SoK

Deep Learning-based Physical Side-channel Analysis

Picek, Stjepan; Perin, Guilherme; Mariot, Luca; Wu, Lichao; Batina, Lejla

DOI
10.1145/3569577

Publication date
2023

Document Version
Final published version

Published in
ACM Computing Surveys

Citation (APA)
Picek, S., Perin, G., Mariot, L., Wu, L., & Batina, L. (2023). SoK: Deep Learning-based Physical Side-channel Analysis. *ACM Computing Surveys*, *55*(11), Article 227. https://doi.org/10.1145/3569577

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

SoK: Deep Learning-based Physical Side-channel Analysis

STJEPAN PICEK, GUILHERME PERIN, and LUCA MARIOT, Radboud University, The Netherlands
LICHAO WU, Delft University of Technology, The Netherlands
LEJLA BATINA, Radboud University, The Netherlands

Side-channel attacks represent a realistic and serious threat to the security of embedded devices for already almost three decades. A variety of attacks and targets they can be applied to have been introduced, and while the area of side-channel attacks and their mitigation 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 the **systematization of knowledge (SoK)** even more critical.

We first dissect deep learning-based side-channel attacks according to the different phases they can be used in and map those phases to the efforts conducted so far in the domain. For each phase, we identify the weaknesses and challenges that triggered the known open problems. We also connect the attacks to the threat models and evaluate their advantages and drawbacks. Finally, we provide a number of recommendations to be followed in deep learning-based side-channel attacks.

CCS Concepts: • **Security and privacy → Cryptanalysis and other attacks**; **Side-channel analysis and countermeasures**; *Embedded systems security;*

Additional Key Words and Phrases: Side-channel attacks, deep learning, profiling attacks, supervised learning, challenges, recommendations

ACM Reference format:
Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. 2023. SoK: Deep Learning-based Physical Side-channel Analysis. *ACM Comput. Surv.* 55, 11, Article 227 (February 2023), 35 pages.
https://doi.org/10.1145/3569577

1 INTRODUCTION

The embedded devices market is constantly growing. Already in the 2020 world of connected devices, there were more IoT (e.g., cars, smart home devices, connected industrial equipment)

This work received funding in the framework of the NWA Cybersecurity Call with project name PROACT with project number NWA.1215.18.014, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). Additionally, this work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019).

Authors' addresses: S. Picek, G. Perin, L. Mariot, and L. Batina, Radboud University, Postbus 9010, Nijmegen, 6500 GL, The Netherlands; emails: {stjepan.picek, guilherme.perin, luca.mariot}@ru.nl, lejla@cs.ru.nl; L. Wu, Delft University of Technology, Delft, 2628 XE, The Netherlands; email: L.Wu-4@tudelft.nl.

0360-0300/2023/02-ART227 $15.00
https://doi.org/10.1145/3569577

connections than non-IoT (e.g., smartphones, laptops, computers) 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 [49]. Such an escalation in the number of devices is followed by increased security concerns and vulnerabilities, resulting in the ever-growing demand for certified products. Consequently, millions of products are undergoing strict security assessments in evaluation labs worldwide on a daily basis [3]. As a part of those evaluations, **side-channel attacks (SCA)** (or analyses) represent a well-known threat since the 90's [51] and are widely studied by researchers in the community of information security and cryptography from both industry and academia [4, 16, 65].

Various physical leakages such as timing delay [50], power consumption [52], and **electromagnetic emanation (EM)** [89] become available during the device's computation with the (secret) data. Those leakages have led to a whole new research area: 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.

Side-channel attacks and corresponding mitigations evolved in the past few decades, and more recently, deep learning-based side-channel analysis became widely used in the SCA community. 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 [97]. While practically not so relevant, the project's main goal was to increase awareness when considering the worst-case adversaries and make such techniques well understood. For example, the 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 [24]. In short, confidence in security evaluation implies considering the strongest adversary, which relates to the choice of attack techniques [15].

The appeal and popularity of using deep learning in side-channel analysis in the last few years are evident, as demonstrated in Figure 1. More precisely, we found 183 papers that investigate **deep learning-based side-channel analysis (DL-SCA)** in the last six years. Clearly, from 2016 when the first paper appeared that used deep learning to conduct side-channel analysis [58], 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-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 the different approaches are 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 step in understanding state-of-the-art deep learning-based SCA.

Some related works already itemized various machine learning-based side-channel attacks, see, e.g., works by Hettwer et al. [40] or S. Picek [81]. This systematization of knowledge paper

[1]More precisely, Maghrebi et al. were the first to use convolutional neural networks [58]. There are earlier papers that use multilayer perceptrons. Still, since they do not report the number of hidden layers or that number is set to one, we do not consider those works as deep learning-based SCA.

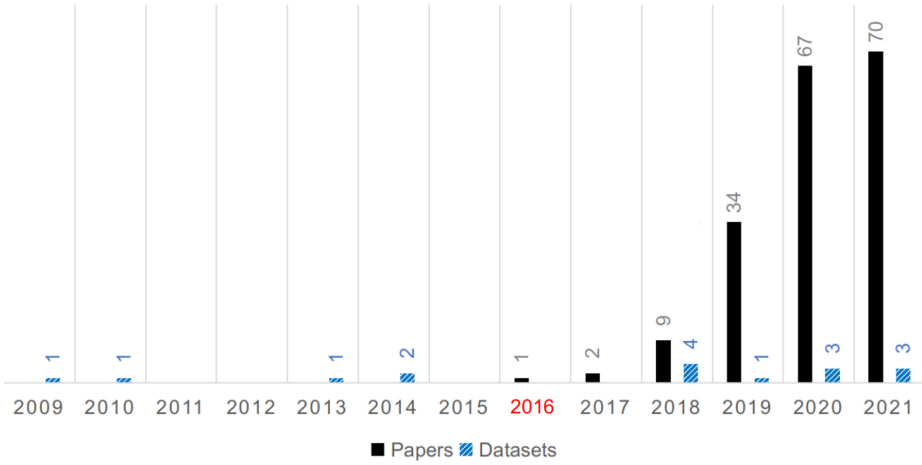

Fig. 1. The distribution of papers and datasets per year that are used by deep learning-based side-channel analysis. For multiple versions of the same papers, we consider the peer-reviewed version. We consider papers published in English only. Observe how datasets appear more consistently (and in larger numbers) in the last few years, which we connect with a considerable interest in the SCA community for deep learning-based SCA.

goes far beyond those works by providing a detailed list of up-to-date challenges with the latest developments in the domain and recommendations on how to address them. This paper does not aim to cover every proposed approach but systematically identifies and analyzes the main approaches in getting deep learning to work for side-channel adversaries. We focus on attacks exploiting the power and EM radiation as the most commonly exploited types of leakage in embedded and IoT devices.

In short, with this **systematization of knowledge (SoK)** paper, we make the following contributions:

(1) Discuss threat models and identify the differences among them and the impact of those in practice.
(2) Enlist relevant phases in deep learning-based SCA and the impact of neglecting those phases.
(3) Evaluate and compare approaches proposed in terms of performance.
(4) Pinpoint why many proposed solutions are not adopted in practice and what are the necessary criteria for new solutions.

Our analysis recognized four threat models. There, it is especially challenging to make the unsupervised deep learning threat model practical. Next, we divided the deep learning process into six phases and provided 18 challenges to be addressed in future research. We discussed 23 recommendations to address the challenges and improve the state of the art. Finally, we recognized seven works that have a significant impact on the deep learning-based SCA.

The rest of this paper is organized as follows. In Section 2, we provide necessary background information about side-channel attacks, where we emphasize deep learning-based attacks. Next, in Section 3, we provide an overview of different threat models commonly used. Section 4 discusses various phases of supervised learning and connects them with important results in the SCA research. Section 5 provides an overview of previous work in AI explainability for SCA, and in Section 6, we briefly discuss publicly available frameworks for deep learning-based SCA. Section 7 provides an overview of the future of deep learning-based SCA, and in Section 8, we conclude the paper. Finally, in Appendix A, we provide additional information about the deep learning architectures commonly used in the profiling SCA.

2 BACKGROUND

We use calligraphic letters like $\mathcal{X}$ to denote sets. The corresponding lower-case letters x represent elements of $\mathcal{X}$ or realizations of a random variable with values in $\mathcal{X}$. Variables denoted in bold such as $\mathbf{x}$ are vectors over the ground set $\mathcal{X}$, e.g., if $\mathcal{X} = \mathbb{R}$ and $n \in \mathbb{N}$, then by $\mathbf{x}$ we mean a vector in the n-dimensional Euclidean space $\mathbb{R}^n$.

A cryptographic computation is commonly formalized as a mapping from a space of plaintexts $\mathcal{P}$ to a space of ciphertexts $\mathcal{C}$, parameterized with a keyspace $\mathcal{K}$. These three spaces are usually sets of vectors over a common alphabet $\mathcal{A}$ (most often $\mathcal{A} = \{0, 1\}$). For example, in block ciphers, plaintexts and ciphertexts have the same block length $b \in \mathbb{N}$, so that $\mathcal{P} = \mathcal{C} = \mathcal{A}^b$.

A dataset $\mathcal{D}$ represents a set of side-channel measurements, also called traces, collected from a device during the execution of a cryptographic computation. A trace is a time sequence of $F \in \mathbb{N}$ features (samples, points of interest). Thus, assuming that a single feature is a value ranging in a set $\mathcal{M}$, a trace is represented by a vector in $\mathcal{M}^F$. Each trace $\mathbf{d}_i \in \mathcal{D}$ is associated with an input value $\mathbf{a}_i$ (plaintext or ciphertext)[2] and a key $\mathbf{k}_i$, where $\mathbf{k}_i$ denotes a generic key candidate, while the correct key is $\mathbf{k}^*$. The keys take their values from the keyspace $\mathcal{K}$.

2.1 Side-channel Analysis

The side-channel analysis considers attacks that do not aim at the weaknesses of the algorithms but their implementations [59]. 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 the data most likely to have been processed. In practice, SCA requires the ability to model the leakage (to come up with the predictions for data) and to have a good comparison tool (so-called distinguisher) to extract the secret information efficiently.

