SoK: Deep Learning-based Physical Side-channel Analysis

Stjepan Picek
Delft University of Technology,
The Netherlands
Email: S.Picek@tudelft.nl

Guilherme Perin
Delft University of Technology,
The Netherlands
Email: G.Perin@tudelft.nl

Luca Mariot
Delft University of Technology,
The Netherlands
Email: L.Mariot@tudelft.nl

Lichao Wu
Delft University of Technology,
The Netherlands
Email: L.Wu-4@tudelft.nl

Lejla Batina
Radboud University, Nijmegen,
The Netherlands
Email: lejla@cs.ru.nl

***Abstract*—Side-channel attacks represent a realistic and serious threat to the security of embedded devices for almost three decades. The variety of attacks and targets they can be applied to have been introduced, and while the area of side-channel attacks and mitigations is very well-researched, it is yet to be consolidated.**

Deep learning-based side-channel attacks entered the field in recent years with the promise of more competitive performance and enlarged attackers' capabilities compared to other techniques. At the same time, the new attacks bring new challenges and complexities to the domain, making a systematization of the existing knowledge ever more necessary.

In this SoK, we do exactly that, and by bringing new insights, we systematically structure the current knowledge of deep learning in side-channel analysis. We first dissect deep learning-assisted attacks into different phases and map those phases to the efforts conducted so far in the domain. For each of the phases, we identify the weaknesses and challenges that triggered the known open problems. We connect the attacks to the existing threat models and evaluate their advantages and drawbacks. We finish by discussing other threat models that should be investigated and propose directions for future works.

***Index Terms*—Side-channel attacks, Deep learning, Profiling attacks, Supervised learning**

I. Introduction

The market of embedded devices constantly grows. Already in the 2020 world of connected devices, there were more IoT connections than non-IoT connections. The current number of active connected devices is estimated to be more than 20 billion and is expected to reach more than 30 billion by 2025 [1]. This escalation in the number of devices is followed by increased security concerns and vulnerabilities, resulting in the ever-growing demand for certified products. As a consequence, millions of products are undergoing strict security assessments in evaluation labs around the world on a daily basis [2]. As a part of those evaluations, side-channel analysis (SCA) represents a well-known threat since the 90s [3] and is widely studied by researchers in the community of information security and cryptography from both industry and academia [4], [5], [6].

Various physical leakages such as timing delay [7], power consumption [8], electromagnetic emanation (EM) [9] become available during the device's computation with the (secret) data. Those phenomena have led to a whole new research area. There, by combining the physical observation of a specific internal state within computation and a hypothesis on the data being manipulated, it is possible to recover the intermediate state processed by the device. Thus, it is possible to "break" the device, i.e., learn its secrets.

SCAs and corresponding mitigation evolved in the past three decades, and more recently, deep learning-assisted side-channel analysis became widely used for this kind of research. Naturally, the security industry has started using such techniques as standard ones in the design and certification process. A recent example is the machine learning approach (unsupervised clustering) to break Google Titan Security Key [10]. The main goal is to consider the worst-case adversaries and make such techniques well understood and used as efficiently as possible. For example, Common Criteria security evaluation of a device evaluates the time required to perform a successful attack and the attack effort, i.e., the difficulty of it, which both have an impact on the chip's final security rating [11]. *In short, confidence in security evaluation implies considering worst-case adversaries, and then one needs to go for the best techniques known, including deep learning-based SCA [12].*

The appeal of using deep learning in side-channel analysis in the last few years is evident, as demonstrated in Figure 1. Clearly, from 2016 when the first paper appeared that uses deep learning to conduct side-channel analysis [13], the domain became very active[1]. By analyzing those works, we can notice two main advantages being commonly brought up: 1) deep learning-based SCA is very powerful and can break targets protected with countermeasures, and 2) deep learning-based SCA requires less (or no) effort to pre-process the side-

[1]To be more precise, this is the first application of convolutional neural networks. There are earlier papers that use multilayer perceptrons, but since they do not report the number of hidden layers or this number is set to one, we do not consider those works as deep learning-based SCA.

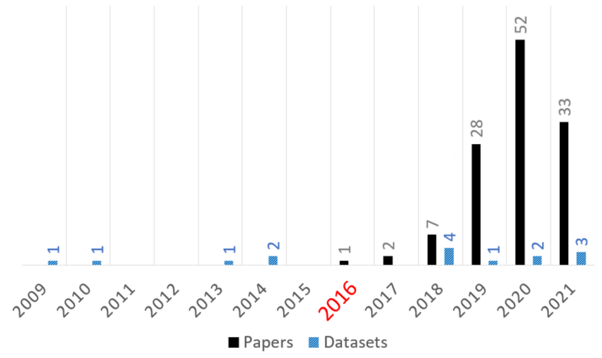

Fig. 1: Distribution of papers and datasets per year that use deep learning in side-channel analysis. Note that the information for 2021 only contains papers and datasets released in the first seven months (until August 1). For multiple versions of the same papers, we consider the peer-reviewed version. We consider papers published in English only. Observe how datasets appear more consistently (and in larger numbers) in the last few years, which we connect with a considerable interest in the SCA community for deep learning-based SCA.

channel measurements and prepare the measurements for the attack.

At the same time, the main disadvantage (and an inspiration for many research works) is the need to conduct hyperparameter tuning, which is considered an important and challenging task. With all the diverse strategies and techniques in deep learning-based side-channel analysis, it is not obvious how effective and efficient are the different approaches and how they compare to each other. In addition, it is hard to identify the primary challenges as they are typically device- or threat model-specific. Our motivation for this work is to systematize and critically evaluate previously proposed approaches. We also aim to identify the main challenges and offer actionable steps to solve those challenges. As such, we consider our work a crucial stepping stone in understanding state-of-the-art deep learning attacks in SCA.

Some related works already itemized various machine learning-based side-channel attacks [14] and research challenges [15]. This systematization of knowledge paper goes far beyond those works by providing a detailed list of up-to-date challenges with the latest development in the domain and recommendations on how to address them. This paper does not aim to cover every proposed approach but systematically identify and analyze the main approaches in getting deep learning to work for side-channel adversaries. Our focus is on attacks exploiting the power and EM radiation

With this systematization of knowledge paper, we make the following contributions:

1) **D**iscuss existing threat models and identify the differences among them and the impact of those to practice.
2) **E**nlist relevant stages in deep learning-based SCA and the impact of neglecting those stages.
3) **E**valuate and compare approaches proposed in terms of performance.
4) **P**inpoint why many proposed solutions are not adopted in practice and what the necessary criteria for new solutions are.

Our analysis recognized three threat models and the need to add one more threat model. Next, we divided the deep learning process into six phases and provided 18 challenges. We discussed 13 recommendations to address the challenges and improve state-of-the-art. Finally, we recognized seven works that are novel and have a significant impact on the deep learning-based SCA.

The rest of this paper is organized as follows. In Section II, we provide necessary background information about side-channel attacks, where we emphasize deep learning-based attacks. Next, in Section III, we provide an overview of different threat models commonly used. Section IV discusses various phases of supervised learning and connects them with important results in the SCA research. Section V provides an overview of previous work in AI explainability for SCA, and in Section VII, we conclude the paper. Finally, in the Appendix, we provide additional information about the side-channel analysis, deep learning, and additional applications of deep learning in SCA.

II. Background

We use calligraphic letters like $\mathcal{X}$ to denote sets. The corresponding upper-case letters denote random variables (X) and random vectors ($\mathbf{X}$) over $\mathcal{X}$. The corresponding lower-case letters (x, $\mathbf{x}$) represent realizations of X and $\mathbf{X}$, respectively.

A dataset $\mathcal{D}$ represents a collection of side-channel measurements (traces). Each trace $\mathbf{d}_i$ from $\mathcal{D}$ is a time sequence and is associated with an input value (plaintext or ciphertext) [2] $\mathbf{a}_i$ and a key $\mathbf{k}_i$, where $\mathbf{k}_i$ denotes a generic key candidate and the correct key is k^*. The keys take their values from the keyspace $\mathcal{K}$. Each trace $\mathbf{d}_i$ consists of a total of F features (samples, points of interest).

A. Side-channel Analysis

Side-channel analysis (SCA) considers attacks that do not aim at the weaknesses of the algorithms but their implementations [16]. The central idea of side-channel analysis is to compare some secret data-dependent predictions of the physical leakages and the actual (measured) leakage to identify what data is most likely to have been processed. In practice, this requires the ability to model the leakages (to come up with the predictions for data) and to have a good comparison tool (distinguisher), to extract the secret information efficiently. A detailed description of this process is given in Appendix A.

A leakage model represents a function mapping the hypothetical data value toward the (approximation of) physical leakage of the device. Common (well-studied and confirmed)

[2] As commonly done, we assume known plaintext and attack from the encryption side or known ciphertext and attack from the decryption side. The attack principle remains the same.

leakage models take either the Hamming weight of the hypothetical data (assuming that the physical leakage is proportional to the number of ones in the intermediate value) or the Hamming distance between the registers that store the data values at two different moments.

Typical for side-channel analysis is the divide-and-conquer approach, which aims at recovering the sensitive variables in parts (e.g., sub-key bytes in the case of AES), making the approach computationally feasible. This means that the approach described above is repeated for each sub-key until the (full) key k^* is recovered. In general, if one sub-key can be recovered, it is enough to argue on the implementation's weakness.

a) SCA Taxonomy: Attacks performed in the SCA field can be divided into direct attacks and two-stage attacks.
In direct (also called non-profiling) attacks, the adversary obtains some (large) number of measurements from the device under attack and uses statistical techniques to infer secret information. Common examples of such attacks are simple power analysis (SPA) and differential power analysis (DPA) [8] [3] While direct attacks assume less powerful attackers, they might require millions of measurements to obtain secret information.

In the two-stage (also called profiling) attacks, a powerful attacker has a clone device identical (or at least similar) to the device to be attacked. The attacker uses the clone device to build the model of a device and ultimately attack the target device. This attack happens in two phases, commonly denoted as the profiling and the attack phases.

b) Profiling Attacks: The foundation behind profiling techniques is that side-channel measurements follow an unknown distribution that can only be approximated by an assumed statistical distribution for the leakage. The first and best-known method for profiling attacks is the *template attack* where an adversary assumes that the leakage follows a multi-variate Gaussian distribution [17], [18]. The profiling phase consists then of computing statistical parameters for a Gaussian mixture model. Thus, the model is built for each possible hypothetical leakage class (e.g., all possible Hamming weight values of a byte). In the attack phase, the adversary computes the probability that a new side-channel measurement (under attack) belongs to a certain class by using the computed probability density function from the approximate statistics. While the profiling attack assumes a more powerful attacker than a non-profiling one, it requires significantly fewer traces than direct attacks to break the target: sometimes, only one trace is sufficient in a profiling attack.

Machine learning techniques were later adopted for profiling attacks, and here the statistics of the unknown leakage distribution are automatically learned from the profiling set. Thus, one advantage of machine learning over template attacks is that the profiling model is learned without any assumption about the statistical distribution of the leakage.

