is derived from the key k and input d_i through a cryptographic function (e.g., XOR operation between k can d_i in the case of the AES cipher) and a leakage model (the function that converts hypothetical values into physical leakage of a device).

To assess the attack performance, i.e., the number of measurements required to break a target, it is common to use the guessing entropy (GE) metric [11]. With Q traces in the attack phase, the attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{\|\mathcal{K}\|}]$ in decreasing order of probability. Thus, g_1 is the most likely and $g_{\|\mathcal{K}\|}$ the least likely key candidate. Guessing entropy is the average position of the correct key k^* in $\mathbf{g}$. In this work, we calculate partial guessing entropy (i.e., we consider a specific key byte) but denote it as guessing entropy for simplicity.

C. Leakage Models

In the context of deep learning-based SCA, there are two commonly considered leakage models:

1) Hamming weight (HW): where the attacker assumes the leakage proportional to the sensitive variable's Hamming weight. This leakage model results in nine classes for the AES cipher (8-bit S-box[2]).
2) Identity (ID): where the attacker considers the leakage as an intermediate value of the cipher. When considering the AES cipher (8-bit S-box), this leakage model results in 256 classes.

[1]https://github.com/lichao-wu9/Ablation-Study

[2]Or any cipher with 8-bit S-box.

D. Portability

The strength of profiling SCAs arises from their capability to characterize the target device fully. In literature, most works conduct profiling and testing on the same device ("single-device" model). However, in practical scenarios, external factors, such as process variation or different acquisition methods, may cause the "single-device-model" attack to fail. These external factors lead to an issue known as portability, subsuming all effects due to different devices and secret information between profiling device and device under attack. Though this is a critical issue and evaluation labs face it daily, few investigations are conducted in this direction.

One of the latest approaches to address portability was presented by Bhasin et al. [9] by considering the Multiple Device Model (MDM). The idea is to use multiple copies of devices similar to the targeted one for training and validation. Then, the trained model could better generalize the leakage and minimize the risk of overfitting to the training device. However, one possible drawback is that the evaluator needs to acquire measurements from multiple copies of the same device, which might be either expensive (in time or equipment) or simply unavailable.

E. Neural Network Architectures

This paper considers two neural network types commonly used in profiling SCA [5], [12]:

1) **Multilayer Perceptron**. The multilayer perceptron (MLP) is a feed-forward neural network mapping sets of inputs onto sets of appropriate outputs. MLP consists of multiple layers (input layer, output layer, and at least one hidden layer) of nodes in a directed graph, with each layer fully connected to the next one.
2) **Convolutional Neural Networks**. Convolutional neural networks (CNNs) commonly consist of three types of layers: convolutional layers, pooling layers, and fully connected layers. The convolutional layer computes the output of neurons connected to local regions in the input, each computing a dot product between their weights and a small input region. Pooling decreases the number of extracted features by performing a down-sampling operation along the spatial dimensions. The fully connected layer computes either the hidden activations or the class scores.

F. Related Work

The domain of profiling SCA started in 2002 with the template attack [3]. While this attack is the most powerful one from the information-theoretic perspective, in practice, it suffers from restrictive assumptions (unlimited number of profiling traces, noise following Gaussian distribution) [13]. The SCA researchers also considered simple machine learning techniques, e.g., random forest [14], support vector machines [15], [16], Naive Bayes [17], and multilayer perceptron [18]. Such techniques commonly performed similarly or better than the template attack.

Since 2016, the SCA community has shifted much of its attention to deep learning techniques [4]. The two most explored approaches were MLP (commonly, more complex architectures than before) and CNN. Both approaches reached excellent attack performance where it is possible to break implementations protected with countermeasures [12], [19]. Only recently, the community expanded the deep learning perspective for profiling SCA, e.g., autoencoders used to preprocess the traces to remove the influence of countermeasures [6] or conduct feature engineering [20].

As deep learning in SCA is a relatively new research direction (compared to other domains), most works still concentrate on improving the attack performance. Indeed, deep learning methods developed in the early stage showed potential but not much more than "simple" machine learning [4], [21]. Soon after, researchers reported strong attack performance even in the presence of countermeasures [12], [19]. More recent results improved the performance with reduced sizes of deep learning architectures [5]–[7], [22]. The latest result show how deep learning-based SCA can be even more powerful, but considering raw measurements instead of selected intervals of features [23], [24].

While far from completed (as the results can be improved even further), deep learning represents a significant step forward for profiling SCA. Simultaneously, the SCA community's understanding of what happens during the learning process (i.e., interpretability and explainability) is much more limited. Several works considered visualization techniques to find relevant features and improve interpretability [25], [26]. Van der Valk et al. considered the activation functions in neural networks to explain what neural networks learn while training on different side-channel datasets or even datasets that are not side-channel measurements [27]. Additionally, some researchers have already noticed the importance of ablation in the context of SCA. For example, in the conclusion of [5], the authors stated the importance of performing an ablation study when a new technique is proposed, which is then realized by [7] with a manual (instead of automated) ablation study of the proposed methodology. Still, no discussions aim at ablation as a tool for explainability in SCA. Finally, some more recent works provided new insights into the interpretability and explainability of neural networks in SCA. Yap et al. used an interpretable neural network called Truth Table Deep Convolutional Neural Network (TT-DCNN) that allows easy transformation into SAT equations to obtain the rules and decisions the neural networks learned when retrieving the secret key from the cryptographic primitive [28]. Zaid et al. developed a generative model, designed from the stochastic attacks, with the goal of reducing the black-box property of deep learning and easier architecture design for real-world crypto-system [29]. Perin et al. proposed a novel methodology for deep learning explainability in SCA where the authors consider every layer and measure the perceived information from it [30].