Formally, a general workflow for SCA can be summarized as follows:

(1) Consider a device performing a cryptographic computation $CF_{\mathbf{k}^*}(\mathbf{a})$ on different inputs $\mathbf{a}$ drawn uniformly random from the input space, using some fixed key $\mathbf{k}^*$ drawn uniformly at random from the keyspace $\mathcal{K}$.
(2) During the cryptographic computation, the device will handle some intermediate values $T_{\mathbf{k}^*,\mathbf{a}}$ utilizing the known input $\mathbf{a}$ and the unknown key $\mathbf{k}^*$.
(3) When such a sensitive intermediate value is computed, the device generates some physical leakage denoted $W_{\mathbf{k}^*,\mathbf{a}}$.
(4) To perform a key recovery, an adversary must select a sensitive value depending on the secret key $\mathbf{k}^*$. Then, the adversary can evaluate its result for the input used to generate $W_{\mathbf{k},\mathbf{a}}$ and all the key candidates $\mathbf{k} \in \mathcal{K}$. This evaluation will result in different hypothetical values $Z_{\mathbf{k},\mathbf{a}}$.
(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 $Z_{\mathbf{k},\mathbf{a}}$ with the actual leakages $W_{\mathbf{k},\mathbf{a}}$. If the attack is successful, the best comparison result (i.e., the extreme value of the distinguisher) should be obtained for the correct key candidate.

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) 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.

[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.

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. Thus, the approach described above is repeated for each sub-key until the (full) key $\mathbf{k}^*$ is recovered. In general, if one sub-key can be recovered, it is enough to argue for the implementation's weakness.

2.1.1 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 a (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)** [52].[3] While direct attacks assume less powerful attackers (since the attackers do not have access to an identical copy of a device), such attacks might require millions of measurements to break the target and 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 to attack the target device. The profiling attack happens in two stages, commonly denoted as the profiling and the attack stages.

2.1.2 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 [5, 19, 23]. The profiling stage 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 stage, 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. The profiling attack assumes a more powerful attacker than a direct one, and it may require significantly fewer attack traces than direct attacks to break the target. Sometimes, only one trace is sufficient in a profiling attack. Note that the two stages of the profiling attacks correspond to the two stages of supervised machine learning: training and testing.

Machine learning techniques were adopted for profiling attacks, where 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 for this work, we list several references for interested readers [42, 55, 84]. Additionally, we recommend a survey of machine learning-based side-channel analysis by Hettwer et al. [40]. The main differences between our work and the one by Hettwer et al. [40] are: (1) we investigate only deep learning-based SCA while Hettwer et al. also considered other machine learning techniques, as well as different-than-SCA approaches, (2) Hettwer et al. wrote their work in the early days of deep learning-based SCA (early 2018), so

[3]Besides power, it is common to use other side channels like EM, which changes 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.

it covers only a few deep learning works, and (3) Hettwer et al. made a survey work that gives detailed information about related works but not a systematization of knowledge [40].

2.1.3 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 [59].

In masking, a random value (mask) is generated to conceal every intermediate value. More precisely, the computation is performed on masked data instead of actual data, and the computation procedure removes the correlation between the measurements and the secret data. Two commonly used types of masking are Boolean masking and arithmetic masking, which refer to the way "a mask" is added to the input and other intermediate variables [59].

On the other hand, the goal of hiding is to make measurements appear 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 having an unlimited number of available side-channel measurements.[4] Similarly, masking countermeasures 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 stage. The knowledge of the mask shares is a common assumption in the white box attack, as we assume the attacker knows the exact places where the leakage occurs. A mask share denotes a randomly selected value combined with the sensitive variable. The common ways to combine the mask and the sensitive value are addition and multiplication. Commonly, highly protected targets consider using high-order masking schemes or a combination of masking and hiding countermeasures.

2.1.4 SCA Adversary. There are two types of adversaries in SCA: 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 a side-channel attack.

2.2 Deep Neural Networks

A deep neural network typically classifies a signal characterized by $F \in \mathbb{N}$ features into one of $C \in \mathbb{N}$ classes (or labels). Both features and labels are usually represented as compact subsets of the n-dimensional Euclidean space. Formally, let us define $\mathcal{X} \subseteq \mathbb{R}^F$ as the domain of input features and $\mathcal{Y} \subseteq \mathbb{R}^C$ as the space of output classes.

A neural network is a mapping $f : \mathcal{X} \to \mathcal{Y}$, which takes a vector of input features $\mathbf{x} \in \mathcal{X}$ and predicts the output class to which $\mathbf{x}$ belongs. In particular, the predicted class is represented as a vector $\mathbf{y} \in \mathcal{Y}$ through one-hot encoding: assuming that f predicts the class $i \in \{1, \ldots, C\}$ for $\mathbf{x}$, the output vector y has 1 in the i-th coordinate, while it is 0 everywhere else.

The neural network transforms the input features vector $\mathbf{x}$ to its predicted label as $\mathbf{y} = \sigma(\varphi(\mathbf{x}))$. The function $\varphi : \mathcal{X} \to \mathcal{Y}$ is a composition of $l \in \mathbb{N}$ layers, that is, $\varphi = \varphi_l \circ \varphi_{l-1} \circ \cdots \circ \varphi_2 \circ \varphi_1$, where for all $i \in \{1, \ldots, l\}$ the i-th layer is a function $\varphi_i : \mathcal{X}_i \to \mathcal{Y}_i$. In particular one has $\mathcal{X}_1 = \mathcal{X}$ and $\mathcal{Y}_l = \mathcal{Y}$, while in all other cases $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ and $\mathcal{Y}_i \subseteq \mathbb{R}^{m_i}$, with $n_i, m_i \in \mathbb{N}$. Given the input $\mathbf{x}_i \in \mathcal{X}_i$, which is the output computed by the previous layer $i - 1$, the function of layer i is defined as

[4] In reality, there are no attackers unbounded in terms of the number of available side-channel measurements, but it is common to assume the attacker to be arbitrarily powerful.

$\varphi_i(\mathbf{x}_i) = \phi_i(\mathbf{w}_i \cdot \mathbf{x}_i + \mathbf{b}_i)$. Here, ϕ_i represents a nonlinear activation function (such as the **Rectified Linear Unit, or ReLU)**, and $\mathbf{w}_i$ and $\mathbf{b}_i$ are respectively the vectors of weights and biases for layer i (the trainable parameters).

The components of the output vector $\varphi(\mathbf{x})$ are also called logits, and the softmax function $\sigma : \mathcal{Y} \rightarrow [0,1]^C$ is applied to first convert these logits into a probability distribution. Then, the softmax function computes the argument $c \in \{1, \ldots, C\}$ that maximizes the resulting distribution. The class (or label) of the input $\mathbf{x}$ inferred by the network f is then the one-hot encoding vector $y \in \mathcal{Y}$ of c.

2.3 Deep Learning-based Side-channel Analysis

This paper 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 $\mathbf{y}_c$ is derived from the key $\mathbf{k}^*$ and input $\mathbf{a}_i$ through a cryptographic function CF and a leakage model LM.

The objective of a learning algorithm Alg is to learn a parameterized $f_\theta \in \mathcal{H}$, 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, $f_\theta \in \mathcal{H}$ 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_θ, 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 stages of the profiling attack. Then, the classification process can be broken down into the following steps:

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

Since the underlying data distribution $\mathcal{X} \times \mathcal{Y}$ is not known, a supervised learning algorithm uses the training set $\mathcal{D}_{tr}$ of size N to learn a hypothesis f_θ that minimizes the empirical risk $\frac{1}{N}\sum_i^N L(\mathbf{y}_i, \hat{\mathbf{y}}_i)$, where $(\mathbf{x}_i, \hat{\mathbf{y}}_i) \sim \mathcal{D}_{tr}$. In other words, the empirical risk is the expected value of the loss function over all realizations of the joint distribution $\mathcal{X} \times \mathcal{Y}$.

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,[5] 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, the training stops, and the neural network, along with its parameters, is stored as f_θ where f_θ represents the SCA profiling model.

2.3.2 Test. In the test stage,[6] the goal is to predict labels y based on previously unseen traces $\mathbf{x} \in \mathcal{D}_{ts}$, and the trained model f_θ. The SCA metrics, such as guessing entropy and success rate, are used to assess the test outcomes, as discussed in the next section.

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

2.4 Evaluation of Side-channel Analysis

The outcome of predicting with a model f_θ on the test set $\mathcal{D}_{ts}$ is a two-dimensional matrix P with dimensions $Q \times C$, where Q is the number of available attack traces. The cumulative sum $S(\mathbf{k})$ for any key byte candidate $\mathbf{k}$ 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 $S(\mathbf{k}) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,\mathbf{y}})$. The value $\mathbf{p}_{i,\mathbf{y}}$ denotes the probability that for a key $\mathbf{k}$ and input $\mathbf{a}_i$, we obtain the label $\mathbf{y}$.

It is common to estimate the effort to obtain the secret key $\mathbf{k}^*$ from the predictions with metrics like **key rank (KR)**, **success rate (SR)**, and **guessing entropy (GE)**. With Q traces in the attack stage, 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 $\mathbf{k}^*$ 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 $\mathbf{k}^*$ in $\mathbf{g}$ [102].[7] Note that since all those metrics assume the knowledge of the correct key $\mathbf{k}^*$, they are appropriate for the evaluation setting. On the other hand, the attacker should test the best guesses until reaching the one that results in the correct plaintext.

2.5 Leakage Assessment and Deep Learning

A standard option to verify that the side-channel traces contain exploitable information is to conduct a leakage assessment, e.g., **Test Vector Leakage Assessment (TVLA)** [6]. Leakage assessment 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 [67]. While leakage assessment 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 works considering deep learning and leakage assessment. Moos et al. recently made an effort to use deep learning to distinguish between two groups of data (fixed-vs-random or fixed-vs-fixed) [66]. The authors denoted this technique as **deep learning leakage assessment - DL-LA**. Cristiani et al. used **MINE (Mutual Information Neural Estimator)** to compute mutual information in high dimensional side-channel traces [26]. The obtained results showed that the MINE technique could be used as a simple tool to obtain an objective leakage evaluation. Note, however, that false positives and false negatives could also happen for deep learning-based leakage assessment. A deep neural network classifier may falsely indicate no occurrence of leakages from side-channel measurements by providing metrics similar to random guessing. This false negative may be caused by the wrong selection of a deep learning classifier that is unable to fit the leakages. At the same time, as pointed out by Moos et al., DL-LA may still leave false positive risks as the confidence is provided per trace set and not per time sample [66].

Challenge 1. *Due to limited results, it is still not clear what are (if any) the advantages of deep learning for leakage assessment.*

Recommendation 1. *It is necessary to evaluate whether state-of-the-art techniques from DL-SCA can also be applied to DL-LA. Special emphasis should be given to assessing the universality of the deep learning models, i.e., that they work for multiple DL-LA scenarios.*

3 THREAT MODELS IN PROFILING SCA

Several threat models are typically used in the deep learning-based side-channel analysis. They all assume the attacker has unlimited power and the capability to obtain an arbitrary number

[7] 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 [61].