Machine learning models can be further divided into classical machine learning, where a common step before conducting the attack is to run feature engineering, and *deep learning*, where one uses raw features or an optimized trace time interval thereof. Although the first category of machine learning models is out of scope in this work, we list several references for interested readers such as [19], [20], [21]. Additionally, we point to a survey of machine learning-based side-channel analysis [14]. The main differences between our work and [14] are: 1) we investigate only deep learning-based SCA while [14] also considered other machine learning techniques, as well as different-than-SCA attacks, 2) [14] is written in the early days of deep learning-based SCA (early 2018), so it is covering only a few deep learning works, and 3) [14] is a survey work that gives detailed information about related works but not a systematization of knowledge.

c) Countermeasures: It is common to protect the implementations with countermeasures against SCA. Countermeasures aim to break the statistical link between intermediate values and traces (e.g., power consumption or EM emanation). There are two main categories of countermeasures for SCA: masking and hiding [16].
In masking, a random value (mask) is generated to conceal every intermediate value. More precisely, random masks are used to remove the correlation between the measurements and the secret data. Two commonly used types of masking are Boolean masking and arithmetic masking [16].

On the other hand, the goal of hiding is to make measurements appearing random or constant. Hiding decreases the signal-to-noise ratio (SNR) only. Hiding can happen in the amplitude (e.g., adding noise) and time (e.g., desynchronization, random delay interrupts, jitter) domains.

Countermeasures are selected based on the adversary's capability. Hiding countermeasures could be theoretically defeated by an adversary unbounded in terms of side-channel measurements [4]. Similarly, masking countermeasure considering a first-order masking scheme (first-order masking means that each sensitive variable is combined with a single mask) can be (in the worst-case) defeated by an adversary that can access mask shares during a profiling phase. Consequently, highly protected targets consider using high-order masking schemes or a combination of masking and hiding countermeasures.

d) SCA Adversary: There are two types of adversaries in SCA, namely the security evaluator and the attacker. The main difference is that the security evaluator knows some secret information about the device under attack beforehand, as they work closely with manufacturers. Also, when considering the security evaluator perspective, it is more common to speak about side-channel analysis, while for attackers, we usually refer to side-channel attacks.

[3] Besides power, it is common to use other side channels like EM, which would, strictly speaking, change the technique names to SEMA and DEMA. Still, as the differentiation among those side channels is not important in our discussion, we use SPA/DPA for all such attacks, regardless of the side channel.

[4] In reality, there are no attackers unbounded in terms of power, but it is common to assume the attacker to be arbitrarily powerful.

B. Deep Neural Networks

A neural network is a mapping $f_{\boldsymbol{\theta}}$ such that $\mathcal{X} \rightarrow \mathcal{Y}$. It operates as a sequence of layers transforming an input $\mathbf{x}$ to an output y, i.e., $f = f_1 \circ f_2 \circ \ldots \circ f_l$, where $f_1 = \mathbf{x}$, $f_{l+1} = y$, and l is the number of layers.

The raw output of $f(\mathbf{x})$ is referred as logits, and the softmax $\sigma(\mathbf{x})$ is applied to convert logits into a probability distribution. The inferred label of an input $\mathbf{x}$ by a network is then defined as $\arg\max_i \sigma(f(\mathbf{x}))$ where $i \in \{1, 2, \cdots, |c|\}$, and $|c|$ is the number of classes (labels). When converting a label c into distribution, it is common to use one-hot encoding $\mathbf{y}_c = \mathbb{1}_i(c)$, i.e., a vector with one at position c all zero at the others.

In a neural network, each layer f_i contains neurons that apply a linear and nonlinear transformation to their inputs, which are the outputs of the previous layers neurons. The number of neurons in a layer is called its width, and the total number of layers is its depth. The i-th layer computes $f_i(\mathbf{x}) = \phi_i(\mathbf{w}_i \cdot \mathbf{x} + \mathbf{b}_i)$. Here, ϕ_i represents a nonlinear activation function, and $\mathbf{w}_i$ and $\mathbf{b}_i$ are the vectors of weights and biases (the trainable parameters).

C. Deep Learning-based Side-channel Analysis

This SoK considers deep learning-based side-channel analysis that follows the supervised learning paradigm and classification task where the output $\mathbf{y}$ is a probability distribution over $|c|$ labels. *In SCA, the label c is derived from the key k^* and input $\mathbf{a}_i$ through a cryptographic function CF and a leakage model LM.*

Let us assume a data distribution $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ (with $\mathcal{X} \subseteq \mathbb{R}^F$ and $F \in \mathbb{N}$) is the domain of input and $\mathcal{Y}$ (with $\mathcal{Y} \subseteq \mathbb{R}^{|c|}$) is the range of outputs. The objective of a learning algorithm Alg is to learn a parameterized $f_{\boldsymbol{\theta}} \in \mathcal{H}$, where $\mathcal{H}$ is the space of hypotheses. More precisely, $\mathcal{H}$ is a functional space where each element $f \in \mathcal{H}$ is a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ between the input and the labels. Finally, $f_{\boldsymbol{\theta}} \in \mathcal{H}$ is a specific function with parameters $\boldsymbol{\theta} \in \mathbb{R}^n$, where n denotes the number of trainable parameters. In what follows, we will restrict the hypothesis space $\mathcal{H}$ to neural network models.

To train the neural network representing $f_{\boldsymbol{\theta}}$, the learning algorithm Alg has access to the dataset $\mathcal{D}$ drawn from the underlying distribution $\mathcal{X} \times \mathcal{Y}$. As we assume the supervised learning setting, $\mathcal{D}$ is partitioned into disjoint subsets commonly denoted as the training set $\mathcal{D}_{tr}$, validation set $\mathcal{D}_{vl}$, and test set $\mathcal{D}_{ts}$. The size of the training dataset equals N, the size of the validation set equals V, and the size of the test set equals Q. *In the context of SCA, the different datasets represent data acquisitions from different targets or phases of the profiling attack.* Then, the classification process can be broken down into the following steps:

1) Training: Let us assume a non-negative, real-valued loss function $L(f_{\boldsymbol{\theta}}(\mathbf{x}), y)$ quantifying how correct the prediction of a neural network is for an input $\mathbf{x} \in \mathcal{X}$ and label $y \in \mathcal{Y}$. For a loss function L, the output of the supervised learning algorithm is a neural network $f_{\boldsymbol{\theta}}$ minimizing the risk $\mathcal{E}_{(\mathbf{x},y)\sim\mathcal{X}\times\mathcal{Y}}\left[L(f(\mathbf{x}), y)\right]$, with $f(\mathbf{x})$ being the predicted label and y the true label.

Since the underlying data distribution $\mathcal{X} \times \mathcal{Y}$ is not known, supervised learning algorithm uses the training set $\mathcal{D}_{tr}$ of size N to learn a hypothesis that minimizes the empirical risk[5] $\frac{1}{N}\sum_i^N L(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$, where $(\mathbf{x}_i, y_i) \sim \mathcal{D}_{tr}$.

To minimize the loss, it is common to use the backpropagation algorithm to update the parameters $\boldsymbol{\theta}$ with a multiplication of the derivative of the empirical risk concerning the model parameters $\boldsymbol{\theta}_t$ at each iteration t.

During the training process [6], a neural network is tested based on how well it generalizes on the unseen examples from the validation set $\mathcal{D}_{vl}$. Once it converges to an acceptable error rate, training stops, and the neural network, along with its parameters, is stored as $f_{\boldsymbol{\theta}}$ where $f_{\boldsymbol{\theta}}$ *represents the SCA profiling model.*

2) Testing: In this phase [7], the goal is to predict labels y based on previously unseen traces $\mathbf{x} \in \mathcal{D}_{ts}$, and the trained model $f_{\boldsymbol{\theta}}$. *The SCA metrics, such as guessing entropy and success rate, are used to assess the testing outcomes as discussed in the next section.*

D. Evaluation of Side-channel Analysis

The outcome of predicting with a model $f_{\boldsymbol{\theta}}$ on the test set $\mathcal{D}_{ts}$ is a two-dimensional matrix P with dimensions $Q \times |c|$. The probability $S(k)$ for any key byte candidate k is a valid side-channel distinguisher (a tool to differentiate between various key guesses), where it is common to use the maximum log-likelihood principle $S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,c})$. The value $\mathbf{p}_{i,c}$ denotes the probability that for a key k and input $\mathbf{a}_i$, we obtain the label c.

It is common to estimate the effort to obtain the secret key k^* from the predictions with metrics like key rank (KR), success rate (SR), and guessing entropy (GE). With Q traces in the attack phase, an attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability where g_1 denotes the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate.

The key rank represents the correct key k^* position in the key guessing vector $\mathbf{g}$.
The success rate of order o is the average empirical probability that the secret key k^* is located within the first o elements of the key guessing vector $\mathbf{g}$.
The guessing entropy is the average position of k^* in $\mathbf{g}$ [22] [8].
Note that since all those metrics assume the knowledge of the correct key k^*, they are appropriate for evaluation setting. On the other hand, the attacker should test the best guesses until reaching the one resulting in the correct plaintext.

III. Threat Models in Profiling SCA

Several threat models are typically used in the deep learning-based side-channel analysis. They all assume the

[5]In other words, the empirical risk is the expected value of the loss function over all realizations of the joint distribution $\mathcal{X} \times \mathcal{Y}$.

[6]We will use the terms training and profiling interchangeably.

[7]We will use the terms testing and attack interchangeably.

[8]J. Massey first defined guessing entropy as the expected number of guesses using an optimal strategy to correctly guess the value of a random variable [23].

attacker with unlimited power and the capability of obtaining an arbitrary number of profiling traces or evaluating any number of profiling models. Naturally, the attacker's power is always bounded, but it is unclear how to do it. One recent attempt tries to set up a framework where the attacker is required to find the smallest number of profiling traces or evaluate the smallest number of neural network architectures to break the target [24]. By doing so, the authors aimed to "force" efficient attacks. We depict different threat models in Figure 2.

A. Classical Threat Model

In this threat model, we assume that the attacker conducts profiling and attack on the same device (thus, this setup is also denoted as "single-device-model" [25]). While this setting is, of course, not realistic, we can notice it represents the most-explored setting in academia, see, e.g., [26], [27], [28]. There are two fundamental reasons for it as it 1) allows easier setup since it is not required to have two devices and conduct data acquisition phase on different devices, 2) assumes the best-case setting for the attacker where the measurements come from the same distribution. Indeed, this attack can be considered as an upper bound to estimate the attacker's power. Additionally, we can recognize two further types of this threat model: 1) the secret key is the same for the profiling and attack set, and 2) the secret key is different for profiling and attack set.

The main advantages of this threat model are that it is easier to run the setup (one device less), and it gives the best-case estimate for the attacker. On the other hand, the main disadvantage is that it is not realistic. This model can be considered a white-box threat model since the attacker knows (all) details about the device under attack. This model corresponds to the role of the attacker as a security evaluator.

B. Portability Threat Model

In this threat model, we assume there are (at least) two devices: one for profiling and the second one for the attack (thus, this attack is also known as the cross-device attack [29]). We refer to such a setting as portability, and it corresponds to all scenarios where an attacker has no access to measurements from the device under attack (to conduct a training) but only to measurements from a similar device, with uncontrolled variations, in process, measurement setup, or other stochastic factors [25].