For portability, already early work [31] showed differences in leakage obtained from four different Atmel XMEGA. Another work [32] has proposed deep learning optimized for a cross-device attack, which was trained on multiple devices running a 128-bit AES encryption module. Specifically, the author used four devices for training a multilayer perceptron,

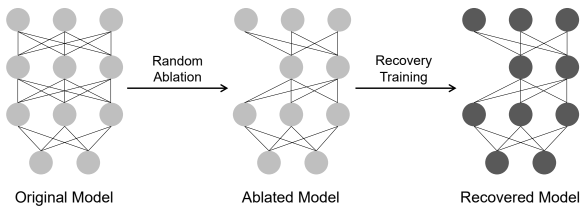


Fig. 1: A demonstration of the proposed ablation method.

resulting in an average accuracy of 99.9% with all devices. In [33], the results for the CHES 2018 CTF dataset show that when targeting different devices, the attack will require at least five traces, while for the same device, it only requires one trace. Although the performance differences are limited, it indicates the existence of the portability challenge even in an optimal attack setting. In [34], the authors addressed the portability issue by proposing a similarity assessment technique for quantifying the differences among various copies of the same device. Bhasin et al. [9] considered the Multiple Device Model, validated by [32], [33]. On the other hand, [34] adopted an alternative approach to measure the difference between devices and exploit that to improve profiling SCA. Later, Zhang et al. [35] investigated the portability issue for heterogeneous device scenarios, analyzing the effect caused by device variation. In [36], the authors adopted meta-transfer learning to transfer and adapt trained neural networks to other devices, even using different side-channel sources. Finally, recent work summarizes various challenges in deep learning-based SCA [37]. We refer interested readers to it.

III. Ablation & Protability

A. Ablation

Ablation is a process long used in neuroscience, where controlled damages are introduced in neural tissue to investigate the impact of injuries on the brain's capabilities to perform assigned tasks. This approach provides deep insights and explanations about each part of the tissue's structure and role when reacting to external stimuli [38]. As the complexity of artificial neural networks increases, the explainability of models has become an open question. As a natural extension, an ablation study removes specific components to understand their contribution to the system [39]. Ablation requires that the system shows slow degradation, i.e., that the system continues to work even when specific components are missing or reduced, so that the contribution of the removed part can be easily assessed by comparing with the original version. A graphical depiction of the SCA ablation process is shown in Figure 1.

There is a connection between ablation and a technique called pruning, corresponding to the systematic removal of parameters from an existing system [40]. However, unlike ablation, which removes a part of the neural network directly, pruning is commonly performed based on the magnitude of the weights - neurons with weights under a threshold value are disabled. Besides, the underlying idea between ablation and pruning is different. Pruning is commonly used to speed up inference/prediction while minimizing the impact on the network's performance. On the other hand, ablation reduces trainable parameters to gain insights and explain the trained network's inner workings. The training speed is also increased as a consequence of a smaller model. As we are interested in improving the portability of a profiling model on different devices, it is crucial to understand how neural networks (over)fit on datasets. We argue that ablation is a proper technique for this objective. We emphasize that there are no widely accepted techniques for AI explainability, but ablation represents a viable choice [41]. Moreover, to our knowledge, no research provides theoretical results for explainability that can also be used in the SCA practice.

B. Layer Sensitivity Assessment

Although ablation can be performed in a neuron/convolution filter manner, we argue that each neuron/convolution filter's contribution can fluctuate due to the random weight initialization. As a result, it is challenging to reach a consistent conclusion when one repeats the proposed ablation methodology with a different pre-trained model. Therefore, we perform the ablation study layer-wise to reach a stable performance.

Algorithm 1 represents our layer assessment methodology and is repeated for every layer. First, the original model under evaluation is trained with a specific dataset. We consider MLP and CNN architectures as profiling models, but our approach is architecture-agnostic. Once training is finished, a layer's neurons/convolution filters are randomly ablated with ablation rate ρ. Next, M_ρ is trained for τ epochs (Line 8). We denote this process as the *recovery training*. By comparing the performance before and after ablation and recovery training, we can understand the properties of each neural network layer. Specifically, we define two criteria to evaluate a layer: sensitivity and resilience.

- Sensitivity: A layer is considered sensitive if the model has a significant performance drop after ablation.
- Resilience: A layer is considered resilient if the model has a limited performance drop with recovery training after ablation.

The sensitivity and resilience of a layer are calculated in Line 7 and Line 10 of Algorithm 1. The ablation recovery training process is repeated σ to cover most elements in a specific layer. The results are averaged to generate representative results for a specific network layer l.

Since Algorithm 1 is performed per layer, more layers lead to higher time consumption (as we repeat the procedure layer-wise). Fortunately, the recovery training is time-efficient due to the small number of the required training epochs to adjust the model. Although this is more computationally expensive than calculating GE for the original model only, the knowledge obtained through ablation could lead to understanding the model and is helpful for future model adjustments.

C. Multiple Device Model from Single Device

The ablation methodology proposed in the previous section defines the sensitivity and resilience of each layer. We now

Algorithm 1 Layer Sensitivity Assessment.

```
1: procedure ABLATE_LAYER(model $M$, training set $\mathbf{T}_{train}$, test set $\mathbf{T}_{test}$, repeat time $\sigma$, ablation rate $\rho$)
2:     $M \leftarrow \mathsf{train}(M, \mathbf{T}_{train})$
3:     $GE \leftarrow \mathsf{attack}(M, \mathbf{T}_{test})$
4:     for $i$ = 1 to $\sigma$ do
5:         $M_\rho \leftarrow \mathsf{ablate}(M, \rho)$
6:         $GE_\rho \leftarrow \mathsf{attack}(M_\rho, \mathbf{T}_{test})$
7:         $Sensi_i$ = $GE$ - $GE_\rho$                         ▷ Layer sensitivity
8:         $M'_\rho \leftarrow \mathsf{train}(M_\rho, \mathbf{T}_{train})$          ▷ Recovery training
9:         $GE'_\rho \leftarrow \mathsf{attack}(M'_\rho, \mathbf{T}_{test})$
10:        $Resi_i$ = $GE$ - $GE'_\rho$                         ▷ Layer resilience
11:    end for
12: end procedure
```

tackle the portability problem for the profiling SCA. As mentioned, the main challenge of the portability problem is that the side-channel leakages from different devices could vary significantly, leading to a low generality of using a profiling model trained on one device to attack other devices. Let us assume two devices A and B, with leakage measurements $\mathbf{T}_A$ and $\mathbf{T}_B$ focusing on the same cryptographic operation. Then, $\mathbf{T}_B$ can be seen as a skewed version of $\mathbf{T}_A$, modeled by a function skew:

$$\mathbf{T}_A = \mathsf{skew}(\mathbf{T}_B). \tag{2}$$

In practice, the skewness of the leakage traces mainly comes from the variation of devices and measurement setups. Following the assumption that the portability can be seen as additive Gaussian noise [9], we rewrite Equation 2 as:

$$\mathbf{T}_a = \mathbf{T}_b + noise, \tag{3}$$

where $noise$ represents the Gaussian noise. From the model training perspective, the additional noise increases the difficulties in breaking the target.