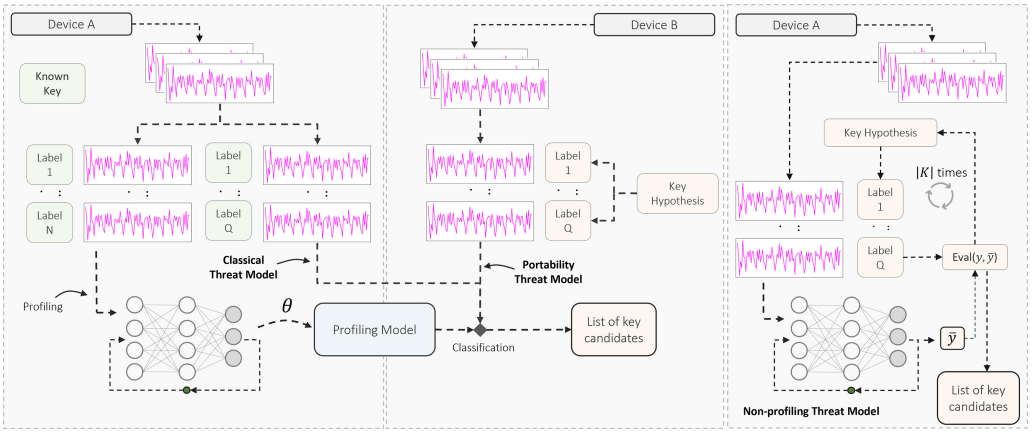

Fig. 2. Threat models in profiling SCA. The attacker uses N measurements from the profiling device to build a model. Next, in the attack stage, the adversary uses Q measurements from the device under attack to infer the secret information.

of profiling traces or evaluate any number of profiling models. Naturally, the attacker's power is always bounded, but it is unclear how to do it. One recent attempt by Picek et al. 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 [85]. By doing so, the authors aimed to "force" efficient attacks. We depict common threat models in Figure 2. Since this restriction in the attacker's power is compatible with all the other threat models, we will not consider it separately.

3.1 Classical Threat Model

In the classical 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" by Bhasin et al. [9]). While this setting is, of course, not realistic, we can notice it represents the most-explored setting in academia, see, e.g., [48, 121, 130]. 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 on different devices, and (2) assumes the best-case setting for the attacker where the measurements come from the same distribution. Indeed, the classical threat model can be used to estimate an upper bound to the attacker's power. Additionally, we can recognize two further variants of the classical threat model: (1) the secret key is the same for the profiling and attack set, and (2) the secret key is different for the profiling and attack set.

The main disadvantage of the classical threat model is that it is not realistic. The classical threat model often corresponds to a white box threat model since the attacker knows (all) details about the device under attack. Commonly, this threat model fits to the role of the attacker as a security evaluator.

3.2 Portability Threat Model

In the portability threat model, we assume there are (at least) two devices: one for profiling and the second one for the attack (Das et al. called such an attack 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 training) but only to measurements from

a similar device, with uncontrolled variations, in process, measurement setup, or other stochastic factors, to mention a few factors as identified by Bhasin et al. [9].

The portability threat model is less used in academia but is getting more attention from 2019 [27, 29, 35, 94, 112]. The main difficulty for the portability 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 (i.e., devices, which is a setup considered in the portability research). The over-specialization effect was discussed by van der Valk et al. [111]. Bhasin et al. recognized the validation as the main problem for the portability threat model since validation on the measurements from the profiling device would commonly indicate a strong attack performance [9]. At the same time, the performance on the attack device can be much worse. Consequently, Bhasin et al. 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 [132]:

- Same devices: the traces used in profiling and attacking stages 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.
- Homogeneousdevices: this case extends the dissimilarity of cross 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 the portability 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 white box/gray box to black box, depending on the device differences.

3.3 Non-profiling Supervised Threat Model

B. Timon proposed a threat model that combines the traits of non-profiling SCAs and supervised deep learning [108]. The author called the attack **differential deep learning analysis (DDLA)**. DDLA runs differential analysis (like in DPA) and then deep learning to assess how well the differential analysis performed. More precisely, the attack requires that for each key hypothesis k, the attacker computes hypothetical values and partitions 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 the non-profiling supervised 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 the attack in the non-profiling supervised threat model 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, the non-profiling supervised threat model can be considered a black box.

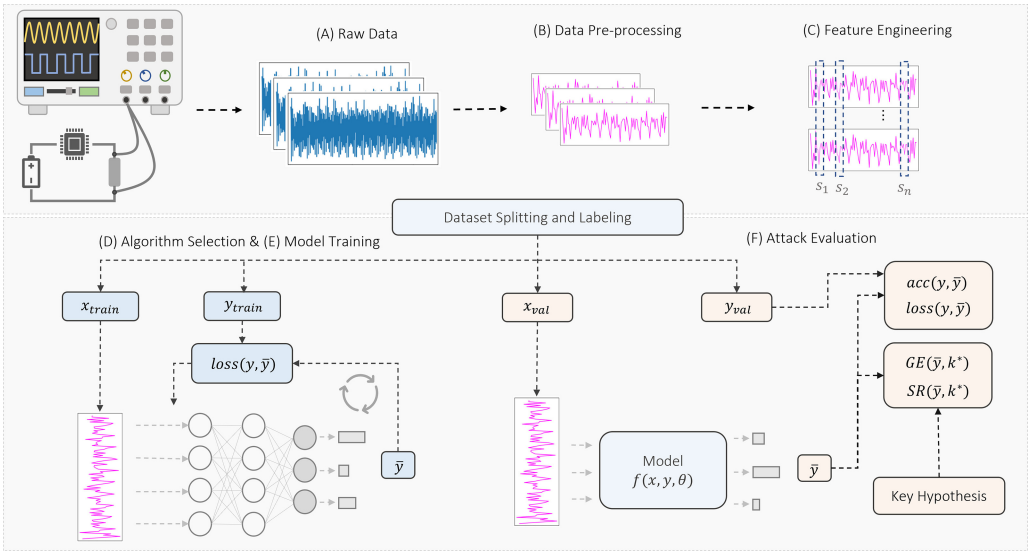

Fig. 3. Deep learning-based SCA flowchart. We do not explicitly write data standardization/normalization as this step is always done.

Achievement 1. *The first application of the non-profiling deep learning threat model in the profiling SCA [108].*

Challenge 2. *To allow deep learning to be deployed to more diverse evaluation settings, it is necessary to design a functional approach for the unsupervised deep learning-based SCA.*

Recommendation 2. *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 [43, 97]. Intuitively, the unsupervised deep learning 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.*

4 DEEP LEARNING PROCESS AND SCA

In Figure 3, we depict the common phases of the deep learning-based SCA. We note that most of the investigated works do not follow the whole flowchart but instead, concentrate on one or two phases only. For each of the phases, we discuss relevant works and challenges to be addressed. We also provide recommendations on how to approach 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) [54, 58, 125]. Finally, there are a few applications of recurrent neural networks (RNNs); more precisely, Long Short-Term Memory (LSTM) [58], residual neural networks (ResNets) [58], and generative adversarial networks (GANs) [68, 113]. We provide a brief description of those neural network types in Appendix A.

4.1 Raw Data

The first task when running side-channel analysis is to conduct data acquisition and obtain the measurements. While the data acquisition task 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 task 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 and public-key implementations, and those can be in software or hardware. 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 as few traces as possible (even a single trace). Still, commonly, one requires significantly more traces for symmetric-key implementations. On the other hand, breaking the target with a single trace is often mandatory 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 available for analysis. We present the most relevant information about publicly available datasets in Table 1.

The DPAcontest v2 dataset (DPAv2) [104] is rarely used in both machine learning and deep learning settings. We believe this is because it is not a protected implementation.

The DPAcontest v4 dataset (DPAv4) [105] represented a standard target for simpler machine learning methods, see, e.g., [55]. 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.

The DPAcontest v4.2 dataset (DPAv4.2) [106] was released to fix the problems found in DPAv4. Unfortunately, this dataset is rarely used in the deep learning-based SCA.[8]

The AES_HD dataset [98] 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 are several versions of this dataset available, where the difference is in the number of available measurements: 50 000 measurements [11], 100 000 measurements [98], and 500 000 measurements [12].

The AES_HD_MM dataset [31] is a masked hardware AES implementation 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 unaware of its availability.

The Random Delay dataset (AES_RD) [25] is a protected dataset but straightforward to break with deep learning (especially convolutional neural networks due to their spatial invariance property), so it has not been used in the last few years.

The ASCAD datasets [87] (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, they 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 hdf5

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

Table 1. Publicly Available Datasets for Side-channel Analysis Research

Dataset	Platform	Traces (Features)	Keys	Implementation	Countermeasures
DPAv2 [104] (2010)	SASEBO GII (FPGA)	100 000 (3 253)	1 fixed key	AES128	None
DPAv4 [105] (2013)	Atmega (Software)	100 000 (435 000)	1 fixed key	AES256	First-order masking (RSM)
DPAv4.2 [106] (2014)	Atmega (Software)	80 000 (1 704 400)	16 different keys	AES128	First-order masking (RSM)
AES_HD [98] (2018)	SASEBO GII (FPGA)	50 000 (1 250)	1 fixed key	AES128	None
AES_HD_MM [31] (2014)	SASEBO GII (FPGA)	5 600 000 (3 125)	1 fixed key	AES128	BM + Affine Masking
AES_RD [25] (2009)	Atmel AVR (Software)	50 000 (3 500)	1 fixed key	AES128	Hiding (Random Delay Interrupt)
ASCADf [87] (2018)	Atmega (Software)	60 000 (100 000)	1 fixed key	AES128	First-order BM (XOR)
ASCADv1 [87] (2018)	Atmega (Software)	300 000 (250 000)	Random keys + 1 fixed key	AES128	First-order BM (XOR)
ASCADv2 [2] (2021)	STM32 (Software)	810 000 (1 000 000)	Random + 1 fixed key	AES128	Second-order BM + Hiding (Shuffling)
CHES_CTF 2018 [95] (2018)	STM32 (Software)	42 000 (650 000)	Random Keys + 3 fixed keys	AES128	First-order BM (XOR)
CHES_CTF 2020 [7] (2020)	Software/Hardware	†	Fixed/random	Clyde128	ISW masking ★
Portability [10] (2020)	Atmega (Software)	50 000 (600)	4 fixed keys	AES128	None
Ed25519 (WolfSSL) [116] (2019)	STM32 (Software)	6 400 (1 000)	Random Ephemeral keys	EdDSA	None
Curve25519 (μNaCl) [20] (2020)	STM32 (Software)	5 997 (5,500)	Random Ephemeral keys	EdDSA	CSWAP, Coord./Scalar Rand.
Curve25519 [114]	STM32 (Software)	300 (8 000)	Random keys	EdDSA	CSWAP, Coord./Scalar Rand.
Curve25519 [114]	STM32 (Software)	300 (1 000)	Random keys	EdDSA	CSPOINTER, Coord./Scalar Rand.