This threat model is less used in academia but is getting more attention in the last two years [29], [30], [31], [32], [33]. The main difficulty for this threat model stems from the fact that the two devices (and corresponding datasets) do not necessarily follow the same distribution, resulting in an effect called over-specialization where a neural network (or a part of it) learns to generalize only for a specific dataset and cannot generalize for other datasets, as encountered in portability [34]. Bhasin et al. [25] recognize the validation phase as the main problem for the portability threat model since validation on the measurements from the profiling device indicates a too optimistic performance of the profiling model. Consequently, they recommended a multi-device model (MDM) with at least three devices: one for training, one for validation, and one for the attack.

Additionally, Zhang et al. proposed several categories of cross-device attacks [35]:

- Same devices: the traces used in profiling and attacking phases are collected from the same device. The only difference is the key variation. *Observe that this setting is the same as described in the previous section, indicating that there is also some disagreement on what is the portability (cross-device) setting.*
- Identical devices: the profiling and target device are two physical copies of the same chip model. All the designs and configurations are identical.
- Homogeneous devices: this case extends the dissimilarity of crossed devices at the chip level. Both devices have chips from the same manufacturer but with different models and structures.
- Heterogeneous devices: the core chips of the two devices to be compared come from different manufacturers, differing in all aspects, such as models, instruction set architectures, and power dissipation.

The main advantage of this threat model is that it is realistic, while the disadvantage is that it requires more effort to conduct data acquisition. This model can be considered from gray-box to black-box, depending on the differences between devices.

C. Non-profiling Supervised Threat Model

B. Timon proposed a threat model that combines the traits of non-profiling SCAs and supervised deep learning [36]. The author called the attack differential deep learning analysis (DDLA). This attack runs differential analysis (like in DPA) and then deep learning to assess how well the differential part worked. The attack requires that for each key hypothesis k, the attacker computes hypothetical values and partition those values based on the leakage model. Next, the attacker runs a deep learning model with traces and partitions. For the correct key k^*, the intermediate values will be correctly guessed, while other key guesses will not be consistent with the traces.

The main advantage of this threat model is that it does not require a copy of a device for training. Compared to other non-profiling attacks like DPA, DDLA can perform better, especially if the target is protected with countermeasures. On the other hand, the main disadvantages of this attack are that the performance of the profiling attack can be significantly better (considering the number of measurements required from the device under attack) and that we must train a neural network for each key guess, commonly resulting in 256 neural networks for the intermediate value leakage model (for the AES cipher when modeling the output of an S-box). As we do not require knowledge about the profiling device, this threat model can be considered a black-box.

***Achievement** 1:* The first application of the non-profiling deep learning threat model in the profiling SCA [36].

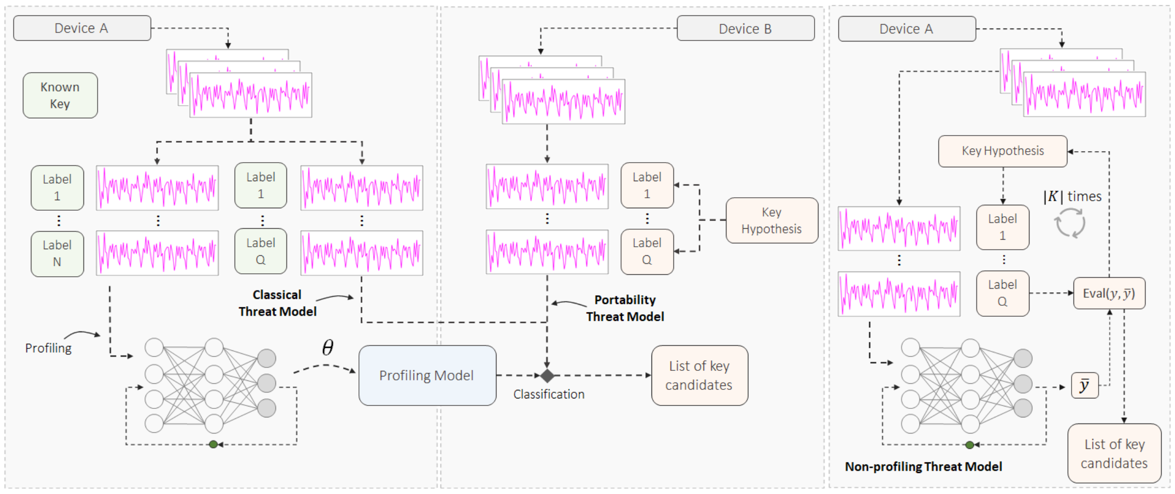

Fig. 2: Threat models in profiling SCA. The attacker uses N measurements from the profiling device to build a model. Next, in the attack phase, the adversary uses Q measurements from the device under attack to infer the secret information.

***Challenge** 1:* Design a functional approach for the unsupervised deep learning-based SCA.

***Recommendation** 1:* While developing an unsupervised deep learning threat model is straightforward, it is more difficult to make it functional. Indeed, there are no reported successful results from a fully unsupervised model to the best of our knowledge. At the same time, unsupervised (simple) machine learning approaches based on, e.g., clustering, give good results [37], [10]. Intuitively, this threat model requires only a single device (device under attack) to obtain a number of measurements with an unknown key. Afterward, it is required to use those measurements and infer the key. We recommend first exploring various semi-supervised deep learning approaches to understand the limitations in the training set size before moving toward the unsupervised approaches.

IV. Deep Learning Process and SCA

In Figure 3, we depict the common phases of the deep learning-based SCA. We note that not all the investigated works follow this flowchart. For each of the phases, we discuss relevant works and challenges to be addressed. We also provide recommendations on how to approach certain open questions.

Our analysis shows that deep learning-based SCAs use mostly multilayer perceptron (MLP) and convolutional neural networks (CNNs). Besides those methods, there are several applications of autoencoders (to denoise the traces, conduct dimensionality reduction, or classify). Finally, there are a few applications of recurrent neural networks (LSTMs), residual neural networks (RNNs), and generative adversarial networks (GANs).

A. Raw Data

The first task when running side-channel analysis is to conduct data acquisition and obtain the measurements. While this step is extremely important, it is commonly considered as "only" engineering and not discussed in related works. This task is also arduous, as it assumes (besides relevant knowledge) that the researchers also have 1) good-enough equipment to record high-quality traces and 2) state-of-the-art implementations. Finally, as the researchers should share and maintain the dataset with the community once those technical details are fulfilled, this phase includes continuous effort. Unfortunately, these conditions are seldom fulfilled, resulting in only a limited number of publicly available datasets.

We can divide datasets into those that consider symmetric-key implementations and public-key implementations. For symmetric-key implementations, a standard target is AES. For public-key implementations, standard targets are RSA and ECC. Let us immediately note a significant practical difference for SCA on symmetric-key or public-key implementations. The goal of SCA on symmetric-key implementations is to break the target with a single attack trace, but commonly, one requires (significantly) more traces for this task. On the other hand, breaking the target with a single trace is required for public-key cryptography as one commonly uses key randomization countermeasures and ephemeral keys. Consequently, there is only a single measurement with a specific key.

We present the most relevant information about publicly available datasets in Table I [9]. Additionally, we briefly discuss the advantages and drawbacks of those datasets in Appendix A. We also note that it is relatively common to simulate the effect of countermeasures by changing the datasets. Common options include the addition of Gaussian noise or simulating the effect of desynchronization, random delay interrupts, jitter, or S-box shuffling [38].

***Achievement** 2:* Release of the ASCAD dataset that became a standard in the profiling SCA evaluation and proposing the hd5 format [52].

[9]It seems that DPAcontest datasets are not available anymore from June this year. We are checking if the datasets are made permanently unavailable or is this just a current issue.

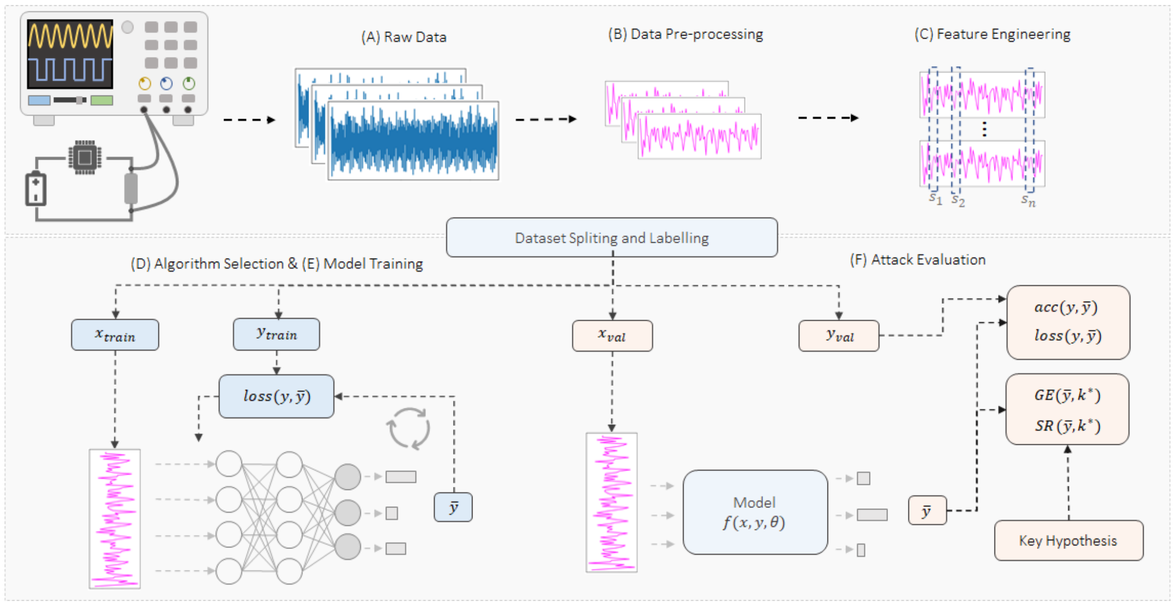

Fig. 3: Deep learning-based SCA flowchart. We do not explicitly write data standardization/normalization as this step is always done.

Dataset	Platform	Traces (Features)	Keys	Implementation	Countermeasures
AES_RD [39]	Atmel AVR (Software)	50 000 (3 500)	1 fixed key	AES128	Hiding (Random Delay Interrupt)
DPAv2 [40]	SASEBO GII (FPGA)	100 000 (3 253)	1 fixed key	AES128	None
DPAv4 [41]	Atmega (Software)	100 000 (435 000)	1 fixed key	AES256	First-order masking (RSM)
DPAv4.2 [42]	Atmega (Software)	80 000 (1 704 400)	16 different keys	AES128	1st-order masking (RSM)
AES_HD_MM [43]	SASEBO GII (FPGA)	5 600 000 (3 125)	1 fixed key	AES128	BM + Affine Masking
ASCADf [44]	Atmega (Software)	60 000 (100 000)	1 fixed key	AES128	1st-order BM (XOR)
ASCADv1 [44]	Atmega (Software)	300 000 (250 000)	Random keys + 1 fixed key	AES128	1st-order BM (XOR)
AES_HD [45]	SASEBO GII (FPGA)	50 000 (1 250)	1 fixed key	AES128	None
CHES_CTF 2018 [46]	STM32 (Software)	42 000 (650 000)	Random Keys + 3 fixed keys	AES128	1st-order BM (XOR)
Ed25519 (WolfSSL) [47]	STM32 (Software)	6 400 (1 000)	Random Ephemeral keys	EdDSA	None
Curve25519 (μNaCl) [48]	STM32 (Software)	5 997 (5,500)	Random Ephemeral keys	EdDSA	CSWAP, Coord./Scalar Rand.
Portability [49]	Atmega (Software)	50 000 (600)	4 fixed keys	AES128	None
ASCADv2 [50]	STM32 (Software)	810 000 (1 000 000)	Random + 1 fixed key	AES128	2nd-order BM + Hiding (Shuffling)
Curve25519 [51]	STM32 (Software)	300 (8000)	Random keys	EdDSA	CSWAP, Coord./Scalar Rand.
Curve25519 [51]	STM32 (Software)	300 (1000)	Random keys	EdDSA	CSPOINTER, Coord./Scalar Rand.