As mentioned, the part of the neural network sensitive to noise could be the main obstacle to the model's portability capability. In other words, if a profiling model overfits training traces, the sensitive part of the neural network could be the main contributor. Any minor change in the input leakage traces would significantly vary the output performance. Therefore, the goal is to remove this part and let the remaining portion of the network fit on new leakage traces, as we expect the new profiling model can better fit the leakage traces than the one without ablation. At the same time, removing these parts should not cause irreparable damage to the model. Indeed, recall the layer assessment criteria defined in the previous section. The layer that is sensitive (i.e., contributes to the model overfitting) and resilient (i.e., the damage in this layer can be recovered, thus does not influence the model's attack capability) is the ideal objective for ablation.

The Multiple Device Model from Single Device (MDMSD) method is shown in Algorithm 2. The basic procedure is similar to Algorithm 1. First, we partially ablate the model trained on the original device, which forces the model to less overfit the original device measurements while keeping most of the predicting capability. Then, the ablated model is recovery trained and tested with perturbed leakages from the original device to simulate the portability effect. Then, the new model can generalize to a range of devices.

Specifically, the adversary collects the traces for training and testing based on the original device o, denoted as $\mathbf{T}^o_{train}$ and $\mathbf{T}^o_{test}$, respectively. The original model, M, is first trained for τ_o epochs with $\mathbf{T}^o_{train}$ on this device. The GE for the pre-trained model is then computed based on model M and $\mathbf{T}^o_{test}$ dataset with additional noise α (representing the measurement noise). The adversary then ablates M with a rate ρ and conducts the recovery training for τ_r epochs to obtain the new ablated model M'_ρ. The attack performance is assessed with the recovery trained model M'_ρ and dataset $\mathbf{T}^o_{test} + noise(\beta)$ (representing the portability-induced noise). Next, the adversary defines the threshold margin m. If the condition $GE'_\rho \leq (m \cdot GE)$ is satisfied, we stop the Algorithm 2 and obtain the final GE from $\mathbf{T}^v_{test}$ dataset of the victim device. Note that this condition measures the resistance of a layer. If two layers have the same resilience, we select the one with higher sensitivity.

Algorithm 2 Methodology for MDMSD.

```
1: procedure MDMSD(The original device $o$ with training, test dataset $\mathbf{T}^o_{train}, \mathbf{T}^o_{test}$, Victim device $v$ with test dataset $\mathbf{T}^v_{test}$, threshold margin $m$, Noise value for training and test $\alpha, \beta$, Ablation rate $\rho$)
2:     $M \leftarrow \mathsf{train}(M, \mathbf{T}^o_{train})$
3:     $GE \leftarrow \mathsf{attack}(M, \mathbf{T}^o_{test} + noise(\alpha))$
4:     while $GE'_\rho > (m \cdot GE)$ do
5:         $M_\rho \leftarrow \mathsf{ablate}(M, \rho)$
6:         $M'_\rho \leftarrow \mathsf{train}(M_\rho, \mathbf{T}^o_{train} + noise(\alpha))$
7:         $GE_\rho \leftarrow \mathsf{attack}(M_\rho, \mathbf{T}^o_{test} + noise(\beta))$
8:         $GE'_\rho \leftarrow \mathsf{attack}(M'_\rho, \mathbf{T}^o_{test} + noise(\beta))$
9:     end while
10:    $GE^v_\rho \leftarrow \mathsf{attack}(M'_\rho, \mathbf{T}^v_{test})$
11: end procedure
```

Noise parameters α and β must be chosen carefully to better represent noise from portability. If α and β take a similar value and are too small, the resulting GE and GE'_ρ will be too similar and will not address the portability issue; an overly large α would fail even with ablation and recovery training. Therefore, we use relatively small α and then conduct ablation to fight more significant portability-induced noise $noise(\beta)$. In terms of margin m, ablation can lead to cases where GE'_ρ could be slightly higher than GE. To counter such scenarios, we empirically set a 5% leverage to GE'_ρ, thus $m = 1.05$.

IV. Experimental Setup

A. Threat Model

We consider a common profiling side-channel setting focusing on power/EM side-channel attacks targeting secret key recovery from cryptographic algorithms. This threat model is standard and realistic as numerous certification laboratories evaluate hundreds of security-critical products under this model daily. Power/EM side-channel is often exploited for exploiting modern communication devices [42] or even used for program flow tracking [43].

We assume an adversary with access to a clone device running the target cryptographic algorithm, normally on an embedded device. This clone device can be queried with known/chosen parameters (keys, plaintext, etc.) while the corresponding leakage measurements, like power or electromagnetic emanation, are recorded. A profiling model is built to map the relationship between the leakages and the key-related intermediate data. This constitutes the profiling phase.

Next, the adversary queries the device under attack with known plaintext to recover the secret key by querying the characterized model with corresponding side-channel leakage traces. This represents the attack phase.

B. General Settings

We provide results with several ablation levels to investigate the behavior of neural networks for various settings (i.e., when we do a small change, medium change, or a significant change to the neural network architecture). Based on our experiments, the ablated model does not require significant training to adapt to the changes (as the models are pre-trained). Therefore, we run the recovery training for ten epochs. GE is calculated over ten attacks with a random shuffling of the attack traces to obtain statistically significant results. Finally, GE and weight variation presented in the experiments are averaged over ten independent ablation experiments for each layer. All experiments are implemented with the TensorFlow [44] computing framework and Keras deep learning framework [45]. The model's training was executed on an Nvidia GTX 1080 graphics processing unit (GPU), managed by Slurm workload manager version 19.05.4.