BM denotes Boolean masking, $CSWAP$ denotes conditional swap, and $CSPOINTER$ denotes conditional pointer swap. † the number of measurements and features will depend on the acquisition setup as the authors provide source code. ★ ISW masking implemented with 3, 4, 6, and 8 shares. There are several more available datasets, but due to the target being too "simple" or lack of support (or other reasons), those datasets are not commonly used.

format. We note that recent research showed that for ASCADf, some bytes show first-order or univariate second-order leakage, which is unexpected for a protected implementation, as noted by Egger et al. [30]. The new version of the ASCAD dataset (ASCADv2) [2] appeared only recently, so there are not many results using it. In fact, to the best of our knowledge, only Masure and Strullu reported results on this dataset up to now [63]. 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.

The CHES_CTF 2018 dataset [95] is an interesting dataset that is more difficult to attack than ASCAD. Still, due to the lack of support, the publicly available version has only 10 000 traces per profiling set, making it less used recently.[9]

The CHES_CTF 2020 dataset [7] contains measurements of the masked implementations of the Clyde-128 cipher. In total, there are seven sub-challenges, four for software implementations and three for hardware implementations. Gohr et al. are the only ones to use deep learning for this dataset and only for software implementation where the target is Clyde-128 protected by an efficient variant of ISW masking [45] with 3, 4, 6, and 8 shares [34]. Each sub-challenge contains both fixed and random keys datasets.

The Portability dataset [10] is a software implementation without countermeasures. It is straightforward to attack, making it a less attractive target for deep learning. At the same time, the Portability dataset is (as far as we are aware) the only publicly available dataset to be used to investigate the portability effects for AES (see the work by Bhasin et al. [9]).

In the case of available datasets based on public-key implementations, Ed25519 (WolfSSL) [116] is unprotected (making it a non-realistic target), as discussed by Weissbart et al. [117]. Still, the lack of publicly available datasets makes it one of the rarely available datasets considering public-key implementations.

The Curve25519 μNaCl dataset [20] is a protected implementation, making it a (somewhat) realistic target, but there are only a few available results. We believe such a lack of results is because most SCA community investigates deep learning attacks on block ciphers (AES).

The Curve25519 datasets [114] contain protected implementations of EdDSA, differing in the number of features and countermeasures. The datasets appeared very recently, so there is only one paper investigating them [115].

We 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 [125].

Achievement 2. *Release of the ASCAD database that became a standard in the profiling SCA evaluation and proposing to use the hdf5 format for the SCA domain [8].*

Recommendation 3. *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 recommend using such datasets sparsely unless the characteristics of the specific dataset are significantly different from those of publicly available datasets. We hope that the list in Table 1 will help researchers understand 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 baseline makes it possible to conduct a meaningful comparison among different deep learning techniques.*

[9] An extended version of the CHES_CTF 2018 dataset, with some pre-processing, is available at http://aisylabdatasets.ewi.tudelft.nl.

Recommendation 4. *Publicly available datasets are available from different sources. As hosting them takes some 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 hdf5 format and data structure as in the ASCAD datasets, which are now widely accepted by the DL-SCA community. While other formats are available (e.g., TFRecord or Petastorm), there does not seem to be some specific need in the context of DL-SCA that hdf5 cannot do.*

Challenge 3. *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). Commonly, deep learning attacks easily break such targets and require significantly less than available traces. Consequently, the gap between academia and industry that uses more realistic targets widens. The DL-SCA community must actively work to reduce this gap. Additionally, at the moment, there are insufficient results with more difficult datasets like ASCADv2 and CHES_CTF 2020 to assess properly how challenging they will be for DL-SCA. Therefore, the DL-SCA community needs to investigate more such challenging datasets.*

Recommendation 5. *Based on the list of publicly available datasets, it is clear that we need to consider protected hardware implementations and corresponding datasets. 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. On the other hand, protected hardware implementations like the CHES_CTF 2020 dataset use a less known cipher, making it rarely investigated with DL-SCA. Since hardware implementations have better performance (than those in software), it is necessary to have 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., the work by Swaminathan et al. [103], requiring appropriate datasets to evaluate the attacks.*

4.2 Data Pre-processing

As stated in Section 1, recall 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 [48, 121, 130]. Using data standardization/normalization is a common practice when working with neural networks [36], so it is not surprising to see it in the SCA domain, but we can already observe that the claim on no pre-processing is (strictly speaking) 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, as pointed out by Cagli et al. [17]. Interestingly, deep learning techniques (especially convolutional neural networks) work well even in the presence of misalignment (due to the spatial invariance property) [17, 46]. Thus, the alignment part of pre-processing does not seem to be required. Nevertheless, Zhou and Standaert found the alignment step to be helpful even when using deep learning techniques [135]. In this sense, the model selection phase tends to be simplified if alignment is feasible.

In many SCA settings, the training set might 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 [17]. The reason is that when dealing with more complex countermeasures, one may need to use more complex neural network architectures. Then, data augmentation can be a simple option to prevent overfitting.

Achievement 3. *Breaking protected datasets and showing that CNNs are naturally adept at defeating the desynchronization countermeasure [17].*

Similar conclusions were also made by Pu et al., who showed how data augmentation with shifts improves the attack performance [88]. Perin et al. utilized data shift data augmentation for side-channel analysis on ECC to achieve up to 100% accuracy [75]. The latter 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, Zhimin et al. proposed to use a mixup data augmentation where several existing traces are combined (mixed up) to produce a new trace [57]. 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 [48]. Due to the binomial data distribution, the datasets are highly imbalanced for certain leakage models like the Hamming weight or Hamming distance. As such, the imbalanced data 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 [83]. 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 [120]. More recently, Wang et al. proposed to use **generative adversarial networks (GANs)** for data augmentation [113]. While the results look interesting, the main problems seem to be relatively poor performance when dealing with many labels and the need for a large dataset to train GANs, making the whole procedure less useful in practice. Mukhtar et al. further explored the direction of data augmentation with GANs and managed to generate realistic leakage traces for both symmetric and public-key cryptographic implementations [68]. To this end, the authors used conditional generative adversarial networks and Siamese networks.

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 [125]. The results showed not only that deep learning attacks can improve the performance after the autoencoder 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 transformations (e.g., phase-only correlation, principal component analysis–PCA, or alignment methods) to input data in each branch, extract the relevant features, and allow a better generalization of the profiling model [119].

Challenge 4. *Data pre-processing can be useful and improve deep learning SCA performance. PCA and* ***linear discriminant analysis (LDA)*** *tends to improve the attack performance in the portability setup, especially in the presence of noisy measurements, as demonstrated by Danial et al. [27]. Still, we are missing a clear set of guidelines on what pre-processing techniques to use for different settings.*

Recommendation 6. *We recommend always using data standardization/normalization as it is common practice when dealing with neural networks. Since it is important to recognize if other pre-processing techniques should be used, we recommend conducting systematic studies on the relevance of pre-processing for DL-SCA.*

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 7. *We recommend using data augmentation when the number of available profiling measurements is not sufficient. As a rule of thumb, a sufficient number of profiling measurements is any value that allows breaking the target (e.g., reaching a guessing entropy of 1). We recommend a systematic comparison among different data augmentation techniques, with a special emphasis on the differences in the performance between standard machine learning techniques and custom ones developed for DL-SCA.*

4.3 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 (besides the input dimension), the key aspect of a successful attack is to use the most informative features. Commonly, to this end, one either uses a feature selection technique (e.g., Pearson correlation or SNR) or dimensionality reduction (e.g., PCA). Still, there are 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 the recent work of Picek et al. showed that many feature selection techniques behave similarly [82], 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 an 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 [91]. The same authors used the autoencoder-based approach for feature engineering, where it also showed good results on other ciphers like ASCON [90]. 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 [69]. Wu et al. followed a different direction and proposed a triplet model to find highly efficient embeddings of input data [123]. The authors combined deep learning-based feature engineering with a classical profiling attack (template attack) and showed that such a combination could reach the performance of state-of-the-art deep learning-based attacks.

Lu et al. showed significantly improved SCA performance when using raw data (thus, no pre-selected windows) in their recent work [56]. With that result, the authors challenged the common practices in the deep learning-based SCA and demonstrated that using more features can bring advantages. While the authors showed potential from the attack perspective (requiring fewer traces to break the targets), the attack also required large neural networks, making the hyperparameter tuning more complex. The common approach is challenged even further by Perin et al. [79]. The authors considered the ASCAD datasets (fixed and random keys) and managed to break them with

Table 2. A Summary of Feature Engineering Methods for Machine Learning-based SCA

Classical methods	Pearson correlation [92], SNR [60], PCA [118], LDA [101], SOST/SOSD [33]
Deep learning-based methods	Autoencoder [90, 91], Similarity learning [123]
Other methods	Use raw features directly [56], re-sampling [79]

very small neural networks (up to two hidden layers), requiring only a single attack trace. The authors investigated three feature selection scenarios and evaluated the attack performance for each. A summary of the feature engineering methods used in SCA is presented in Table 2.

While the DL-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 simplification, in the end, reduces the benefits of automation steps in SCA, where an appropriate model would automatically find relevant features inside raw traces. Moreover, selecting hyperparameters of successful models becomes easier, which might slow down the advances in the hyperparameter tuning process.