TABLE I: Publicly available datasets for side-channel analysis research. BM denotes Boolean masking. There are several more available datasets, but due to simple target, or lack of support (or any third reason), those datasets are not commonly used.

***Recommendation** 2:* While most of the SCA community uses a subset of the datasets listed above, it also happens that some research works use proprietary datasets. We advise against it unless the characteristics of that specific dataset are significantly different from those of publicly available datasets. We hope that the list in Table I will help researchers in understanding the differences among available datasets. For scenarios where new datasets are used, we consider it a must that those datasets are also made publicly available. Using a common evaluation base makes it possible to frame a proper comparison among different deep learning techniques.

***Recommendation** 3:* At the moment, various publicly available datasets are available from different sources. As hosting them takes effort in the long run, we believe there is a need for one central place where all datasets will be available. All the datasets should follow the same format. We suggest using the hd5 format as given for the ASCAD datasets.

***Challenge** 2:* As seen from the list of publicly available datasets, most of the targets are software implementations offering limited countermeasures (e.g., first-order masking and simulation of misalignment). Hence, deep learning attacks commonly easily break such targets and require significantly less than available traces. As a consequence, the gap between academia and industry that is using more realistic targets widens.

***Challenge** 3:* Researchers reported many neural network architectures that can successfully recover secret information without assuming any knowledge of the secret mask shares for the datasets mentioned above. The defined leakage models are constructed based on an intermediate value s, ignoring the fact that this value is masked with a secret share, r, by implementing a Boolean masking countermeasure: $s_r = s \oplus r$. The main conclusion is that neural networks inherently find the points of interest related to r and s_r and combine both of

them in the unmasking process.
The only exception is the ASCADv2 dataset, where an additional affine multiplicative share m masks the target value with the following operation: $s_{r,m} = m \times (s \oplus r)$. For this specific dataset (and at the moment of writing of this document), no successful attacks were reported demonstrating the possibility of key recovery without assuming the knowledge of at least one of the two secret shares m or r. This represents a challenge to be addressed in the future.

Recommendation *4:* Based on the list of publicly available datasets, it is clear that we need to consider protected hardware implementations and corresponding datasets. Since hardware implementations have better performance (than those in software), this necessitates having datasets containing the whole encryption or decryption function (compared with the current setup where only one round is given). For settings where multiple rounds execute in a single clock cycle, a potentially only way to attack is to consider inner rounds, see, e.g., [53], requiring appropriate datasets to evaluate the attacks. Available "hardware" datasets, such as AES_HD and AES_HD_MM, contain a fixed key, and therefore they do not represent realistic datasets for profiling SCA.

B. Data Pre-processing

Recall, we stated in Section I that one commonly mentioned advantage of deep learning techniques in SCA is that they do not require pre-processing. First, we notice that most of the works use either data normalization or standardization [27], [28], [26]. This is a common practice when working with neural networks, so it is not surprising to see it in the SCA domain, but we can already notice that the claim on no pre-processing is (technically) not true.

On the other hand, when dealing with side-channel measurements, it is common that traces will be misaligned due to the environment setup or countermeasures [54]. Interestingly, deep learning techniques (especially convolutional neural networks) work well even in the presence of misalignment (due to the spatial invariance property) [55], [54]. Thus, the alignment part of pre-processing does not seem to be required. Nevertheless, some researchers found the alignment step to be helpful even when using deep learning techniques [56]. In this sense, the model selection phase tends to be simplified if alignment is feasible.

In many SCA settings, the training set will not be sufficiently large due to the limitations of the data acquisition setup or countermeasures. In such scenarios, it is common to use data augmentation techniques. For instance, Cagli et al. showed how data augmentation in the form of data shifts (mimicking random delay countermeasure) and add-remove (mimicking jitter countermeasure) could significantly improve attack performance for datasets with hiding countermeasures [57].

Achievement *3:* Breaking protected datasets and showing that CNNs are naturally adept at defeating the desynchronization countermeasure [57].

Similar conclusions were also made by Pu et al., where they showed how data augmentation with shifts improves the attack performance [58]. Perin et al. utilized data shift data augmentation for side-channel analysis on ECC to achieve 100% accuracy [51]. This example is relevant as, without data augmentation, the authors reported that the attack would not be successful.

The data augmentation techniques discussed up to now significantly improved the SCA performance, and they were designed to simulate the effect of various hiding countermeasures. Nevertheless, it is also possible to consider data augmentation that is not SCA-specific. For example, researchers proposed to use a mixup data augmentation where several existing traces are combined (mixed up) to produce a new trace [59]. Providing a different approach, Picek et al. showed how noise addition to the data at the input of neural networks could improve the SCA performance due to the regularization effect [26]. Due to the binomial data distribution, the datasets are highly imbalanced for certain leakage models like the Hamming weight or Hamming distance. As such, this setting can cause difficulties for deep learning approaches to learn all possible classes equally. Picek et al. showed that a standard data balancing technique called SMOTE could significantly improve the attack performance [60]. Won et al. experimented with numerous variants of SMOTE, showing different variants to be more or less aligned with specific SCA datasets and implemented countermeasures [61]. More recently, researchers proposed to use generative adversarial networks (GANs) for data augmentation [62]. While the results look interesting, the main problems seem to be relatively poor performance when dealing with many labels and the need for a relatively large dataset to train GANs, making the whole procedure less useful in practice.

Wu et al. used a denoising autoencoder with clean and noisy data (where noise is the consequence of countermeasures) to pre-process the traces and improve the attack performance [38]. The results showed not only that deep learning attacks can improve the performance after this pre-processing step but also simpler profiling techniques like the template attack. Won et al. used multi-scale convolutional neural networks that can apply independent transformation (e.g., phase-only correlation, principal component analysis (PCA), alignment methods) to input data in each branch to extract the relevant features and allow a better generalization of the profiling model [63].

Challenge *4:* Data pre-processing can be useful and improve the deep learning SCA performance. Still, we are missing a clear set of guidelines on what pre-processing techniques to use and in what settings. It would be especially important to recognize if there are pre-processing techniques that should always be used (besides standardization/normalization).

Challenge *5:* Considering data augmentation techniques, we can recognize two directions: either use standard machine learning techniques or customize data augmentation for SCA. We believe there is a need for a systematic comparison of those approaches. Indeed, if custom approaches do not offer significant advantages, we question their need as they also require more effort to develop. On the other hand, if custom data augmentation is needed, there should be a clear set of recommendations on how to build synthetic traces for various settings.

***Recommendation** 5:* Data pre-processing is not a required step for SCA (besides data normalization/standardization), but such techniques could improve the SCA performance. We recommend always to use data standardization/normalization and report results with and without data augmentation. Still, the freedom not to use pre-processing is a significant advantage of deep learning over other types of profiling SCA.

C. Feature Engineering

Feature engineering represents (probably) the most important phase of classical profiling SCA. Since techniques like the template attack do not have any hyperparameters to tune, the key aspect of a successful attack is to use the most informative features. Commonly, to this end, one either uses feature selection (e.g., Pearson correlation or signal-to-noise ratio) or dimensionality reduction (e.g., PCA). Thus, this opens two potential issues: 1) what feature engineering technique to use (not necessarily a problem as there are only a few techniques commonly used) and 2) how many features to use? While recent work showed that many feature selection techniques behave similarly [64], making the selection of a feature engineering technique less crucial, there are no definitive pointers on the number of features to use. Most related works use 50 features, but there is no deeper reason for that choice (besides being relatively computationally efficient).

On the other hand, deep learning techniques are significantly more efficient and can work with more features. Techniques like neural networks also make implicit feature selection by assigning small weights to features that are not important. As such, the SCA community commonly claims to use raw data. Nevertheless, several commonly used (publicly available) datasets propose using a pre-selected feature window containing the relevant information. Thus, it would be fair to conclude that some feature engineering is used but not considered a part of the deep learning-based SCA.

More recently, feature engineering techniques based on deep learning showed potential. Ramezanpour et al. showed that an LSTM autoencoder could extract features from side-channel traces and obtain the secret information with ten times fewer traces than required for a direct attack like DPA [65]. The autoencoder-based approach for feature engineering also showed good results on other ciphers like ASCON [66]. Mukhtar et al. experimented with the PCA dimensionality reduction technique and showed that deep learning attacks on public-key implementations could result in a highly successful attack [67]. Wu et al. followed a different direction and proposed a triplet model to find highly efficient embeddings of input data [68]. Interestingly, the authors combined the deep learning-based feature engineering with a classical profiling attack (template attack) and showed that such combinations could rival the performance of state-of-the-art deep learning-based attacks. Finally, recent work showed significantly improved SCA performance if using raw data (thus, no pre-selected windows) [69]. With this work, the authors challenged the common practices in the deep learning-based SCA and demonstrated that using more features can bring advantages.

While the SCA community commonly considers that no feature engineering is required and all is left for implicit feature selection, they still use pre-selected windows of relevant data. A direct consequence of working on pre-selected and smaller windows is a trend of working with small neural network models. This, in the end, reduces the benefits of automation steps in SCA, where an appropriate model would automatically find relevant features inside raw traces. Also, the selection of hyperparameters of successful models becomes easier, which might slow down the advances in the hyperparameter tuning process.

***Challenge** 6:* Feature engineering is not needed for deep learning-based SCA, or we need only very basic techniques. Still, as it is common to use side-channel traces with thousands (or tens of thousands) features, we must investigate the possible drawbacks of using such extremely lengthy traces. Indeed, limited results indicate that more features help but also require using significantly larger neural networks, so a proper trade-off should be found.

***Challenge** 7:* If feature engineering is required, we must understand what techniques to use. Up to now, feature selection, dimensionality reduction with PCA, and autoencoders reported good results. Nevertheless, we are missing a systematic comparison of those techniques.

***Recommendation** 6:* Deep learning brings a significant advantage as it does not require feature engineering. We recommend using raw data (no feature engineering) in the deep learning-based SCA.

D. Algorithm Selection

The design of an efficient deep learning model is, for any application domain, the most difficult and laborious analytical step in the deep learning process [70]. Maghrebi et al. proposed the first comparison of deep learning models in the context of profiling SCA, showing that MLPs, CNNs, and stacked autoencoders exhibit similar or superior performance when compared to template attacks and machine learning methods [13].

***Achievement** 4:* The first application of convolutional neural networks for profiling SCA and demonstrating the potential of deep learning [13].

In, e.g., [13] and [54], the authors did not provide details about the reasoning for CNN hyperparameters selection [10]. The works presented in [26] and [52] selected CNN models based on the VGG structure, commonly applied to image and audio classification domains [71], [72]. This way, the results indicated that model selection based on a VGG-like structure could be efficient across several domains and datasets.