C. Datasets

We first consider two popular datasets widely adopted in SCA research: ASCAD with the fixed key (ASCAD_F) and ASCAD with random keys (ASCAD_R).[3] The measurements are obtained from an 8-bit AVR microcontroller running an AES-128 implementation [46]. Both datasets are protected with a Boolean masking countermeasure.

ASCAD_F This version of the ASCAD dataset has 50 000 traces for profiling and 10 000 traces for the attack. 5 000 traces from the profiling set are used for validation. We use a pre-selected window of 700 features for the side-channel trace, and we attack key byte 3, the first masked key byte (as recommended by the authors of the dataset).

ASCAD_R. The second ASCAD version has random keys, and the dataset consists of 200 000 traces for profiling and 100 000 traces for the attack. We use 5 000 traces from the attack set for validation. We use a pre-selected window of 1 400 features for this dataset and attack key byte 3 (the first masked key byte).
In addition, two portability-specific datasets are considered to demonstrate the application of the ablation in tackling portability issues for profiling SCA. Table I summarizes the detailed setup for these datasets.

Portability_2020. This dataset was introduced in [9]. The dataset contains measurements from four copies of the target, AVR Atmega328p 8-bit microcontroller, set up in parallel. It measures 50 000 power side-channel traces corresponding to 50 000 random plaintexts. A trace comprises of 600 sample points (features), containing only the execution of the first SubBytes operation of an unprotected AES-128. The dataset was then collected based on the measurements from four boards (B1, B2, B3, B4) with three randomly chosen secret fixed keys (K1, K2, K3).

CHESCTF_2018. This dataset refers to the CHES Capture-the-flag (CTF) AES-128 trace set running on an STM32 microcontroller, released in 2018 [47]. It consists of different sets of power traces of masked AES-128, with 650 000 sample points per trace. In this paper, we focus on a window of 600 points representing the leakages of the target execution. The first four sets contained 10 000 power traces. The first three sets (Set 1 to 3) were collected from three devices (denoted A, B, and C), and each trace corresponds to encryption with a randomly chosen key. Set 4 contains power traces from Device C with a single fixed key (K4). Set 5 contains 1 000 power traces collected from device C with a fixed key K5, and Set 6 contains 1 000 power traces collected from a new device D with a fixed key K6.

Dataset	Device	Key Type	Key	Notation
Portability_2020 [9]	B1	Fix	K1	B1_K1
	B2	Fix	K2	B2_K2
	B3	Fix	K1	B3_K1
	B4	Fix	K3	B4_K3
CHESCTF_2018 [47]	Device A	Random	-	A_RN
	Device B	Random	-	B_RN
	Device C	Random	-	C_RN
	Device C	Fix	K4	C_K4
	Device C	Fix	K5	C_K5
	Device D	Fix	K6	D_K6

TABLE I: The target datasets for portability settings.

D. Neural Network Architectures

In Table II, we depict the neural network hyperparameters selected after a tuning phase. Here, modified MLP and CNN [46] are used for evaluation. The architectures can be easily tuned based on specific requirements. The input layer is adapted based on the dataset tested; the output layer is adjusted based on the used leakage model. For CNN models, the size of the convolution filters is set to 11. An average pooling layer follows each convolutional layer with both pooling size and stride set to two. The experiments are conducted under the widely-used Hamming Weight (HW) leakage model for a key byte. The ID leakage model results are omitted due to similar observations and conclusions.

V. Experimental Results for Layer Assessment

Recall that in Algorithm 1, a model M is pre-trained. Here, we denote M as the original model, as the following analysis is based on this model. To assess the sensitivity and resilience of a layer, the attack performance before and after the recovery training are considered and presented in the GE difference plots (e.g., Figure 2). As defined in Algorithm 1, we use the following notations in the figure:

[3]https://github.com/ANSSI-FR/ASCAD

Network	Leakage Model	Architecture	Learning Rate	Epochs	Batch Size
MLP	HW ID	Dense(200)*8	1e-4 3e-5	100 200	100
CNN	HW ID	Conv(64,128,256,512,512) + Dense(1 024)*2	1e-4	75	200

TABLE II: Baseline deep learning architectures. The width of the output layer depends on the leakage model.

- Sensitivity: $GE_{\rho=0} - GE_{\rho=\rho}$, denoting the GE difference between the original and ablated model *before recovery training*.
- Resilience: $GE_{\rho=0} - GE'_{\rho=\rho}$, denoting the GE difference between the original and ablated models *after recovery training*.

When the GE difference is below zero, the ablation (or ablation with recovery training) of the neural network introduces negative effects when compared to the performance of the original model M. When positive, the new model after ablation/recovery training performs better than the original one.

A. Results for the ASCAD with the Fixed Key Dataset

Figure 2 presents the sensitivity and resistance of each neural network layer. Note that both models (MLP and CNN) break the target with a given number of attack traces. When ablating layers for MLP architectures, more significant changes are caused in the first layers. This tendency becomes more evident when the ablation percentage ρ becomes larger. Note that while it seems there are significant GE changes in the beginning layers for $\rho = 10\%$, the scale is different, so the changes are limited. Thus, a designer who wants to optimize these MLP architectures (i.e., reduce their size) should start by tuning the neurons in the final layers of MLP (as they are less sensitive). Meanwhile, increasing the capacity of shallower layers (by adding more neurons/layers) would increase the robustness of the model. For CNN, when increasing the ablation rate, similar to MLP models, deeper layers are less sensitive to the ablation on average; both layers are resilient to ablation after the recovery training. Indeed, due to the high complexity of the model, the side effect of $\rho = 90\%$ ablation in one layer can be easily compensated by recovery training. Interestingly, as shown in Figure 2d, GE can be slightly better (0.01) when ablating the deeper layers. This is because models with extra capacity would learn from the noise easily, finally causing overfitting. The ablation and recovery training helps the network to “lose weights”, providing a regularization effect and, thus, increasing the attack performance. For instance, when looking at the model from [46], it is rather large compared with the state-of-the-art and performs less stable when training multiple times with random weight initialization. By reducing the model’s size carefully, the model can indeed achieve more reliable performance [7].