Challenge 6. *Feature engineering is either 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. Preliminary results indicate that more features help, but it remains unclear how to decide on the number of features to use or whether larger traces also require larger neural networks.*

Recommendation 8. *We recommend investigating the performance of DL-SCA with significantly more features than currently used to assess the limitations of implicit feature selection as done by neural networks. Additionally, it is necessary to assess whether working with longer traces will require working with larger neural network architectures.*

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

Recommendation 9. *We recommend using raw data in the deep learning-based SCA. Additionally, it would be beneficial to provide a systematic comparison among different feature engineering techniques, both based on classical methods and deep learning techniques as reported in Table 2.*

It is important to understand that raw data or raw side-channel measurements are often subject to inherent feature engineering as part of side-channel acquisitions. One example is the trigger signal that is implemented either as part of the source code running in the target device or by adjusting the oscilloscope to detect a specific pattern. Later, side-channel measurements are usually trimmed in order to contain (mainly) the leakages from the target execution operation (e.g., the AES encryption interval). The amount of noisy and irrelevant measurement samples that the acquired traces may contain depends on the availability and quality of a trigger acquisition signal. Therefore, the meaning of raw data for deep learning-based SCA may be side-channel measurements that mostly include the leakages from the target operation (e.g., the full AES encryption) without necessarily passing through alignment and trimming processes to end up with a more optimized trace interval (e.g., the specific location of an `S-box` operation for a single target key byte).

4.4 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 [36]. 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 [58].

Achievement 4. *The first application of convolutional neural networks for profiling SCA and demonstrating the potential of deep learning [58].*

In some early works, e.g., [17, 58], the authors did not provide details about the reasoning for CNN hyperparameters selection.[10] Other works, e.g., [8, 48] selected CNN models based on the VGG structure, commonly applied to image and audio classification domains [39, 99]. The results indicated that model selection based on a VGG-like structure could be efficient across several domains and datasets. More precisely, such architectures were successfully applied to, e.g., software implementations with Boolean masking (ASCAD), software implementations with Boolean masking and desynchronization (ASCAD), software implementation with random delay interrupts (AES_RD), and unprotected hardware implementation (AES_HD).

Achievement 5. *Showing that VGG-like architectures perform very well in the SCA context and that one architecture can break different (software/hardware, protected/unprotected, masking/hiding) implementations [8, 48].*

Datasets evaluated in these first deep learning-based SCA publications [8, 17, 48, 58] 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 that knowledge gap, Zaid et al. [130] proposed a research direction focused on optimizing CNN models that are dataset-specific. One of the main goals of Zaid et al. was to demonstrate that CNN models could break previously evaluated public datasets with very small architectures and show remarkable key recovery performance [130]. The authors provided the first attempt to build a methodology for constructing CNNs for SCA. While it is not easy to follow the proposed methodology with different datasets, the authors are the first to clearly write about the need to have such methodologies in DL-SCA.

Achievement 6. *Recognizing (and clearly stating) the need for methodologies to design small and well-performing neural networks for SCA [130].*

Later, Wouters et al. improved the previous architectures from Zaid et al. [130] with a pooling-based dimensionality reduction layer as the input layer [121]. There, the required number of attack traces to recover the key was reduced (slightly) further with the improved CNN models containing less than half of the trainable parameters compared to the original architectures proposed Zaid et al. [130]. Those works showed that selecting small CNN architectures based on a set of guidelines (methodologies) results in an excellent attack performance. Furthermore, since the selected models are small (small number of trainable parameters), there was no need to use explicit regularization, which is a common option to prevent overfitting and improve generalization.

Following the same concept of finding small and efficient neural network models for profiling SCA, Rijsdijk et al. proposed the application of reinforcement learning to select neural network model hyperparameters [93]. Their search system is rewarded based on attack performance (the

[10]For instance, Maghrebi et al. [58] mentioned they used genetic algorithms for hyperparameter tuning, but provided no details.

guessing entropy of the correct key), and the decision process is based on defining the smallest possible CNN models. This work improved upon the previous best results and showed that the considered datasets could be broken with even fewer traces.

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, complex datasets would be a situation where one solution is to explore larger architectures [38]. Along these lines, several works started to adopt methods for hyperparameter search on the same previously analyzed datasets (e.g., ASCAD, DPAv4, CHES_CTF 2018). A random search was adopted by Perin et al. to define a set of CNN and MLP models to be later combined into ensemble models [76]. There, 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 using a Bayesian optimization search to define efficient MLP and CNN models [122]. Bayesian optimization is beneficial for large search space scenarios and when 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 can give better results than 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. Such 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, 135]. 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 show 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, it would be useful to develop efficient guidelines to determine the most important 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.*

Recommendation 10. *Evaluate the common hyperparameter ranges and automated tuning techniques with new, more challenging datasets. Furthermore, report the performance with the best architectures and the percentage of the well-performing models (i.e., those that break the target).*

In the context of public-key implementations, Carbone et al. were the first to consider deep learning-based profiling SCA [18]. 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 EdDSA in WolfSSL and showed it is possible to break the target with a single attack trace [117]. Interestingly, the authors used the same CNN as Kim et al. used when attacking AES [48]. Finally, Perin et al. used a fundamentally different approach to attacking a protected ECDSA (Elliptic Curve Digital Signature Algorithm) implementation [75]. After running a horizontal attack[11] (for details about the horizontal attack, see the

[11]Horizontal attacks treat every key bit as represented by a small trace interval and, therefore, split every side-channel measurement into a set of sub-traces.

Table 3. A Summary of Masking Countermeasures Successfully Attacked with DL-SCA

Countermeasure	Platform
First-order Boolean masking	SW [8, 58], HW [119]
First-order Boolean masking with desync/RDI/jitter/S-box shuffling	SW [130]
ISW masking	SW [34]
First-order RSM	SW [48]
Affine masking (with known affine share m)	SW [63]

work by Nascimento et al.) [70], the authors used deep learning to iteratively correct errors stemming from the horizontal attack. The authors managed to get 100% accuracy with their approach, even if starting with only 52% accuracy. Despite attacking protected implementations and the need to succeed in a single attack trace, the results indicate that goal to be relatively easily achievable, 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 as we need to distinguish between bit values 0 and 1 only).

Challenge 9. *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 (details about broken masking countermeasures are in Table 3). The defined leakage models are constructed based on an intermediate value s, ignoring the fact that the 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: $s_{r,m} = (s \times m) \oplus r$. For the ASCADv2 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. Moreover, no publication demonstrated how to bypass high-order masking schemes in cryptographic algorithms with DL-SCA. Finally, it is unclear what are the capabilities of various neural network architectures against different SCA countermeasures.

Recommendation 11. *To allow more meaningful comparisons, we must define a concept of an optimized architecture. For instance, a model optimized to recover a key with as few as possible attack traces, and at the same time, that can generalize in different conditions such as portability. Additionally, we recommend studying the limitations of DL-SCA against stronger countermeasures, e.g., assessing the difference when attacking Boolean and affine masking.*

4.5 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 the training process. 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 by Perin and Picek [77]. Indeed, the 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 training). More importantly, the authors observed that certain optimizers perform better, and this behavior is common for any model hyperparameters when the number of trainable parameters is restricted to some bounds.

Training performance and duration are 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. Zhang et al. proposed a Cross-Entropy Ratio loss function to remove negative effects from imbalanced labels in the Hamming weight leakage model [133]. Ranking Loss was proposed by Zaid et al. as a method to minimize the ranking error of the correct key concerning other key hypotheses [129]. Kerkhof et al. proposed a loss function called Focal Loss Ratio that provides very good attack performance but does not have high computational overhead [47]. Later, Zaid et al. proposed Ensembling Loss as a custom loss function to maximize diversity in an ensemble-based approach [131].

Challenge 10. *Defining a loss function that provides general behavior for multiple datasets and attack models is hard. It still remains unclear whether we require custom loss functions (or any other custom neural network element) for DL-SCA.*

Recommendation 12. *Using categorical cross-entropy represents a safe choice (at the moment), 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). One effort to make the selected neural network models even smaller was made by Perin et al. [80]. The authors used pruning to design small 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.

Challenge 11. *As one of the dominant problems in DL-SCA is overfitting, it is necessary to investigate how to prevent or, at least reduce it.*

Recommendation 13. *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. Then, the learning rate is adapted (manually or adaptively) during training. It is necessary to conduct a systematic evaluation of various regularizers and assess which ones perform the best in the context of DL-SCA.*

4.6 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 first investigated by Picek et al. for the case of AES [83]. The authors demonstrated that an imbalanced class problem leads to inconsistency between, e.g., accuracy and the SCA performance. This inconsistency is more likely to be observed for side-channel measurements collected from protected or highly noisy implementations.

Achievement 7. *Investigating the differences between the machine learning and side-channel metrics and possible issues stemming from those differences [83].*

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 [76]. 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, these results confirm 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. Robissout et al. proposed the evaluation of the success rate of (small portions of) training and validation sets to determine the best training epoch [96]. Still, it is unclear how to use the proposed approach when there are random keys in the profiling set. Perin et al. evaluated a mutual information-based metric to identify the best epoch to stop training [74].

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. By doing so, we ensure that GE is a measure of model generalization. Such version of GE computation is explored by Wu et al. and defined as generalized guessing entropy [126]. Furthermore, Wu et al. proposed to compute GE using the geometric mean over multiple key ranks instead of the arithmetic mean (as commonly done) case as it seems to lead to more stable results [124]. Zhang et al. proposed a guessing entropy estimation algorithm based on theoretical distributions of the ranking score vectors [134]. That 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 a context since the GE evaluation is expensive and needs to be done multiple times. 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 [126].

Challenge 12. *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.*

Recommendation 14. *It would be beneficial to develop a metric that measures the level of desynchronization (or even noise) that the trained model can process. Such a metric would also require some level of AI explainability in SCA.*

Challenge 13. *It is unlikely but still relevant to try 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.*

Recommendation 15. *We should measure and understand how the selected hyperparameters make the model succeed (or fail) in fitting existing leakages. Additionally, it would be relevant to develop techniques allowing us to quantify the differences in the targets to assess how "universal" a certain model is.*

Challenge 14. *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. Finally, due to the random weight re-initialization, retraining the same dataset and architecture from scratch leads to a different result.*

Recommendation 16. *We recommend using guessing entropy as the metric of choice to evaluate deep learning-based SCA. Furthermore, we recommend always sampling the attack traces from a large pool of available traces and averaging the estimation by using the median. Finally, we recommend reporting the results for a number of independent model trainings.*

5 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 unsuccessful. 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, the information on how to design stronger countermeasures 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 work in the direction of SCA and AI explainability was made by Hettwer et al. and it aimed at interpreting neural network decisions by using heatmapping techniques [41]. 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 [62]. Van der Valk 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 [110].

Van der Valk et al. made the first step toward the 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 [111]. 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. used ablation to explain how neural networks process hiding countermeasures [127]. The authors concluded that simpler countermeasures are getting processed in the shallow layers, while more complex countermeasures are processed in deeper layers.