[10][13] mentioned they used genetic algorithms for hyperparameter tuning, but give no details.

***Achievement** 5:* Showing that VGG-like architectures perform very well in the SCA context and that one architecture can break different implementations [26].

Datasets evaluated in these first deep learning-based SCA publications [13], [54], [26], [52] are low-noise AES implementations, and the attacked trace intervals were optimized based on the access to the mask shares from the first-order Boolean masking. It is not surprising that various trained neural network models can efficiently recover the key within a few hundred attack traces. Moreover, even if evaluated datasets vary in the number of features and traces, nothing informative was provided as strong guidelines to select hyperparameters and define the appropriate model size (i.e., number of layers, neurons, and convolution filters).

To partially fill-up this gap, authors of [27] proposed a research direction focused on optimizing CNN models that are dataset-specific. One of the main goals from [27] was to demonstrate that CNN models could break previously evaluated public datasets with very small architectures and showing remarkable key recovery performance. What is more, the authors provided the first attempt of building a methodology for constructing CNNs for SCA.

***Achievement** 6:* Proposing a methodology to design small and well-performing CNNs for SCA [27].

Later, Wouters et al. [28] improved the previous architectures from [27] with a pooling-based dimensionality reduction layer as the input layer. This time, the required number of attack traces to recover the traces was reduced (slightly) further with the improved CNN models containing less than half of trainable parameters compared to original architectures proposed by [27]. An important takeaway from these two works is that model selection following a methodology to identify the smallest possible CNN architectures is an alternative to using (explicit) regularization in SCA [11].

Following the same concept of finding small and efficient neural network models for profiling SCA, in [73], the authors proposed, for the first time in the SCA domain, the application of reinforcement learning to select model hyperparameters. Their search system is rewarded based on attack performance (guessing entropy of correct key), and the decision process is based on defining the smallest possible CNN models. This work improved upon previous best results and showed that the considered datasets could be broken with even fewer traces, a task that sounded impossible only two years before.

Optimizing a neural network model to be as small as possible could face performance limitations if evaluated datasets become more complex (protected and noisy). As for other deep learning application domains, this would be a situation where the alternative solution is to explore larger architectures [74]. Along these lines, several works started to adopt methods for hyperparameter search on the same previously analyzed datasets (i.e., ASCAD, DPAv4, CHES CTF). A random search was adopted in [75] to define a set of CNN and MLP models to be later combined into ensemble models. In this case, the authors demonstrated that a random search usually results in both good and bad models and that combining them into a final ensemble provides better results than simply selecting the best model from the conducted search. Additionally, the authors showed that even randomly selected neural network architectures perform well, indicating that the commonly considered datasets are not very difficult to break. Wu et al. proposed the usage of Bayesian optimization search to define efficient MLP and CNN models [76]. Bayesian optimization is beneficial for large search space scenarios, and the training process is expensive, which is the case of hyperparameter search for profiling SCA. The authors showed that Bayesian optimization is more efficient than classical random search when the number of searches is limited and that it can give better results than even the reinforcement learning approach.

Although the search space in all the works described in the last paragraph is quite large, the authors always selected optimized ranges for each hyperparameter. This selection, of course, resulted from intuitive decisions related to the dataset and its countermeasures. In most cases, we rarely see deep learning models where the number of hidden layers goes beyond ten hidden layers. Exceptional cases that consider deeper models are [56], [69]. For deep learning standards (and for the evaluated datasets), a model can be considered small to medium size with such a number of layers. Even then, the search space is already quite large, and more efficient methods (e.g., Bayesian optimization, ensembles, reinforcement learning) are used to improve model selection. The main drawback of what was reported for hyperparameter search on a larger search space is that many models end up showing good attack performance, limiting the understanding of what are good hyperparameter options even for specific datasets or based on what criteria to compare the results.

***Challenge** 8:* As research papers consider relatively small datasets, there are no efficient guidelines to determine the hyperparameters on more realistic settings containing millions of noisy and protected side-channel traces. In this case, it is expected that the search space will increase, and the number of well-performing models will decrease significantly.

In the context of public-key implementations, Carbone et al. were the first to consider deep learning-based profiling SCA [77]. More precisely, the authors attacked a protected RSA algorithm (three countermeasures) and showed that CNNs have significant potential. Soon afterward, Weissbart et al. ran a deep learning attack against EcDSA in WolfSSL and showed it is possible to break the target with a single attack trace [78]. Interestingly, the authors used the same CNN as used in attacking AES [26]. Finally, Perin et al. used a fundamentally different approach to attack protected EcDSA implementation [51]. After running a horizontal attack, they use deep learning to iteratively correct errors stemming from the horizontal attack. With this approach, the authors managed to get 100% accuracy even if starting with only 52% accuracy.

Interestingly, despite attacking protected implementations and the need to succeed in a single attack trace, the results

[11]Regularization is a well-known technique that can prevent overfitting and, most of the time, improve generalization.

indicate this is relatively easily possible, while such attack performance for symmetric-key cryptography implementations is rarely seen. We believe this happens due to fewer classes for public-key implementations (commonly, only two).

***Challenge** 9:* At the moment of writing this paper, no publication demonstrated how to bypass high-order masking schemes in cryptographic algorithms. It is unclear what are the capabilities of various neural network architectures against different SCA countermeasures. Therefore, a study defining the limitations of such techniques against stronger countermeasures is missing.

***Recommendation** 7:* To allow more meaningful comparison, we first must define an optimized architecture:

Definition 1: A model that is optimized to recover a key with a minimum number of attack traces and at the same time can generalize in different conditions such as portability.

***Recommendation** 8:* We believe there should be an emphasis on reproducible research. Every phase in profiling SCA needs to be clearly defined. First, the pre-processing, feature engineering, and experimental setup must be discussed. Next, the researchers must clearly define the selected hyperparameters (as well as how those are found).
Evaluate attack performance with a different number of profiling and attack traces. Evaluate the performance of different profiling models. Publish the trained models.

E. Model Training

Model selection is accompanied by model training to infer if the chosen architecture delivers satisfactory performance for the given attack scenario. However, it is possible to improve the selection of the model if some training aspects are carefully defined based on common knowledge and user experience.

Among all tunable hyperparameters in a neural network, some are directly related to training performance. In SCA, learning rate, optimizer, batch size, number of epochs, or regularizers are usually less searched than other hyperparameters related to model structure (e.g., number of layers, neurons, and activation functions) but directly affect how the model learns. For instance, different optimizers work better for a different number of epochs, as reported in [79]. In this case, authors demonstrated that *Adam* or *RMSprop* require fewer epochs than *Adagrad*, *SGD*, or *Adadelta* to achieve good attack performance (although the last options tend to overfit less even for much longer trainings). More importantly, the authors observed that this behavior is common for any model hyperparameters when the number of trainable parameters is restricted to some bounds.

Training performance is also directly affected by the chosen loss function. In side-channel analysis, researchers predominantly consider categorical cross-entropy (while some works use MSE). Recently, custom loss functions were proposed in the context of profiling SCA. The authors of [80] proposed *Cross-Entropy Ratio* loss function to remove negative effects from imbalanced labels in the Hamming weight leakage model. *Ranking Loss* was proposed in [81] as a method to minimize the ranking error of the correct key concerning other key hypotheses. Later, Zaid et at proposed *Ensembling Loss* [82] as a custom loss function to maximize diversity in an ensemble-based approach. The implementation of custom loss functions directly impacts training time as it requires different computation aspects.

***Recommendation** 9:* Defining a loss function that provides general behavior for multiple datasets and attack models is hard. As such, the usage of categorical cross-entropy is still a safe choice, although it could present inferior performance compared to the loss function customized and validated to specific training data.

Fighting overfitting during training is usually done with regularization techniques. Most of the publications we analyzed are not affected by overfitting for three main reasons: 1) datasets are easy to attack, 2) selected models are often very small (i.e., a small fitting capacity that inherently regularizes the model), and 3) training epochs are usually set to small values (normally less than 300).

***Challenge** 10:* Recognize the most important hyperparameters for deep learning-based SCA. Evaluate how custom neural networks can enhance the attack performance and generalize for different settings.

***Recommendation** 10:* Setting efficient regularizers (e.g., dropout, *l1*, *l2*) is an alternative solution to determine the best number of epochs. As regularizers prevent model overfitting, the number of epochs can be set to a larger number without risking evaluating the model after generalization is already degraded. Moreover, overfitting should also be minimized by setting a learning rate scheduler. This way, the learning rate is adapted (manually or adaptively) during training.

F. Attack Evaluation

Attack evaluation considers different types of metrics depending on the target cryptographic algorithm. For symmetric-key algorithms, the key is fixed during multiple encryption executions, and therefore metrics from the predictions of multiple traces (class probabilities) are combined into a final metric. On the other hand, if the key needs to be recovered from a single measurement, which is the case of public-key algorithms, the model needs to be validated from a final metric that is not a combination of multiple trace predictions, such as accuracy.

The discrepancy between SCA and machine learning metrics was investigated in [60] for the case of AES. Authors demonstrated that imbalanced class problem leads to inconsistency between, e.g., accuracy and the SCA attack performance. This is more likely to be observed for side-channel measurements collected from protected or highly noisy implementations.

***Achievement** 7:* Investigating the difference between the machine learning and side-channel metrics and possible issues stemming from those differences [60].

Perin et al. visually showed how output class probabilities are ranked for both successful and unsuccessful attacks when accuracy is not enough to make a distinction between both situations [75]. The authors demonstrated that accuracy is low or close to a random guessing value because expected classes are not always predicted as first, but usually among the first ones, and the summation of these probabilities (based on the label's guessing for the correct key) is what explains the success of certain attacks. Thus, this confirms that GE is an appropriate metric for model validation. One of the drawbacks of GE is its computational complexity. Computing GE only once at the end of the training process is feasible. However, if GE validates the model during training (e.g., for early stopping), it might add excessive time overheads if the number of validation/attack traces is too large. In [83], the authors proposed the evaluation of the success rate of (small portions of) training and validation sets to determine the best training epoch. Still, it is not clear how to use the proposed approach when there are random keys in the profiling set. The work of [84] evaluated a mutual information-based metric to identify the best epoch to stop training. After that training moment, the authors observed that training performance tends to degrade.

Since GE is the most common metric in deep learning-based profiling SCA, one must ensure its computation is correct and reliable. Ideally, to draw consistent conclusions about the model's learnability, it is recommended to compute GE from the maximum possible number of attack traces. As GE is the average resulting vector of multiple key rank calculations, each key rank execution should be computed from a randomly selected trace subset from all available attack traces. This way, we ensure that GE is a measure of model generalization. This type of GE computation is explored in [85] and defined as *generalized guessing entropy*. Furthermore, computing GE using the geometric mean over multiple key ranks instead of the arithmetic mean (as commonly done) case seems to lead to more stable results [86]. Zhang et al. proposed a guessing entropy estimation algorithm based on theoretical distributions of the ranking score vectors [87]. This approach allows for better accuracy and efficiency than previous estimation techniques. While the authors did not discuss the deep learning perspective, it seems intuitive to use it in such context since the evaluation is expensive and needs to be done multiple times.