B. Results for the ASCAD with Random Keys Dataset

Figure 3 presents the layer’s sensitivity and resilience. Again, the used MLP and CNN break the target with a given number of attack traces. Regarding layer sensitivity (blue bars), shallower MLP layers are more sensitive to ablation than the deeper layers. Indeed, even 90% of the ablation could result in limited performance degradation in the last layers (Figure 3c), confirming the extra capacity in these layers. In contrast, for CNN, deeper layers (L5/L6/L7) are more sensitive to ablation, as they introduce more GE variation before recovery training. (Figure 3f). This observation indicates that the deeper layers are more critical in the classification process for ASCAD_R. This observation is well-aligned with established CNN designs such as VGG16 [48]: the number of convolution filters increases when adding more convolutional layers. At the same time, the dense layer also has many neurons.

The layer resilience, represented by the GE difference after the recovery training, can validate the above conclusions. Although the model can adapt to the ablation effect in most cases, its recovery capability varies when ablating different layers. For MLP, ablating the shallower layers with a greater ablation rate, as shown in Figure 3c, results in the performance degradation. Still, when controlling the ablation rate in the reasonable range, the attack performance can be improved (Figure 3a), indicating a remarkable resilience of these layers. For CNN, the ablation effect can be minimal for almost all layers. The evaluator/designer can simplify the network without harming the attack performance.

VI. Experimental Results on Multiple Device Model from Single Device

The previous results show that our layer assessment method can accurately reflect the layer’s sensitivity and resistance. In this section, we use the knowledge obtained from this method to deal with the portability problem.

Aligned with Algorithm 1, we test three different ablation rates (10%, 50%, and 99%) for the Portability_2020 dataset. 99% ablation gave the best result, about 6× better than other ablation rates (see Figure 4). By 99% ablation, we consider ablating the whole layer except for a single neuron (to maintain the connectivity between layers), which is equivalent to creating a bottleneck layer. We hypothesize that portability can easily cause overfitting, affecting the whole layer. Thus, ablating the full layer (99%) could resolve the issues. Consequently, we use this configuration in the following experiments.

We use an MLP architecture with four hidden layers where each layer has 500 neurons, the $ReLU$ activation function, the batch size is 256, the number of epochs is 50, the loss function is categorical cross-entropy, and the optimizer is $RMSprop$ with a learning rate of 0.001, as proposed in [9]. This architecture is selected as the best-performing one since it has sufficient capacity to model the data and yet does not

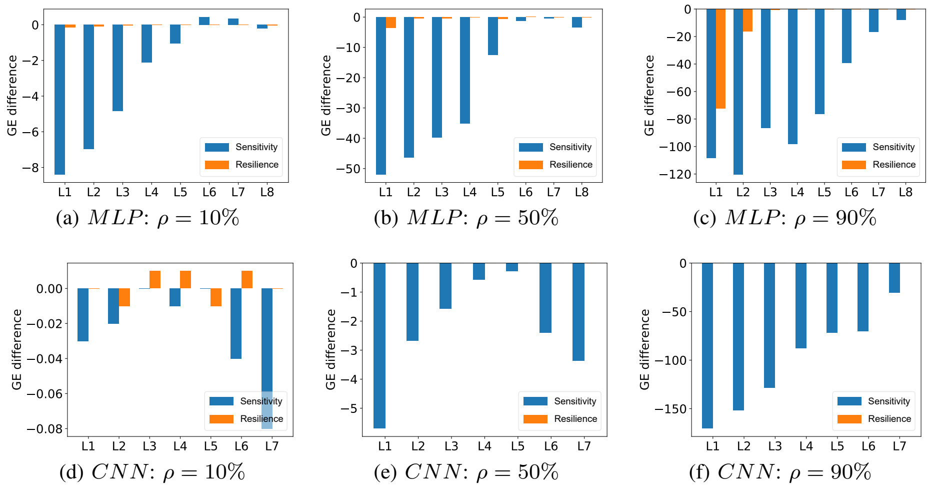


Fig. 2: Layer assessment for the ASCAD_F dataset.

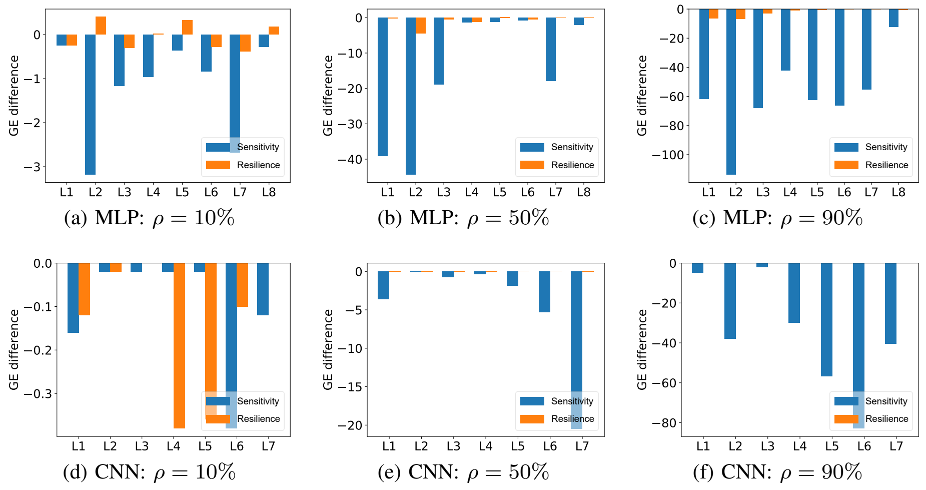


Fig. 3: Layer assessment for the ASCAD_R dataset.