Challenge 15. *Understand how neural networks process masking and circuit-level countermeasures.*

Examples of circuit-level countermeasures include, to name a few, **Wave Dynamic Differential Logic (WDDL)** [109], **Masked DPL (MDPL)** [86], **Dynamic and Differential Swing-Limited Logic (DDSLL)** [71]), **Low-Dropout (LDO)** regulators [100], **Current Domain Signature Attenuation (CDSA)** [28, 32], hardware accelerators [53], and randomization-based SCA protection [13].

Recommendation 17. *We believe it is important to concentrate on both the techniques that provide insights into what features are the most important and techniques that provide insights into what is happening in the neural network. We consider the latter direction to be more challenging but potentially more impactful.*

Challenge 16. *Propose efficient countermeasures tuned to fight against deep learning-based SCA.*

Recommendation 18. *It would be interesting to consider the developments in the security of machine learning and recognize if some failure modes (e.g., poisoning attacks or adversarial examples could be used as a countermeasure against DL-SCA. This possibility should evaluate not only the (hopefully) decreased attack performance but also how realistic it is to deploy such a countermeasure in hardware.*

6 FRAMEWORKS

The current state of the art for deep learning-based SCA mostly uses the Keras framework [22]. Still, it is not uncommon to also encounter TensorFlow [1] and PyTorch [72] implementations. Some authors also use Matlab [64] and scikit-learn [73], especially in the context of machine learning.

Recommendation 19. *Ideally, the DL-SCA community should use the same framework to make the results easily reproducible. As this is difficult to expect, we recommend running experiments multiple times to reduce the influence of randomness stemming from the choice of different frameworks.*

To the best of our knowledge, there are not many publicly available customized 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 DLSCA [14].[12] The tool is meant to be modular, but the current version supports only basic functionalities. It seems there has been no active development in the last two years.

The second publicly available framework is called the AISY framework. This framework was recently developed by Perin et al. [78], and it contains a significant number of functionalities developed in the last few years in the deep learning-based SCA domain.[13] The main advantage of the AISY framework seems to be the relative ease of conducting reproducible research.

Recommendation 20. *The SCA community would benefit from a publicly available tool that is regularly updated and contains 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 those functionalities would still represent a significant effort for the authors of the framework, we propose establishing a standard coding style so it is easy to import functionalities from various research works.*

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

7 THE FUTURE OF DEEP LEARNING-BASED SCA

The recent work of F. Chollet defines measures for intelligence and generalization in artificial intelligence [21]. The author provided 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 suggested 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, the existing state of the art does not necessarily mean that deep learning-based SCA cannot go beyond this point. The state-of-the-art DL-SCA results are limited to the concept of local generalization. According to F. Chollet, "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" [21]. 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 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 the 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, 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 stage. This way, the model could adapt to the new unknown leakage models. One option to (at least) approach broader generalization could be to use transfer learning as investigated by Thapar et al. [107].

While local generalization aspects limit research in AI-based side-channel analysis, every small improvement on 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 17. *Design countermeasures that achieve broad generalization to cover unknown attack scenarios.*

Recommendation 21. *Unfortunately, it seems we are rather far from having countermeasures that achieve broad generalization. For this to happen, we believe we first need substantial progress in understanding how to mount powerful deep learning-based attacks but also understanding how those attacks work.*

Challenge 18. *There are many developments for deep learning-based SCA. Consequently, it becomes difficult to recognize what technique to use and when. This difficulty becomes especially pronounced as there are no comparisons among different techniques (e.g., if ensembles are more beneficial than data augmentation or custom loss functions).*

Recommendation 22. *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.*

Recommendation 23. *As a general recommendation, we believe there should be an emphasis on reproducible research. Every phase in DL-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). As a preferred practice, researchers should evaluate the attack performance with a different number of profiling and attack traces. Finally, it should become a standard to publish the trained models.*

8 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 techniques are not well justified and critically evaluated and thus remain neglected. Consequently, it is not easy 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. This work might help with that goal.

This paper discusses the most important developments in DL-SCA and signalizes many open challenges. Each challenge is accompanied by a recommendation on how to address it. Besides those recommendations, we recognize the need to facilitate the results' reproducibility: making publicly available datasets, writing all the required details to reproduce the experiments, and publishing the neural network models used in the experiments. In short, we hope our work will help other researchers and practitioners to pave new research avenues and move forward in the DL-SCA domain.

APPENDIX

A COMMON NEURAL NETWORK TYPES IN SCA

We briefly describe several types of neural networks used in 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.

Multilayer Perceptron. The multilayer perceptron (MLP) 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 [36].

Convolutional Neural Networks. Convolutional neural networks (CNNs) commonly consist of three types of layers: convolutional layers, pooling layers, and fully connected layers [36]. 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.

Autoencoders. Autoencoders (AEs) 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 [54, 125]. The second type is autoencoder models for feature extraction, reducing the dimensionality of the input data [119]. Finally, autoencoders were also used for classification [58], where the authors placed a *softmax* layer as the output layer.

Recurring Neural Networks. **Recurrent Neural Networks (RNNs)** are designed to recognize patterns in sequences of data [36]. RNNs perform the same function on every input data and store 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)** [128], and LSTM is a well-known example of RNN-based architecture [44], also used in SCA ([58]).

Residual Neural Networks. **Residual Neural Networks (ResNet)** consist of residual blocks with several layers and a shortcut connection [38]. 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 becoming so deep that they cannot propagate the information anymore.

Generative Adversarial Networks. Generative adversarial networks (GANs) are specific neural network-based structures that can generate artificial data based on an evaluated data distribution [37]. 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.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Agence nationale de la sécurité des systèmes d'information (ANSSI). 2021. ASCADv2. Github repository. https://github.com/ANSSI-FR/ASCAD.

[3] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security evaluation of home-based IoT deployments. In *2019 IEEE Symposium on Security and Privacy, SP*. IEEE, 1362–1380.

[4] Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. 2021. SoK: Computer-aided cryptography. In *42nd IEEE Symposium on Security and Privacy, SP 2021*. IEEE, 777–795. https://prosecco.gforge.inria.fr/personal/bblanche/publications/BarbosaetalOakland21.pdf.

[5] Lejla Batina, Milena Djukanovic, Annelie Heuser, and Stjepan Picek. 2021. It started with templates: The future of profiling in side-channel analysis. In *Security of Ubiquitous Computing Systems: Selected Topics*, Gildas Avoine and

Julio Hernandez-Castro (Eds.). Springer International Publishing, Cham, 133–145. https://doi.org/10.1007/978-3-030-10591-4_8

[6] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab. 2013. Test Vector Leakage Assessment (TVLA) Methodology in Practice.

[7] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Charles Momin, François-Xavier Standaert, and Balazs Udvarhelyi. 2021. Spook SCA CTF. https://doi.org/10.14428/DVN/W2SV5G

[8] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 2020. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptographic Engineering* 10, 2 (2020), 163–188. https://doi.org/10.1007/s13389-019-00220-8

[9] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. 2020. 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. https://www.ndss-symposium.org/ndss-paper/mind-the-portability-a-warriors-guide-through-realistic-profiled-side-channel-analysis/.

[10] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. 2020. Portability Dataset. Website. http://aisylabdatasets.ewi.tudelft.nl/.

[11] Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2020. AES HD Dataset - 50 000 Traces. Github repository. https://github.com/AISyLab/AES_HD.

[12] Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2020. AES HD Dataset - 500 000 Traces. Github repository. https://github.com/AISyLab/AES_HD_2.

[13] Rinat Breuer, François-Xavier Standaert, and Itamar Levi. 2022. Fully-digital randomization based side-channel security - toward ultra-low cost-per-security. *IEEE Access* 10 (2022), 68440–68449. https://doi.org/10.1109/ACCESS.2022.3185995

[14] Martin Brisfors and Sebastian Forsmark. 2019. DLSCA: A Tool for Deep Learning Side Channel Analysis. Cryptology ePrint Archive, Report 2019/1071. https://ia.cr/2019/1071.

[15] Olivier Bronchain and François-Xavier Standaert. 2020. Side-channel countermeasures' dissection and the limits of closed source security evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 2 (Mar. 2020), 1–25. https://doi.org/10.13154/tches.v2020.i2.1-25

[16] Ileana Buhan, Lejla Batina, Yuval Yarom, and Patrick Schaumont. 2021. SoK: Design tools for side-channel-aware implementions. *CoRR* abs/2104.08593. (2021). arXiv:2104.08593 https://arxiv.org/abs/2104.08593.

[17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. 2017. 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 (Lecture Notes in Computer Science, Vol. 10529)*, Wieland Fischer and Naofumi Homma (Eds.). Springer, 45–68.

[18] Mathieu Carbone, Vincent Conin, Marie-Angela Cornélie, François Dassance, Guillaume Dufresne, Cécile Dumas, Emmanuel Prouff, and Alexandre Venelli. 2019. Deep learning to evaluate secure RSA implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 2 (Feb. 2019), 132–161. https://doi.org/10.13154/tches.v2019.i2.132-161

[19] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2002. Template attacks. In *CHES (LNCS, Vol. 2523)*. Springer, 13–28. San Francisco Bay (Redwood City), CA, USA.

[20] Łukasz Chmielewski. 2020. REASSURE (H2020 731591) ECC Dataset. https://doi.org/10.5281/zenodo.3609789

[21] François Chollet. 2019. On the measure of intelligence. *CoRR* abs/1911.01547 (2019). arXiv:1911.01547. http://arxiv.org/abs/1911.01547.

[22] Francois Chollet et al. 2015. Keras. https://github.com/fchollet/keras. Github repository.

[23] Omar Choudary and Markus G. Kuhn. 2014. Efficient template attacks. In *Smart Card Research and Advanced Applications*, Aurélien Francillon and Pankaj Rohatgi (Eds.). Springer International Publishing, Cham, 253–270.

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

[25] Jean-Sébastien Coron and Ilya Kizhvatov. 2009. Trace Sets with Random Delays – AES_RD. Github repository. https://github.com/ikizhvatov/randomdelays-traces.

[26] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. 2020. 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 (Lecture Notes in Computer Science, Vol. 12418)*, Jianying Zhou, Mauro Conti, Chuadhry Mujeeb Ahmed, Man Ho Au, Lejla Batina, Zhou Li, Jingqiang Lin, Eleonora Losiouk, Bo Luo, Suryadipta Majumdar, Weizhi Meng, Martín Ochoa, Stjepan Picek, Georgios Portokalidis, Cong Wang, and Kehuan Zhang (Eds.). Springer, 144–162.

[27] Josef Danial, Debayan Das, Anupam Golder, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. 2020. EM-X-DL: Efficient cross-device deep learning side-channel attack with noisy EM signatures. *CoRR* abs/2011.06139 (2020). arXiv:2011.06139. https://arxiv.org/abs/2011.06139.