***Challenge** 11:* Little is understood about the relationship between commonly used SCA metrics (GE, SR) and model-learned parameters (i.e., the neural network weights). Metrics in SCA are all derived from output class probabilities, and everything "inside" the model is usually not considered. The research for specific metrics that could measure how well deep learning models understand and fool a countermeasure would be a perfect fit for a bi-objective problem, i.e., a model would be selected based on its ability to reach low guessing entropy and be efficient against certain countermeasures.

***Challenge** 12:* It is unlikely to find a universal profiling model that could defeat all types of available countermeasures and that could be used in a wide variety of targets. Therefore, we should measure and understand how the selected hyperparameters make the model succeed (or fail) in fitting existing leakages. For example, it would be beneficial to develop an additional metric that measures the level of desynchronization (or even noise) that the trained model can process.

***Challenge** 13:* Efficient attack evaluation has an appeal from the security perspective. A wrong attack evaluation based on an unreliable metric would mean an unreliable security assessment. In some cases, security can be overestimated not because the target is actually secure but because the evaluated metric is wrongly interpreted. Therefore, if such an evaluating metric indicates unsuccessful attack performance, the evaluator needs to cover at least two questions: 1) whether the model is trained correctly and the selected hyperparameters come from a reasonable process, and 2) whether the evaluation metrics are correctly estimated and interpreted for the specific target.

***Recommendation** 11:* We recommend using guessing entropy as the metric of choice to evaluate deep learning-based SCA. Furthermore, we recommend to always sample the attack traces from a large pool of available traces and averaging the estimation by using the median.

V. AI Explainability and SCA

In general, side-channel analysis is done to break a target and assess its security, but also to propose stronger countermeasures. However, using deep neural networks to break the targets leaves many questions open. Indeed, if the attack is not successful, we cannot understand why it was not successful. More precisely, it is difficult to know if we were unsuccessful due to a poor attack method or strong countermeasures (or both). On the other hand, if the attack is successful, one could (naively) think the goal is accomplished, but that is not the case. Once we succeed with an attack, a natural step should be to propose better countermeasures. Unfortunately, this kind of information is difficult to obtain from neural networks, and the security evaluator is left wondering how to strengthen the security of a target. Consequently, without understanding the security flaws of the target, expensive timing (i.e., random delays) or noise (i.e., extra logic) countermeasures could be added to the target, while a simpler solution directly mitigating the leakage could be more cost-efficient.

The first works in the direction of SCA and AI explainability aimed at interpreting neural network decisions by using heatmapping techniques [88]. The authors concluded that all tested techniques perform similarly and give relevant information about the most important features (that caused specific neural network decisions). Masure et al. used a sensitivity analysis technique called gradient visualization to pinpoint where information leakage happens [89]. Van der Val and Picek extended upon bias-variance decomposition and proposed GE bias-variance decomposition to understand the performance of machine learning techniques and how a change in a setting influences the SCA performance [90].

Van der Valk et al. made the first step toward explainability of deep neural networks in SCA by using the Singular Vector Canonical Correlation Analysis (SVCCA) tool to explain what neural networks learn while training on different side-channel datasets [34]. The obtained results were interesting as they showed that even datasets from different domains could have "more" similarity" than two side-channel datasets. Wu et al.

discussed how guessing entropy can be a misleading metric, and they proposed a new metric called profiling model fitting metric to estimate how reliable the guessing entropy estimation is [85]. Since the authors claimed that the newly proposed metric could also provide additional information about the generalization ability of the profiling model, we consider it also relevant from the AI explainability perspective.

Perin et al. used pruning to design smaller neural networks that exhibit good SCA behavior, and they showed that a recently proposed hypothesis called The Lottery Ticket Hypothesis also holds for the SCA domain [91]. While this work does not deal directly with AI explainability, the authors mentioned it as one of their main motivations since smaller neural networks should hopefully also be easier to explain. Finally, Wu et al. used ablation to explain how neural networks process hiding countermeasures [92]. They concluded that simpler countermeasures are getting processed in the shallow layers, while more complex countermeasures are processed in deeper layers. While the explainability part is limited to hiding countermeasures, we consider this work to show potential in explaining how neural networks deal with countermeasures.

***Challenge** 14:* Understand how neural networks process masking countermeasures.

***Challenge** 15:* Propose efficient countermeasures based on the AI explainability that are tuned to fight against deep learning-based SCA.

***Recommendation** 12:* Design neural networks that are as small as possible to allow easier interpretation.
Connect the results for AI explainability with the developments in the various phases discussed before.

VI. The Future of Deep Learning-based SCA

The recent work of F. Chollet defines measures for intelligence and generalization in artificial intelligence [93]. The author provides an important discussion to point out what a human-like intelligent system must provide to be considered intelligent for the terms defined in the paper. He suggests that, so far, deep learning has been restricted to specific tasks, and, as a consequence, this does not necessarily result in an intelligent system. In the SCA domain, the story is no different. *At the moment, it is more fair to assume that recent achievements also showed task-specific results, and the level of intelligence developed for SCA is still quite limited.*

However, this does not necessarily mean that deep learning-based SCA cannot go beyond this point. The state-of-the-art deep learning-based SCA results are limited to the concept of *local generalization*. According to [93], "this is the ability of a system to handle new points from a known distribution for a single task or a well-scoped set of known tasks". It is also defined as the "*adaptation to known unknowns within a single task or well-defined set of tasks*".

Once a deep neural network is trained from a profiling set, the scope of target operation (e.g., a target intermediate cipher state), leakage model selection, attacked trace interval (number of features), trace pre-processing, nature of side-channel leakage (e.g., power consumption, electromagnetic emissions, or time) and possible countermeasures are all well defined and taken into consideration to build the model. If a single aspect changes, the model loses its generalization ability unless the training data distribution shows statistical similarities to other targets and the trained model can cover them. In this case, the "*known unknown*" refers to a separate trace set measured from an identical target.

As an example, if a deep neural network is trained on a profiling set defined for exactly two classes (e.g., a byte contains all zeros or a byte contains all ones) and a separate test data is represented by a class that is different from these two classes (e.g., a byte containing Hamming weight equal to 3), the model cannot provide a generalization error associated to this third type of class. It can only indicate how likely this (unseen) data belongs to each of two well-defined classes. Therefore, for the SCA domain to enter a *broad generalization* case [93], the model should be able to cover *unknown unknowns*. An example would be the ability to generalize to leakage models not specifically covered during the training phase. This way, the model would be able to adapt to the new unknown leakage models. One option to (at least) approach this could be to use transfer learning [94].

While local generalization aspects limit research in AI-based side-channel analysis, every small improvement in the offensive side will require a correspondent defensive improvement to mitigate the new potential attack. Hence, the defensive side is also limited to threat models involving single and well-scoped tasks or attacks. Countermeasures are not developed to protect against adversaries dynamically, and they do not "generalize" to any possible attack. As side-channel analysis is a non-invasive attack, the target system is not aware of the presence of an adversary. This means that the protection cannot even adapt according to the attack's invasive effect. Therefore, a countermeasure needs to cover as many as possible *known unknowns* attack scenarios, restricting the dynamic and adaptation condition.

***Challenge** 16:* Design countermeasures that achieve broad generalization to cover *unknown* attack scenarios.

***Challenge** 17:* There are many developments for deep learning-based SCA. Consequently, it becomes difficult to recognize what technique to use and when. We require an automated system that will not only result in a strong attack but also suggest the steps required for the attack to be successful.

VII. Conclusions

Both academia and industry have been investigating side-channel analysis for several decades already. This process resulted in tremendous progress, but it also left many questions unanswered. The recent trend of using deep learning in SCA makes the situation even less straightforward. While the prevailing number of works report the new techniques resulting in improved attack performance, most of the techniques are not well justified and critically evaluated and thus remain neglected. As a consequence, it is difficult to comprehend the truly beneficial techniques and when to use them. We believe the researchers need to take a step back and observe the big picture, and that this work might help with that goal.

This SoK paper advocates the need for new threat models and signalizes many open challenges. Our analysis provides several recommendations that might help in addressing some of the challenges. Besides those recommendations, we recognize the need in facilitating the results' reproducibility: make publicly available datasets, write all the required details to reproduce the experiments, and publish the neural network models used in the experiments. In short, we hope this SoK will help other researchers and practitioners to pave new research avenues and move forward in this important domain.

References

[1] Knud Lasse Lueth, "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time," https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/, 2020, accessed August 4, 2021.

[2] 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.

[3] 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

[4] M. Barbosa, G. Barthe, K. Bhargavan, B. Blanchet, C. Cremers, K. Liao, and B. Parno, "Sok: Computer-aided cryptography," in *IEEE Symposium on Security and Privacy (S&P'21)*. IEEE Computer Society, 2021. [Online]. Available: https://prosecco.gforge.inria.fr/personal/bblanche/publications/BarbosaetalOakland21.pdf

[5] J. V. Monaco, "Sok: Keylogging side channels," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 211–228.

[6] I. Buhan, L. Batina, Y. Yarom, and P. Schaumont, "Sok: Design tools for side-channel-aware implementions," *CoRR*, vol. abs/2104.08593, 2021. [Online]. Available: https://arxiv.org/abs/2104.08593

[7] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Proceedings of CRYPTO'96*, ser. LNCS, vol. 1109. Springer-Verlag, 1996, pp. 104–113.

[8] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99. London, UK, UK: Springer-Verlag, 1999, pp. 388–397. [Online]. Available: http://dl.acm.org/citation.cfm?id=646764.703989

[9] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*, I. Attali and T. Jensen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 200–210.

[10] T. Roche, V. Lomné, C. Mutschler, and L. Imbert, "A side journey to titan," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 231–248. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/roche

[11] Common Criteria, "Supporting Document Mandatory Technical Document Application of Attack Potential to Smartcards," 2013, https://www.commoncriteriaportal.org/files/supdocs/CCDB-2013-05-002.pdf.

[12] O. Bronchain and F.-X. Standaert, "Side-channel countermeasures dissection and the limits of closed source security evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 2, pp. 1–25, Mar. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8542

[13] 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.

[14] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: a survey," *Journal of Cryptographic Engineering*, vol. 10, no. 2, pp. 135–162, Apr. 2019. [Online]. Available: https://doi.org/10.1007/s13389-019-00212-8

[15] S. Picek, "Challenges in deep learning-based profiled side-channel analysis," in *Security, Privacy, and Applied Cryptography Engineering - 9th International Conference, SPACE 2019, Gandhinagar, India, December 3-7, 2019, Proceedings*, ser. Lecture Notes in Computer Science, S. Bhasin, A. Mendelson, and M. Nandi, Eds., vol. 11947. Springer, 2019, pp. 9–12. [Online]. Available: https://doi.org/10.1007/978-3-030-35869-3_3

[16] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006, ISBN 0-387-30857-1, http://www.dpabook.org/.

[17] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," in *CHES*, ser. LNCS, vol. 2523. Springer, August 2002, pp. 13–28, San Francisco Bay (Redwood City), USA.

[18] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds. Cham: Springer International Publishing, 2014, pp. 253–270.

[19] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, Nov. 2014. [Online]. Available: https://doi.org/10.1007/s13389-014-0089-3

[20] A. Heuser and M. Zohner, "Intelligent machine homicide," in *Constructive Side-Channel Analysis and Secure Design*, W. Schindler and S. A. Huss, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 249–264.

[21] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, "Side-channel analysis and machine learning: A practical perspective," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4095–4102.