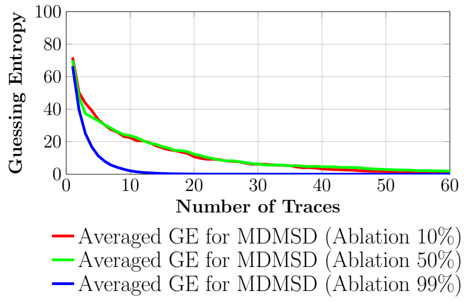


Fig. 4: Results of averaged GE for (B1_K1)—(B2_K2) and (B1_K1)—(B4_K3).

overfit as easily as the previously investigated CNNs. For the test settings, to simulate noise behavior for portability issues, we generate $20\times\alpha$ for the β value ($\alpha = 5{\cdot}10^{-4}$). This is based on the assumption that the additional noise due to portability will be larger than the measurement noise. We use 50 epoch for training (and recovery training) as in [9].

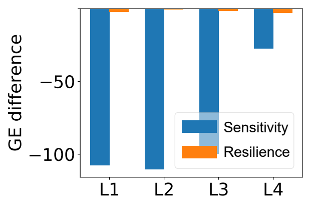
A. Results for the Portability_2020 Dataset

We train MLP for the dataset (Line 2 of Algorithm 2), with the (train) - (test) datasets as follows: (B1_K1) - (B1_K1), (B2_K2) - (B2_K2), (B3_K1) - (B3_K1), (B4_K3) - (B4_K3).

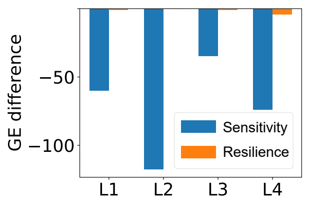
The layer performance for each dataset is shown in Figure 5. The second layer (L2) is the best candidate for ablation due to its high sensitivity and strong resilience (it achieves the best performance as GE_r is less than $1.05 \times GE_o$ for all experiments). Therefore, we utilize the recovery-trained architecture (ML^{ρ}_{r}) by ablating the layer L2. We benchmark the original results [9] with our method, shown in Figure 6a and Figure 6b. MDMSD outperforms the original work in almost all cases except for (B4_K3)—(B2_K2). Moreover, it mostly only requires 10-20 traces to recover the correct key.

B. Results for the CHESCTF_2018 Dataset

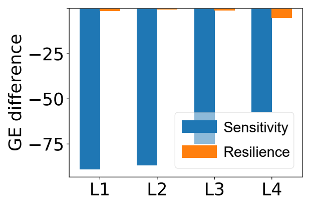
For this dataset, we focus on the KeySchedule leakage rather than S-box operation as reported in [49]. Specifically, we aim to recover the first byte of the round key in the KeySchedule



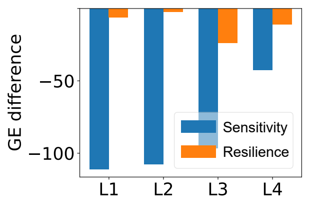
(a) (B1_K1)—(B1_K1)



(b) (B2_K2)—(B2_K2)

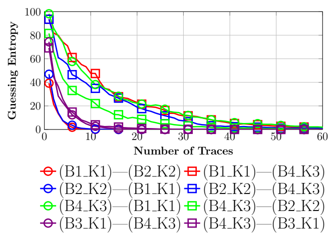


(c) (B3_K1)—(B3_K1)

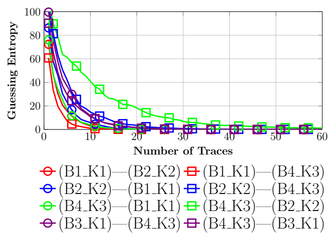


(d) (B4_K3)—(B4_K3)

Fig. 5: Layer assessment for the Portability_2020 dataset.



(a) Single device result



(b) MDMSD result

Fig. 6: Results for the Portability_2020 dataset.

operation. As the leakage happens in the HW leakage model, the range for GE is between 0 and 8.

In Figure 7, since L4 is the best candidate for ablation as it is both sensitive and resilient (it satisfies $GE_r \leq 1.05 \times GE_o$), the corresponding recovery-trained model is used to perform attacks. We first perform the cross-device attack on the CHESCTF_2018 dataset. Notice that we cannot recover the key when the model is trained with the B_RN dataset (Figure 8a) as the guessing entropy never converges. In contrast, our method recovers all secret information using less than 50 traces (see Figure 8b). More precisely, except for (B_RN)—(D_K6), only ten traces are needed to recover the round key (about 5 on average, considering all experiments).

We also compute the averaged GE for all cases for both datasets to represent the results more clearly. The averaged results are shown in Figure 9. For the Portability_2020 dataset, MDMSD requires half the traces (about 30) compared to the original results (about 60 traces) to break the target. For the CHESCTF_2018 dataset, MDMSD breaks the target easily while its counterpart hardly converges. Following these results, we confirm the effectiveness of our method in dealing with the portability problem. One may argue that pruning can be an alternative to ablation. However, it is impossible to know which neuron/convolution filter contributes to the device overfitting by only seeing the weights. Compared with the conventional approach that always uses the same model, ablation prevents the model from overfitting on a specific device, thus allowing more accessible adaptation to other devices. Since most weight info is kept after ablation, a reduced effort is required to rebuild the link between the leakages and labels.

VII. Conclusions and Future Work

Portability represents a real challenge in profiling SCA. While there is one approach (MDM) devised to resolve the problem, it can be difficult to use it in practice due to the requirement to have multiple open copies of the device to be attacked. This paper introduces a layer assessment methodology for deep learning-based SCA. Our methodology helps researchers understand which layers are sensitive to ablation and which are resilient after recovery training. With this knowledge, we can bridge the gap between the single-device model (the commonly used one) and MDM when multiple devices are unavailable, significantly improving the model's ability to generalize to different devices. By applying our layer assessment methodology, we achieve better results than the current state-of-the-art without overfitting to a single device.

In future work, we plan to optimize neural network models or design more resilient countermeasures with the insight of the model from our method. Besides, we only considered ablation performed in a layer-wise manner. While we are confident that such an approach gives the most explainable results, future works could examine ablating multiple layers simultaneously. This is especially interesting for CNNs, where we can ablate convolutional and fully connected layers. As the current deep learning-based SCA trend uses relatively small neural networks, we consider our work perfectly aligned with the state-of-the-art. Still, it would be interesting to investigate ablation on larger neural network architectures, as such architectures will become increasingly important with the improvements in the countermeasures and larger corresponding datasets (more features and more profiling traces, which could require larger neural network models).