[28] Debayan Das, Josef Danial, Anupam Golder, Nirmoy Modak, Shovan Maity, Baibhab Chatterjee, Dong-Hyun Seo, Muya Chang, Avinash Varna, Harish Krishnamurthy, Sanu Mathew, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. 2020. 27.3 EM and power SCA-Resilient AES-256 in 65nm CMOS through >350× current-domain signature attenuation. In *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16–20, 2020*. IEEE, 424–426. https://doi.org/10.1109/ISSCC19947.2020.9062997

[29] Debayan Das, Anupam Golder, Josef Danial, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. 2019. X-DeepSCA: Cross-device deep learning side channel attack. In *Proceedings of the 56th Annual Design Automation Conference 2019* (Las Vegas, NV, USA) *(DAC'19)*. Association for Computing Machinery, New York, NY, USA, Article 134, 6 pages. https://doi.org/10.1145/3316781.3317934

[30] Maximilian Egger, Thomas Schamberger, Lars Tebelmann, Florian Lippert, and Georg Sigl. 2022. A second look at the ASCAD databases. In *Constructive Side-Channel Analysis and Secure Design*, Josep Balasch and Colin O'Flynn (Eds.). Springer International Publishing, Cham, 75–99.

[31] Yunsi Fei. 2014. Northeastern University TeSCASE Dataset – AES_HD_MM. Website. https://chest.coe.neu.edu/?current_page=POWER_TRACE_LINK&software=ptmasked.

[32] Archisman Ghosh, Debayan Das, Josef Danial, Vivek De, Santosh Ghosh, and Shreyas Sen. 2021. 36.2 An EM/Power SCA-Resilient AES-256 with synthesizable signature attenuation using digital-friendly current source and RO-bleed-based integrated local feedback and global switched-mode control. In *IEEE International Solid-State Circuits Conference, ISSCC 2021, San Francisco, CA, USA, February 13–22, 2021*. IEEE, 499–501. https://doi.org/10.1109/ISSCC42613.2021.9365978

[33] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. 2006. Templates vs. stochastic methods. In *Cryptographic Hardware and Embedded Systems - CHES 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4249)*, Louis Goubin and Mitsuru Matsui (Eds.). Springer, 15–29.

[34] Aron Gohr, Friederike Laus, and Werner Schindler. 2022. Breaking masked implementations of the Clyde-cipher by means of side-channel analysis - a report on the CHES challenge side-channel contest 2020. *IACR Cryptol. ePrint Arch.* (2022), 471.

[35] Anupam Golder, Debayan Das, Josef Danial, Santosh Ghosh, Shreyas Sen, and Arijit Raychowdhury. 2019. Practical approaches toward deep-learning-based cross-device power side-channel attack. *IEEE Trans. Very Large Scale Integr. Syst.* 27, 12 (2019), 2720–2733. https://doi.org/10.1109/TVLSI.2019.2926324

[36] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.

[37] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. 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*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.). 2672–2680. https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. 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, 770–778. https://doi.org/10.1109/CVPR.2016.90

[39] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson. 2017. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017*. IEEE, 131–135.

[40] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. 2019. Applications of machine learning techniques in side-channel attacks: A survey. *Journal of Cryptographic Engineering* 10, 2 (April 2019), 135–162. https://doi.org/10.1007/s13389-019-00212-8

[41] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. 2020. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In *Selected Areas in Cryptography – SAC 2019*, Kenneth G. Paterson and Douglas Stebila (Eds.). Springer International Publishing, Cham, 645–666.

[42] Annelie Heuser and Michael Zohner. 2012. Intelligent machine homicide. In *Constructive Side-Channel Analysis and Secure Design*, Werner Schindler and Sorin A. Huss (Eds.). Springer Berlin, Berlin, 249–264.

[43] Johann Heyszl, Andreas Ibing, Stefan Mangard, Fabrizio De Santis, and Georg Sigl. 2013. Clustering Algorithms for Non-Profiled Single-Execution Attacks on Exponentiations.

[44] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[45] Yuval Ishai, Amit Sahai, and David Wagner. 2003. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003*, Dan Boneh (Ed.). Springer Berlin, Berlin, 463–481.

[46] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. 2015. 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. https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

[47] Maikel Kerkhof, Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. Focus is key to success: A focal loss function for deep learning-based side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design*, Josep Balasch and Colin O'Flynn (Eds.). Springer International Publishing, Cham, 29–48.

[48] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. 2019. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 3 (2019), 148–179.

[49] Knud Lasse Lueth. 2020. 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/. Accessed August 4, 2021.

[50] Paul C. Kocher. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proceedings of CRYPTO'96 (LNCS, Vol. 1109)*. Springer-Verlag, 104–113.

[51] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Advances in Cryptology - CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara*, California, USA, August 15–19, 1999, *(Proceedings Lecture Notes in Computer Science, Vol. 1666)*, Michael J. Wiener (Ed.). Springer, 388–397. https://doi.org/10.1007/3-540-48405-1_25

[52] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'99)*. Springer-Verlag, London, UK, 388–397. http://dl.acm.org/citation.cfm?id=646764.703989.

[53] Raghavan Kumar, Vikram B. Suresh, Mark A. Anders, Steven K. Hsu, Amit Agarwal, Vivek K. De, and Sanu K. Mathew. 2022. An 8.3-to-18Gbps reconfigurable SCA-resistant/dual-core/blind-bulk AES engine in intel 4 CMOS. In *IEEE International Solid-State Circuits Conference, ISSCC 2022*, San Francisco, CA, USA, February 20–26, 2022. IEEE, 1–3. https://doi.org/10.1109/ISSCC42614.2022.9731739

[54] Donggeun Kwon, HeeSeok Kim, and Seokhie Hong. 2020. Improving non-profiled side-channel attacks using autoencoder based preprocessing. *IACR Cryptol. ePrint Arch.* 2020 (2020), 396. https://eprint.iacr.org/2020/396.

[55] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. 2014. A machine learning approach against a masked AES. *Journal of Cryptographic Engineering* 5, 2 (Nov. 2014), 123–139. https://doi.org/10.1007/s13389-014-0089-3

[56] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. 2021. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 3 (Jul. 2021), 235–274. https://doi.org/10.46586/tches.v2021.i3.235-274

[57] Zhimin Luo, Mengce Zheng, Ping Wang, Minhui Jin, Jiajia Zhang, Honggang Hu, and Nenghai Yu. 2021. Towards strengthening deep learning-based side channel attacks with mixup. *CoRR* abs/2103.05833 (2021). arXiv:2103.05833. https://arxiv.org/abs/2103.05833.

[58] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. 2016. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Proceedings (Lecture Notes in Computer Science, Vol. 10076)*, Claude Carlet, M. Anwar Hasan, and Vishal Saraswat (Eds.). Springer, 3–26.

[59] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2006. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer. 338 pages. ISBN 0-387-30857-1, http://www.dpabook.org/.

[60] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Vol. 31. Springer Science & Business Media.

[61] James L. Massey. 1994. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*. IEEE, 204.

[62] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. 2019. 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 (Lecture Notes in Computer Science, Vol. 11421)*, Ilia Polian and Marc Stöttinger (Eds.). Springer, 145–167. https://doi.org/10.1007/978-3-030-16350-1_9

[63] Loïc Masure and Rémi Strullu. 2021. Side Channel Analysis against the ANSSI's protected AES Implementation on ARM. Cryptology ePrint Archive, Report 2021/592. https://ia.cr/2021/592.

[64] MATLAB. 2010. *Version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts.

[65] John V. Monaco. 2018. SoK: Keylogging side channels. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings*. IEEE Computer Society, 211–228.

[66] Thorben Moos, Felix Wegener, and Amir Moradi. 2021. DL-LA: Deep learning leakage assessment: A modern roadmap for SCA evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 3 (Jul. 2021), 552–598. https://doi.org/10.46586/tches.v2021.i3.552-598

[67] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. 2018. Leakage detection with the x2-Test. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 1 (Feb. 2018), 209–237. https://doi.org/10.13154/tches.v2018.i1.209-237

[68] Naila Mukhtar, Lejla Batina, Stjepan Picek, and Yinan Kong. 2022. Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices. In *Topics in Cryptology – CT-RSA 2022*, Steven D. Galbraith (Ed.). Springer International Publishing, Cham, 297–321.

[69] Naila Mukhtar, Apostolos P. Fournaris, Tariq M. Khan, Charis Dimopoulos, and Yinan Kong. 2020. Improved hybrid approach for side-channel analysis using efficient convolutional neural network and dimensionality reduction. *IEEE Access* 8 (2020), 184298–184311. https://doi.org/10.1109/ACCESS.2020.3029206

[70] Erick Nascimento and Lukasz Chmielewski. 2017. Applying horizontal clustering side-channel attacks on embedded ECC implementations. In *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017*, Lugano, Switzerland, November 13–15, 2017, Revised Selected Papers *(Lecture Notes in Computer Science, Vol. 10728)*, Thomas Eisenbarth and Yannick Teglia (Eds.). Springer, 213–231. https://doi.org/10.1007/978-3-319-75208-2_13

[71] Kashif Nawaz, Dina Kamel, François-Xavier Standaert, and Denis Flandre. 2017. Scaling trends for dual-rail logic styles against side-channel attacks: A case-study. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017*, Paris, France, April 13–14, 2017, Revised Selected Papers *(Lecture Notes in Computer Science, Vol. 10348)*, Sylvain Guilley (Ed.). Springer, 19–33. https://doi.org/10.1007/978-3-319-64647-3_2

[72] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[74] Guilherme Perin, Ileana Buhan, and Stjepan Picek. 2021. Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design*, Shivam Bhasin and Fabrizio De Santis (Eds.). Springer International Publishing, Cham, 53–81.

[75] Guilherme Perin, Lukasz Chmielewski, Lejla Batina, and Stjepan Picek. 2020. Keep it unsupervised: Horizontal attacks meet deep learning. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 1 (Dec. 2020), 343–372. https://doi.org/10.46586/tches.v2021.i1.343-372

[76] Guilherme Perin, Lukasz Chmielewski, and Stjepan Picek. 2020. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 4 (Aug. 2020), 337–364. https://doi.org/10.13154/tches.v2020.i4.337-364

[77] Guilherme Perin and Stjepan Picek. 2020. 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 (Lecture Notes in Computer Science, Vol. 12804)*, Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn (Eds.). Springer, 615–636. https://doi.org/10.1007/978-3-030-81652-0_24

[78] Guilherme Perin, Lichao Wu, and Stjepan Picek. 2021. AISY - Deep Learning-based Framework for Side-channel Analysis. Cryptology ePrint Archive, Report 2021/357. https://ia.cr/2021/357.