[22] 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.

[23] J. L. Massey, "Guessing and entropy," in *Proceedings of 1994 IEEE International Symposium on Information Theory*, 1994, pp. 204–.

[24] S. Picek, A. Heuser, G. Perin, and S. Guilley, "Profiling side-channel analysis in the efficient attacker framework," Cryptology ePrint Archive, Report 2019/168, 2019, https://eprint.iacr.org/2019/168.

[25] 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/

[26] 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.

[27] 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

[28] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 3, pp. 147–168, 2020.

[29] 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*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317934

[30] A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury, "Practical approaches toward deep-learning-based cross-device power side-channel attack," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 27, no. 12, pp. 2720–2733, 2019. [Online]. Available: https://doi.org/10.1109/TVLSI.2019.2926324

[31] J. Danial, D. Das, A. Golder, S. Ghosh, A. Raychowdhury, and S. Sen, "EM-X-DL: efficient cross-device deep learning side-channel attack with noisy EM signatures," *CoRR*, vol. abs/2011.06139, 2020. [Online]. Available: https://arxiv.org/abs/2011.06139

[32] H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova, "How diversity affects deep-learning side-channel attacks," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, 2019, pp. 1–7.

[33] 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*, A. Nitaj and A. Youssef, Eds. Cham: Springer International Publishing, 2020, pp. 337–357.

[34] 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*, G. M. Bertoni and F. Regazzoni, Eds. Cham: Springer International Publishing, 2021, pp. 175–199.

[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 *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[36] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 2, pp. 107–131, 2019. [Online]. Available: https://doi.org/10.13154/tches.v2019.i2.107-131

[37] J. Heyszl, A. Ibing, S. Mangard, F. D. Santis, and G. Sigl, "Clustering algorithms for non-profiled single-execution attacks on exponentiations," 2013.

[38] 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

[39] "AES_RD," Github repository, 2009, https://github.com/ikizhvatov/randomdelays-traces.

[40] T. TELECOM ParisTech SEN research group, "DPA_V2," Website, 2010, http://www.dpacontest.org/v2/.

[41] ——, "DPA_V4.1," Website, 2013, http://www.dpacontest.org/v4/.

[42] ——, "DPA_V4.2," Website, 2014, http://www.dpacontest.org/v4/42_doc.php.

[43] "AES_HD_MM," Website, 2014, https://chest.coe.neu.edu/?current_page=POWER_TRACE_LINK&software=ptmasked.

[44] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ASCAD database," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 53, 2018.

[45] Shivam Bhasin and Dirmanto Jap and Stjepan Picek, "AES_HD," Github repository, 2018, https://github.com/AESHD/AES_HD_Dataset.

[46] "CHES CTF 2018," Website, 2018, https://chesctf.riscure.com/2018/news.

[47] "Ed25519 WolfSSL," Website, 2019, https://github.com/leoweissbart/MachineLearningBasedSideChannelAttackonEdDSA.

[48] Ł. Chmielewski, "Reassure (h2020 731591) ecc dataset," Jan. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3609789

[49] "Portability Dataset," Website, 2020, http://aisylabdatasets.ewi.tudelft.nl/.

[50] Agence nationale de la scurit des systmes d'information (ANSSI), "ASCADv2," Github repository, 2021, https://github.com/ANSSI-FR/ASCAD.

[51] G. Perin, . Chmielewski, L. Batina, and S. Picek, "Keep it unsupervised: Horizontal attacks meet deep learning," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 1, pp. 343–372, Dec. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8737

[52] 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

[53] S. Swaminathan, L. Chmielewski, G. Perin, and S. Picek, "Deep learning-based side-channel analysis against aes inner rounds," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 981, 2021. [Online]. Available: https://eprint.iacr.org/2021/981

[54] 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.

[55] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf

[56] Y. Zhou and F.-X. Standaert, "Deep learning mitigates but does not annihilate the need of aligned traces and a generalized resnet model for side-channel attacks." *J. Cryptogr. Eng.*, vol. 10, no. 1, pp. 85–95, 2020.

[57] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing," in *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, ser. Lecture Notes in Computer Science, W. Fischer and N. Homma, Eds., vol. 10529. Springer, 2017, pp. 45–68.

[58] S. Pu, Y. Yu, W. Wang, Z. Guo, J. Liu, D. Gu, L. Wang, and J. Gan, "Trace augmentation: What can be done even before preprocessing in a profiled sca?" in *Smart Card Research and Advanced Applications*, T. Eisenbarth and Y. Teglia, Eds. Cham: Springer International Publishing, 2018, pp. 232–247.

[59] Z. Luo, M. Zheng, P. Wang, M. Jin, J. Zhang, H. Hu, and N. Yu, "Towards strengthening deep learning-based side channel attacks with mixup," *CoRR*, vol. abs/2103.05833, 2021. [Online]. Available: https://arxiv.org/abs/2103.05833

[60] 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 Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 209–237, Nov. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7339

[61] Y.-S. Won, D. Jap, and S. Bhasin, "Push for more: On comparison of data augmentation and smote with optimised deep learning architecture for side-channel," in *Information Security Applications*, I. You, Ed. Cham: Springer International Publishing, 2020, pp. 227–241.

[62] P. Wang, P. Chen, Z. Luo, G. Dong, M. Zheng, N. Yu, and H. Hu, "Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks," *CoRR*, vol. abs/2007.05285, 2020. [Online]. Available: https://arxiv.org/abs/2007.05285

[63] Y.-S. Won, X. Hou, D. Jap, J. Breier, and S. Bhasin, "Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3215–3227, 2021.

[64] S. Picek, A. Heuser, A. Jovic, and L. Batina, "A systematic evaluation of profiling through focused feature selection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2802–2815, 2019.

[65] K. Ramezanpour, P. Ampadu, and W. Diehl, "Scaul: Power side-channel analysis with unsupervised learning," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1626–1638, nov 2020.

[66] ——, "SCARL: side-channel analysis with reinforcement learning on the ascon authenticated cipher," *CoRR*, vol. abs/2006.03995, 2020. [Online]. Available: https://arxiv.org/abs/2006.03995

[67] N. Mukhtar, A. P. Fournaris, T. M. Khan, C. Dimopoulos, and Y. Kong, "Improved hybrid approach for side-channel analysis using efficient convolutional neural network and dimensionality reduction," *IEEE Access*, vol. 8, pp. 184 298–184 311, 2020.

[68] L. Wu, G. Perin, and S. Picek, "The best of two worlds: Deep learning-assisted template attack," Cryptology ePrint Archive, Report 2021/959, 2021, https://ia.cr/2021/959.

[69] 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, pp. 235–274, Jul. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8974

[70] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[71] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[72] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney,

R. J. Weiss, and K. Wilson, “Cnn architectures for large-scale audio classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[73] 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

[74] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[75] 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

[76] L. Wu, G. Perin, and S. Picek, “I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis.” *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1293, 2020.

[77] M. Carbone, V. Conin, M.-A. Cornlie, F. Dassance, G. Dufresne, C. Dumas, E. Prouff, and A. Venelli, “Deep learning to evaluate secure rsa implementations,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, pp. 132–161, Feb. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7388

[78] L. Weissbart, S. Picek, and L. Batina, “One trace is all it takes: Machine learning-based side-channel attack on eddsa,” in *Security, Privacy, and Applied Cryptography Engineering*, S. Bhasin, A. Mendelson, and M. Nandi, Eds. Cham: Springer International Publishing, 2019, pp. 86–105.

[79] G. Perin and S. Picek, “On the influence of optimizers in deep learning-based side-channel analysis,” in *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, ser. Lecture Notes in Computer Science, O. Dunkelman, M. J. J. Jr., and C. O’Flynn, Eds., vol. 12804. Springer, 2020, pp. 615–636. [Online]. Available: https://doi.org/10.1007/978-3-030-81652-0_24

[80] J. Zhang, M. Zheng, J. Nan, H. Hu, and N. Yu, “A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 73–96, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8583

[81] G. Zaid, L. Bossuet, F. Dassance, A. Habrard, and A. Venelli, “Ranking loss: Maximizing the success rate in deep learning side-channel analysis,” *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2021, no. 1, pp. 25–55, 2021. [Online]. Available: https://doi.org/10.46586/tches.v2021.i1.25-55

[82] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, “Efficiency through diversity in ensemble models applied to side-channel attacks - A case study on public-key algorithms -,” *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2021, no. 3, pp. 60–96, 2021. [Online]. Available: https://doi.org/10.46586/tches.v2021.i3.60-96

[83] D. Robissout, G. Zaid, B. Colombier, L. Bossuet, and A. Habrard, “Online performance evaluation of deep learning networks for profiled side-channel analysis,” in *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*, ser. Lecture Notes in Computer Science, G. M. Bertoni and F. Regazzoni, Eds., vol. 12244. Springer, 2020, pp. 200–218. [Online]. Available: https://doi.org/10.1007/978-3-030-68773-1_10

[84] G. Perin, I. Buhan, and S. Picek, “Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis,” Cryptology ePrint Archive, Report 2020/058, 2020, https://eprint.iacr.org/2020/058.

[85] L. Wu, L. Weissbart, M. Krek, H. Li, G. Perin, L. Batina, and S. Picek, “On the attack evaluation and the generalization ability in profiling side-channel analysis,” Cryptology ePrint Archive, Report 2020/899, 2020, https://eprint.iacr.org/2020/899.

[86] L. Wu, G. Perin, and S. Picek, “On the evaluation of deep learning-based side-channel analysis,” *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 952, 2021. [Online]. Available: https://eprint.iacr.org/2021/952

[87] Z. Zhang, A. A. Ding, and Y. Fei, “A fast and accurate guessing entropy estimation algorithm for full-key recovery,” *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 2, pp. 26–48, 2020.

[88] 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.

[89] 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

[90] D. van der Valk and S. Picek, “Bias-variance decomposition in machine learning-based side-channel analysis,” Cryptology ePrint Archive, Report 2019/570, 2019, https://eprint.iacr.org/2019/570.

[91] G. Perin, L. Wu, and S. Picek, “Gambling for success: The lottery ticket hypothesis in deep learning-based sca,” Cryptology ePrint Archive, Report 2021/197, 2021, https://ia.cr/2021/197.

[92] L. Wu, Y.-S. Won, D. Jap, G. Perin, S. Bhasin, and S. Picek, “Explain some noise: Ablation analysis for deep learning-based physical side-channel analysis,” Cryptology ePrint Archive, Report 2021/717, 2021, https://ia.cr/2021/717.

[93] F. Chollet, “On the measure of intelligence,” *CoRR*, vol. abs/1911.01547, 2019. [Online]. Available: http://arxiv.org/abs/1911.01547

[94] D. Thapar, M. Alam, and D. Mukhopadhyay, “Deep learning assisted cross-family profiled side-channel attacks using transfer learning,” in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 178–185.

[95] TELECOM ParisTech SEN research group, “DPA Contest (4th edition),” 2013–2014, http://www.DPAcontest.org/v4/.

[96] S. Bhasin, D. Jap, and S. Picek, “AES HD dataset - 500 000 traces,” AISyLab repository, 2020, https://github.com/AISyLab/AES_HD_2.

[97] ——, “AES HD dataset - 50 000 traces,” AISyLab repository, 2020, https://github.com/AISyLab/AES_HD.