References

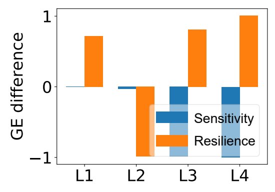
[1] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE symposium on security and privacy (sp)*. IEEE, 2019, pp. 1362–1380.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 388–397. [Online]. Available: https://doi.org/10.1007/3-540-48405-1_25

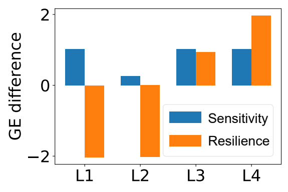
[3] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, B. S. K. Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523. Springer, 2002, pp. 13–28. [Online]. Available: https://doi.org/10.1007/3-540-36400-5_3

[4] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

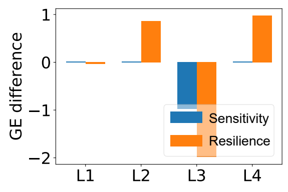
[5] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient cnn architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 1–36, Nov. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8391



(a) (A_RN)—(C_K4)

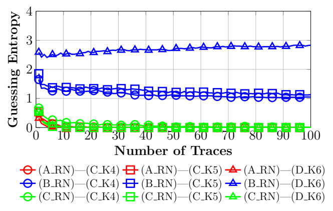


(b) B_RN—(C_K4)

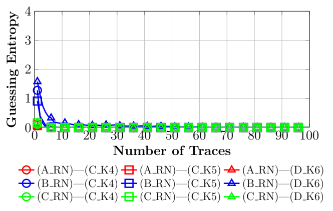


(c) (C_RN)—(C_K4)

Fig. 7: Layer assessment for the CHESCTF_2018 dataset

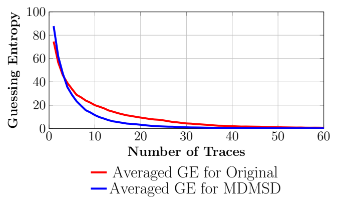


(a) Single device result

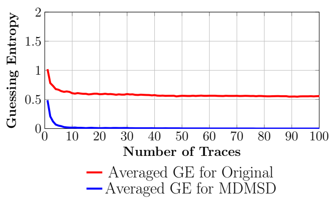


(b) MDMSD result

Fig. 8: Results for the CHESCTF_2018 dataset.



(a) Averaged GE result for the Portability_2020 dataset



(b) Averaged GE result for the CHESCTF_2018 dataset

Fig. 9: Results for the two datasets considering portability.

[6] L. Wu and S. Picek, "Remove some noise: On pre-processing of side-channel measurements with autoencoders," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 389–415, Aug. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8688

[7] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient cnn architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 147–168, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8586

[8] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, pp. 677–707, Jul. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8989

[9] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa, "Mind the portability: A warriors guide through realistic profiled side-channel analysis," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/mind-the-portability-a-warriors-guide-through-realistic-profiled-side-channel-analysis/

[10] V. Lomne, "Common criteria certification of a smartcard: a technical overview," *CHES 2016*, 2016.

[11] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Advances in Cryptology - EUROCRYPT 2009*, A. Joux, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 443–461.

[12] 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 Transactions on Cryptographic Hardware and Embedded Systems*, pp. 148–179, 2019.

[13] 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 *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 20–33.

[14] L. Lerman, S. F. Medeiros, G. Bontempi, and O. Markowitch, "A Machine Learning Approach Against a Masked AES," in *CARDIS*, ser. Lecture Notes in Computer Science. Springer, November 2013, berlin, Germany.

[15] G. Hospodar, B. Gierlichs, E. D. Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *J. Cryptogr. Eng.*, vol. 1, no. 4, pp. 293–302, 2011. [Online]. Available: https://doi.org/10.1007/s13389-011-0023-x

[16] A. Heuser and M. Zohner, "Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines," in *COSADE*, ser. LNCS, W. Schindler and S. A. Huss, Eds., vol. 7275. Springer, 2012, pp. 249–264.

[17] S. Picek, A. Heuser, and S. Guilley, "Template attack versus bayes classifier," *J. Cryptogr. Eng.*, vol. 7, no. 4, pp. 343–351, 2017. [Online]. Available: https://doi.org/10.1007/s13389-017-0172-7

[18] R. Gilmore, N. Hanley, and M. O'Neill, "Neural network based attack on a masked implementation of AES," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2015, pp. 106–111.

[19] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 45–68.

[20] D. Kwon, H. Kim, and S. Hong, "Non-profiled deep learning-based side-channel preprocessing with autoencoders," *IEEE Access*, vol. 9, pp. 57 692–57 703, 2021.

[21] S. Picek, I. P. Samiotis, J. Kim, A. Heuser, S. Bhasin, and A. Legay, "On the performance of convolutional neural networks for side-channel analysis," in *Security, Privacy, and Applied Cryptography Engineering*, A. Chattopadhyay, C. Rebeiro, and Y. Yarom, Eds. Cham: Springer International Publishing, 2018, pp. 157–176.

[22] G. Perin, L. Chmielewski, and S. Picek, "Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 337–364, Aug. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8686

[23] 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 Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, p. 235–274, Jul. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8974

[24] G. Perin, L. Wu, and S. Picek, "Exploring feature selection scenarios for deep learning-based side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 4, p. 828–861, Aug. 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9842

[25] B. Hettwer, S. Gehrer, and T. Güneysu, "Deep neural network attribution methods for leakage analysis and symmetric key recovery," in *Selected Areas in Cryptography – SAC 2019*, K. G. Paterson and D. Stebila, Eds. Cham: Springer International Publishing, 2020, pp. 645–666.