[98] L. Masure and R. Strullu, “Side channel analysis against the anssis protected aes implementation on arm,” Cryptology ePrint Archive, Report 2021/592, 2021, https://ia.cr/2021/592.

[99] D. Kwon, H. Kim, and S. Hong, “Improving non-profiled side-channel attacks using autoencoder based preprocessing,” *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 396, 2020. [Online]. Available: https://eprint.iacr.org/2020/396

[100] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of CNN and RNN for natural language processing,” *CoRR*, vol. abs/1702.01923, 2017. [Online]. Available: http://arxiv.org/abs/1702.01923

[101] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, p. 17351780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[102] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2672–2680. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[103] M. Brisfors and S. Forsmark, “Dlsca: a tool for deep learning side channel analysis,” Cryptology ePrint Archive, Report 2019/1071, 2019, https://ia.cr/2019/1071.

[104] G. Perin, L. Wu, and S. Picek, “Aisy - deep learning-based framework for side-channel analysis,” Cryptology ePrint Archive, Report 2021/357, 2021, https://ia.cr/2021/357.

[105] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab, “Test vector leakage assessment (tvla) methodology in practice,” 2013.

[106] T. Moos, F. Wegener, and A. Moradi, “Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*,

vol. 2021, no. 3, pp. 552–598, Jul. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8986

[107] V. Cristiani, M. Lecomte, and P. Maurine, "Leakage assessment through neural estimation of the mutual information," in *Applied Cryptography and Network Security Workshops - ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19-22, 2020, Proceedings*, ser. Lecture Notes in Computer Science, J. Zhou, M. Conti, C. M. Ahmed, M. H. Au, L. Batina, Z. Li, J. Lin, E. Losiouk, B. Luo, S. Majumdar, W. Meng, M. Ochoa, S. Picek, G. Portokalidis, C. Wang, and K. Zhang, Eds., vol. 12418. Springer, 2020, pp. 144–162.

Appendix

Publicly Available SCA Datasets

DPAcontest v2 dataset (**DPAv2**) [40] is rarely used in both machine learning and deep learning settings. We believe this is because it is not a protected implementation.
DPAcontest v4 dataset (**DPAv4**) [41] represented a standard target for simpler machine learning methods, see, e.g., [95], [19]. However, it is not often used with deep learning methods as it is a very easy dataset to break. Also, the dataset authors reported a problem with the implemented countermeasure and first-order leakages in the traces.
DPAcontest v4.2 dataset (**DPAv4.2**) [42] was released to fix the problems found in DPAv4. Unfortunately, this dataset is rarely used in the deep learning-based SCA [12].
AES_HD dataset [45] is an unprotected but very noisy dataset, so it represents a much more challenging target than, e.g., DPAv4. Still, it is not widely used as it is unprotected and uses the same key for profiling and attack sets.
There is an extended version of the dataset with 500 000 traces [96] [97].

The Random Delay dataset (**AES_RD**) [39] is a protected dataset but straightforward to break with deep learning (especially convolutional neural networks due to their spatial invariance property), so it is not used in the last few years.

The **ASCAD** datasets [44] (two versions, one with fixed key - **ASCADf** and one with random keys **ASCADv1**) represent de-facto standards when evaluating deep learning-based SCA. Still, considering the developments in the domain, there are relatively easy to break, and the version with random keys is not significantly more complex than the fixed key version. Finally, these datasets also introduced novelties from the data management perspective as they used the hd5 format.
The new version of the ASCAD dataset (**ASCADv2**) [50] appeared only recently, so there are not many results with it [98]. The available information indicates that the dataset can be challenging to attack if assuming no knowledge of secret shares or easy to attack if shares are known.

CHES CTF dataset [46] is an interesting dataset that is more difficult to attack than ASCAD, but due to the lack of support, the publicly available version has only 10 000 traces per profiling set, making it less used recently (due to the reproducibility issues).
An extended version of this dataset, with some pre-processing, is available at http://aisylabdatasets.ewi.tudelft.nl.

The **Portability** dataset [49] is a software implementation without countermeasures. It is straightforward to attack it, making it a less attractive target for deep learning. At the same time, this is (as far as we are aware) the only publicly available dataset to investigate portability effects for AES [25].

This is a masked hardware AES implementation (**AES_HD_MM**) dataset [43] that is not widely used with deep learning techniques. We postulate that a possible reason for it is that the dataset is older, so many researchers are not aware of its availability.

In the case of available datasets based on public-key implementations, **Ed25519 (WolfSSL)** [47] is unprotected (making it a non-realistic target) [78], but the lack of publicly available datasets makes it one of the rarely available datasets considering public-key implementations.

Curve25519 μNaCl dataset [48] is a protected implementation, making it a (somewhat) realistic target, but there are only a few available results. We believe this is because most SCA community investigates deep learning attacks on block ciphers (AES).

Curve25519 dataset [51] are protected implementations of EdDSA, differing in the number of features and countermeasures. These datasets appeared very recently, so there is only one paper using them.

Side-channel Analysis Flowchart

A general workflow for SCA can be summarized as follows:

1) Consider a device performing a cryptographic computation $CF_{k^*}(\mathbf{a})$ on different inputs $\mathbf{a}$ drawn uniformly from the input space, using some fixed key k^* drawn uniformly from the keyspace $\mathcal{K}$.
2) During this computation, the device will handle some intermediate values $T_{k^*,\mathbf{a}}$ utilizing the known input $\mathbf{a}$ and the unknown key k^*.
3) When such a sensitive intermediate value is computed, the device generates some physical leakage, denoted $W_{k^*,\mathbf{a}}$.
4) To perform a key recovery, an adversary must select a sensitive value depending on the secret key k^*. Then, the adversary can evaluate its result for the input used to generate $W_{k,\mathbf{a}}$ and all the key candidates $k \in \mathcal{K}$. This will result in different hypothetical values $Z_{k,\mathbf{a}}$.
5) The adversary uses a leakage model to map these values from the original space $\mathcal{T}$ into a hypothetical leakage space $\mathcal{Z}$.
6) The adversary uses a statistical tool called a distinguisher to compare the different models $Z_{k,\mathbf{a}}$ with the actual leakages $W_{k,\mathbf{a}}$. If the attack is successful, the best comparison

[12]There are also other versions of the DPAcontest datasets (v1 and v3), but to the best of our knowledge, they were never used in the context of deep learning attacks

result (i.e., the highest value of the distinguisher) should be obtained for the correct key candidate.

Common Neural Network Types in SCA

Next, we briefly describe several types of neural networks used in the deep learning-based SCA. The first two types (CNNs and MLP) are commonly used, while autoencoders are not widely used, despite being investigated in different contexts. Finally, the last three neural network types are rarely used (one paper for RNN, two for ResNet, and two for GANs).

a) Multilayer Perceptron: The multilayer perceptron is a feed-forward neural network mapping input sets onto sets of appropriate outputs. MLP consists of multiple layers of nodes in a directed graph, where each layer is fully connected to the next one, and training of the network is done with the backpropagation algorithm [70].

b) Convolutional Neural Networks: Convolutional neural networks commonly consist of three types of layers: convolutional layers, pooling layers, and fully connected layers [70]. The convolution layer computes the output of neurons connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. Pooling decrease the number of extracted features by performing a down-sampling operation along the spatial dimensions. Finally, the fully connected layer computes the hidden activations or the class scores.

c) Autoencoders: Autoencoders are neural network structures composed essentially of two main blocks: encoder and decoder. The output of the encoder usually contains a reduced dimension in comparison to the dimension of input data. The decoder block takes the encoder output as input and outputs the same dimension of input data. Autoencoders can be employed in supervised and unsupervised learning, where the main purpose is to transform input data as part of the (but not necessarily) pre-processing phase. There are several types of autoencoders used in deep learning-based SCA. One type is an autoencoder to remove noise from side-channel traces [38], [99]. The second type is autoencoder models for feature extraction, reducing the dimensionality of the input data [63], [68]. Autoencoders were also used for classification [13], where the authors placed a *softmax* layer as the output layer.

d) Recurring Neural Networks: Recurrent Neural Networks (RNNs) are designed to recognize patterns in sequences of data [70]. RNNs performs the same function on every input data and stores the output for the following input. This means that RNN stores the previous step input and merges that information with the current step input. The output of the neural network is thus dependent on past computation. Because the RNN has a memory, it is especially useful in training on sequential data (a stream of interdependent data). RNNs are widely used in the natural language processing domain (NLP) [100], and LSTM is a well-known example of RNN-based architecture [101], also used in SCA ([13]).

e) Residual Neural Networks: Residual Neural Networks (ResNet) consist of residual blocks with several layers and a shortcut connection [74]. The shortcut connection connects the input and output of each residual block. Each residual block contains basic layers of any type (e.g., convolution, pooling, dense). The main purpose of shortcuts in a residual network is to avoid that deeper networks with several hidden layers become so deep that they cannot propagate the information anymore.

f) Generative Adversarial Networks: Generative adversarial networks are specific neural network-based structures that can generate artificial data based on an evaluated data distribution [102]. Two types of models are trained: a generator and a discriminator. The generator takes random data as input and tries to create output data indistinguishable from the original and available distribution. This generator model is trained until the discriminator model cannot differentiate from which distribution (original or fake) the input is coming.

Publicly Available Frameworks

To the best of our knowledge, there are not many publicly available frameworks for deep learning-based side-channel analysis. While having a publicly available framework is not necessary for conducting deep learning-based SCA, we consider it very helpful, especially from the reproducibility perspective. The first published framework we are aware of is developed by Brisfors and Forsmark and is called DLCSA [103][13]. The tool is meant to be modular, but the current version supports only basic functionalities. It seems there is no active development in the last two years.

The second publicly available framework is called the AISY framework. This framework is recently developed by Perin et al. [104], and it contains a significant number of functionalities developed in the last few years in the deep learning-based SCA domain [14]. The main advantage of this framework seems to be the relative ease of conducting reproducible research.

***Recommendation** 13:* The SCA community would benefit from a publicly available tool that is regularly updated and containing up-to-date functionalities. Naturally, it is not easy to know what functionalities to support, but we propose to include those published in top conferences. As this would still represent a significant effort for the authors of the framework, we propose establishing a standard coding style so that it is easy to import functionalities from various research works.

Leakage Assessment and Deep Learning

A standard option to verify that the side-channel traces contain exploitable information is to conduct leakage assessment, e.g., TVLA [105]. This step only reveals if there is a leakage and not how to exploit it. Leakage assessment is far from perfect as it can have many false positives and false negatives. *While this step is not a part of deep learning-based SCA, deep learning can be used to assess leakage.* Differing from deep learning-based SCA, there does not seem to be many

[13] github.com/brisfors/DLSCA
[14] https://github.com/AISyLab/AISY_Framework

works considering deep learning and leakage assessment. A recent effort used deep learning to distinguish between two groups of data (fixed-vs-random or fixed-vs-fixed) [106]. [107] used MINE (mutual information neural estimator) to compute mutual information in high dimensional side-channel traces. The obtained results showed that this technique could be used as a simple tool to obtain an objective leakage evaluation.

Challenge *18:* Due to very limited results, it is still not clear what are (if any) the advantages of deep learning for leakage assessment. What is more, the model selection phase is very simple in the research published up to now, indicating that more investigation is needed.