[26] L. Masure, C. Dumas, and E. Prouff, "Gradient visualization for general characterization in profiling attacks," in *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*,

ser. Lecture Notes in Computer Science, I. Polian and M. Stöttinger, Eds., vol. 11421. Springer, 2019, pp. 145–167. [Online]. Available: https://doi.org/10.1007/978-3-030-16350-1_9

[27] D. van der Valk, S. Picek, and S. Bhasin, "Kilroy was here: The first step towards explainability of neural networks in profiled side-channel analysis," in *Constructive Side-Channel Analysis and Secure Design*. Springer International Publishing, 2021, pp. 175–199. [Online]. Available: https://doi.org/10.1007/978-3-030-68773-1_9

[28] T. Yap, A. Benamira, S. Bhasin, and T. Peyrin, "Peek into the black-box: Interpretable neural network using sat equations in side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 2, p. 24–53, Mar. 2023. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/10276

[29] G. Zaid, L. Bossuet, M. Carbone, A. Habrard, and A. Venelli, "Conditional variational autoencoder based on stochastic attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 2, p. 310–357, Mar. 2023. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/10286

[30] G. Perin, L. Wu, and S. Picek, "I know what your layers did: Layer-wise explainability of deep learning side-channel analysis," Cryptology ePrint Archive, Paper 2022/1087, 2022, https://eprint.iacr.org/2022/1087. [Online]. Available: https://eprint.iacr.org/2022/1087

[31] O. Choudary and M. G. Kuhn, "Template attacks on different devices," in *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, ser. Lecture Notes in Computer Science, E. Prouff, Ed., vol. 8622. Springer, 2014, pp. 179–198. [Online]. Available: https://doi.org/10.1007/978-3-319-10175-0_13

[32] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-deepsca: Cross-device deep learning side channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019, DAC 2019, Las Vegas, NV, USA, June 02-06, 2019*. ACM, 2019, p. 134. [Online]. Available: https://doi.org/10.1145/3316781.3317934

[33] A. Gohr, S. Jacob, and W. Schindler, "CHES 2018 side channel contest CTF - solution of the AES challenges," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 94, 2019. [Online]. Available: https://eprint.iacr.org/2019/094

[34] U. Rioja, L. Batina, and I. Armendariz, "When similarities among devices are taken for granted: Another look at portability," in *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, ser. Lecture Notes in Computer Science, A. Nitaj and A. M. Youssef, Eds., vol. 12174. Springer, 2020, pp. 337–357. [Online]. Available: https://doi.org/10.1007/978-3-030-51938-4_17

[35] F. Zhang, B. Shao, G. Xu, B. Yang, Z. Yang, Z. Qin, and K. Ren, "From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices," in *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*. IEEE, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/DAC18072.2020.9218693

[36] H. Yu, H. Shan, M. Panoff, and Y. Jin, "Cross-Device Profiled Side-Channel Attacks using Meta-Transfer Learning," http://jin.ece.ufl.edu/papers/DAC2021_AI4SCA.pdf, 2021.

[37] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, "Sok: Deep learning-based physical side-channel analysis," *ACM Comput. Surv.*, vol. 55, no. 11, feb 2023. [Online]. Available: https://doi.org/10.1145/3569577

[38] P. H. Schiller, "The effect of superior colliculus ablation on saccades elicited by cortical stimulation." *Brain research*, 1977.

[39] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, "Ablation studies in artificial neural networks," *CoRR*, vol. abs/1901.08644, 2019. [Online]. Available: http://arxiv.org/abs/1901.08644

[40] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1135–1143.

[41] G. Vilone and L. Longo, "Explainable artificial intelligence: a systematic review," *CoRR*, vol. abs/2006.00093, 2020. [Online]. Available: https://arxiv.org/abs/2006.00093

[42] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 163–177.

[43] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1095–1108.

[44] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

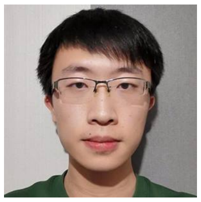
[45] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[46] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptographic Engineering*, vol. 10, no. 2, pp. 163–188, 2020. [Online]. Available: https://doi.org/10.1007/s13389-019-00220-8

[47] C. on Cryptographic Hardware and E. Systems, "Ches 2018 ctf," 2018, https://chesctf.riscure.com/2018/news. [Online]. Available: https://chesctf.riscure.com/2018/news

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[49] T. Damm, S. Freud, and D. Klein, "Dissecting the ches 2018 aes challenge." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 783, 2019.



Lichao Wu is a PhD student in the cybersecurity research group at the Delft University of Technology. After obtaining a bachelor's degree at Northwestern Polytechnical University (2015), Wu received his Master's degree in Microelectronic at the Delft University of Technology in 2017. His main research interests are at the intersection of implementation attacks, cryptography, and machine learning.

Yoo-Seung Won is a Research Scientist at PACE Lab, Temasek Laboratories, Nanyang Technological University (NTU), Singapore. He received his PhD from Kookmin University in 2018. Before NTU, Yoo-Seung held the position of Staff Engineer in the foundry business of Samsung Electronics, South Korea from 2018. His research interests include embedded security, secure boot solution, cold boot attack, side-channel analysis and fault attack schemes and countermeasures, practical laser/EM fault injection, and deep learning security.

Dirmanto Jap is a Research Scientist at PACE Lab, Temasek Laboratories, Nanyang Technological University (NTU), Singapore. He previously received his PhD in Mathematics from NTU in 2016. His main research topics include physical attacks and countermeasures, practical fault injection, application of machine learning for security.

Guilherme Perin is a postdoctoral researcher at Radboud University. He graduated in Electrical Engineering (2008) and has Master in Informatics (2011) from the Federal University of Santa Maria. In 2014, he received his PhD in Microelectronics and Automated Systems at the University of Montpellier. His research areas include hardware security, cryptography, optimization algorithms, and machine learning.

Shivam Bhasin is a Senior Research Scientist and Programme Manager (Cryptographic Engineering) at NTU Singapore. He received his PhD from Telecom Paristech (2011) and Master's from Mines Saint-Etienne (2008). Before NTU, Shivam held the position of Research Engineer in Institut Mines-Telecom, France. His research interests include embedded security and trusted computing.

Stjepan Picek is an associate professor at Radboud University, The Netherlands. He received his PhD in 2015. From 2017 to 2021, he was an assistant professor at the Delft University of Technology, The Netherlands; from 2015 to 2017, he was a postdoctoral researcher at KU Leuven, Belgium and MIT, USA. His research interests include security, machine learning, and evolutionary algorithms.