[79] Guilherme Perin, Lichao Wu, and Stjepan Picek. 2021. Exploring Feature Selection Scenarios for Deep Learning-based Side-Channel Analysis. Cryptology ePrint Archive, Report 2021/1414. https://ia.cr/2021/1414.

[80] Guilherme Perin, Lichao Wu, and Stjepan Picek. 2022. *Gambling for Success: The Lottery Ticket Hypothesis in Deep Learning-Based Side-Channel Analysis.* Springer International Publishing, Cham, 217–241. https://doi.org/10.1007/978-3-030-97087-1_9

[81] Stjepan Picek. 2019. 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 (Lecture Notes in Computer Science, Vol. 11947)*, Shivam Bhasin, Avi Mendelson, and Mridul Nandi (Eds.). Springer, 9–12. https://doi.org/10.1007/978-3-030-35869-3_3.

[82] S. Picek, A. Heuser, A. Jovic, and L. Batina. 2019. A systematic evaluation of profiling through focused feature selection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 12 (2019), 2802–2815.

[83] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. 2018. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 1 (Nov. 2018), 209–237. https://doi.org/10.13154/tches.v2019.i1.209-237

[84] Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A. Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. 2017. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks, IJCNN 2017*. IEEE, 4095–4102.

[85] Stjepan Picek, Annelie Heuser, Guilherme Perin, and Sylvain Guilley. 2022. Profiled side-channel analysis in the efficient attacker framework. In *Smart Card Research and Advanced Applications*, Vincent Grosso and Thomas Pöppelmann (Eds.). Springer International Publishing, Cham, 44–63.

[86] Thomas Popp and Stefan Mangard. 2005. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3659)*, Josyula R. Rao and Berk Sunar (Eds.). Springer, 172–186. https://doi.org/10.1007/11545262_13

[87] Emmanuel Prouff, Rémi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. 2018. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptol. ePrint Arch.* 2018 (2018), 53.

[88] Sihang Pu, Yu Yu, Weijia Wang, Zheng Guo, Junrong Liu, Dawu Gu, Lingyun Wang, and Jie Gan. 2018. Trace augmentation: What can be done even before preprocessing in a profiled SCA? In *Smart Card Research and Advanced Applications*, Thomas Eisenbarth and Yannick Teglia (Eds.). Springer International Publishing, Cham, 232–247.

[89] Jean-Jacques Quisquater and David Samyde. 2001. ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, Isabelle Attali and Thomas Jensen (Eds.). Springer Berlin, Berlin, 200–210.

[90] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. 2020. SCARL: Side-channel analysis with reinforcement learning on the Ascon authenticated cipher. *CoRR* abs/2006.03995 (2020). arXiv:2006.03995. https://arxiv.org/abs/2006.03995.

[91] K. Ramezanpour, P. Ampadu, and W. Diehl. 2020. SCAUL: Power side-channel analysis with unsupervised learning. *IEEE Trans. Comput.* 69, 11 (Nov. 2020), 1626–1638. https://doi.org/10.1109/TC.2020.3013196

[92] Christian Rechberger and Elisabeth Oswald. 2004. Practical template attacks. In *Information Security Applications, 5th International Workshop, WISA 2004 (Lecture Notes in Computer Science, Vol. 3325)*, Chae Hoon Lim and Moti Yung (Eds.). Springer, 440–456.

[93] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. 2021. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 3 (Jul. 2021), 677–707. https://doi.org/10.46586/tches.v2021.i3.677-707

[94] Unai Rioja, Lejla Batina, and Igor Armendariz. 2020. When similarities among devices are taken for granted: Another look at portability. In *Progress in Cryptology - AFRICACRYPT 2020*, Abderrahmane Nitaj and Amr Youssef (Eds.). Springer International Publishing, Cham, 337–357.

[95] Riscure. 2018. CHES CTF 2018. Website. https://chesctf.riscure.com/2018/news.

[96] Damien Robissout, Gabriel Zaid, Brice Colombier, Lilian Bossuet, and Amaury Habrard. 2020. 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 (Lecture Notes in Computer Science, Vol. 12244)*, Guido Marco Bertoni and Francesco Regazzoni (Eds.). Springer, 200–218. https://doi.org/10.1007/978-3-030-68773-1_10

[97] Thomas Roche, Victor Lomné, Camille Mutschler, and Laurent Imbert. 2021. A side journey to Titan. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 231–248. https://www.usenix.org/conference/usenixsecurity21/presentation/roche.

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

[99] Karen Simonyan and Andrew Zisserman. 2015. 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*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.1556.

[100] Arvind Singh, Monodeep Kar, Sanu Mathew, Anand Rajan, Vivek De, and Saibal Mukhopadhyay. 2019. A 128b AES engine with higher resistance to power and electromagnetic side-channel attacks enabled by a security-aware integrated all-digital low-dropout regulator. In *IEEE International Solid- State Circuits Conference, ISSCC 2019, San Francisco, CA, USA, February 17–21, 2019*. IEEE, 404–406. https://doi.org/10.1109/ISSCC.2019.8662344

[101] François-Xavier Standaert and Cédric Archambeau. 2008. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *Cryptographic Hardware and Embedded Systems - CHES 2008. Proceedings (Lecture Notes in Computer Science, Vol. 5154)*, Elisabeth Oswald and Pankaj Rohatgi (Eds.). Springer, 411–425.

[102] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. 2009. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009*, Antoine Joux (Ed.). Springer Berlin, Berlin, 443–461.

[103] Sudharshan Swaminathan, Lukasz Chmielewski, Guilherme Perin, and Stjepan Picek. 2021. Deep learning-based side-channel analysis against AES inner rounds. *IACR Cryptol. ePrint Arch.* 2021 (2021), 981. https://eprint.iacr.org/2021/981.

[104] TELECOM ParisTech SEN research group. 2010. DPA_V2. Website. http://www.dpacontest.org/v2/.

[105] TELECOM ParisTech SEN research group. 2013. DPA_V4.1. Website. http://www.dpacontest.org/v4/.

[106] TELECOM ParisTech SEN research group. 2014. DPA_V4.2. Website. http://www.dpacontest.org/v4/42_doc.php.

[107] Dhruv Thapar, Manaar Alam, and Debdeep Mukhopadhyay. 2021. Deep learning assisted cross-family profiled side-channel attacks using transfer learning. In *22nd International Symposium on Quality Electronic Design, ISQED 2021*. IEEE, 178–185.

[108] Benjamin Timon. 2019. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 2 (2019), 107–131. https://doi.org/10.13154/tches.v2019.i2.107-131

[109] Kris Tiri and Ingrid Verbauwhede. 2004. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16–20 February 2004, Paris, France*. IEEE Computer Society, 246–251. https://doi.org/10.1109/DATE.2004.1268856

[110] Daan van der Valk and Stjepan Picek. 2019. Bias-variance Decomposition in Machine Learning-based Side-channel Analysis. Cryptology ePrint Archive, Report 2019/570. https://eprint.iacr.org/2019/570.

[111] Daan van der Valk, Stjepan Picek, and Shivam Bhasin. 2021. Kilroy was here: The first step towards explainability of neural networks in profiled side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design*, Guido Marco Bertoni and Francesco Regazzoni (Eds.). Springer International Publishing, Cham, 175–199.

[112] Huanyu Wang, Martin Brisfors, Sebastian Forsmark, and Elena Dubrova. 2019. How diversity affects deep-learning side-channel attacks. In *2019 IEEE Nordic Circuits and Systems Conference, NORCAS 2019: NORCHIP and International Symposium of System-on-Chip (SoC)*, Jari Nurmi, Peeter Ellervee, Kari Halonen, and Juha Röning (Eds.). IEEE, 1–7.

[113] Ping Wang, Ping Chen, Zhimin Luo, Gaofeng Dong, Mengce Zheng, Nenghai Yu, and Honggang Hu. 2020. Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks. *CoRR* abs/2007.05285 (2020). arXiv:2007.05285. https://arxiv.org/abs/2007.05285.

[114] Léo Weissbart, Łukasz Chmielewski, Stjepan Picek, and Lejla Batina. 2020. Curve25519 datasets. Dropbox. https://www.dropbox.com/s/e2mlegb71qp4em3/ecc_datasets.zip?dl=0.

[115] Léo Weissbart, Łukasz Chmielewski, Stjepan Picek, and Lejla Batina. 2020. Systematic side-channel analysis of Curve25519 with machine learning. *Journal of Hardware and Systems Security* 4, 4 (Oct. 2020), 314–328. https://doi.org/10.1007/s41635-020-00106-w

[116] Léo Weissbart, Stjepan Picek, and Lejla Batina. 2019. Ed25519 WolfSSL. Github repository. https://github.com/leoweissbart/MachineLearningBasedSideChannelAttackonEdDSA.

[117] Léo Weissbart, Stjepan Picek, and Lejla Batina. 2019. One trace is all it takes: Machine learning-based side-channel attack on EdDSA. In *Security, Privacy, and Applied Cryptography Engineering*, Shivam Bhasin, Avi Mendelson, and Mridul Nandi (Eds.). Springer International Publishing, Cham, 86–105.

[118] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1–3 (1987), 37–52.

[119] Yoo-Seung Won, Xiaolu Hou, Dirmanto Jap, Jakub Breier, and Shivam Bhasin. 2021. Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3215–3227. https://doi.org/10.1109/TIFS.2021.3076928

[120] Yoo-Seung Won, Dirmanto Jap, and Shivam Bhasin. 2020. Push for more: On comparison of data augmentation and SMOTE with optimised deep learning architecture for side-channel. In *Information Security Applications*, Ilsun You (Ed.). Springer International Publishing, Cham, 227–241.

[121] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. 2020. Revisiting a methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020, 3 (2020), 147–168.

[122] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2020. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1293.

[123] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. The best of two worlds: Deep learning-assisted template attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022, 3 (Jun. 2022), 413–437. https://doi.org/10.46586/tches.v2022.i3.413-437

[124] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. On the evaluation of deep learning-based side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design: 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11–12, 2022, Proceedings* (Leuven, Belgium). Springer-Verlag, Berlin, 49–71. https://doi.org/10.1007/978-3-030-99766-3_3

[125] Lichao Wu and Stjepan Picek. 2020. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 4 (Aug. 2020), 389–415. https://doi.org/10.13154/tches.v2020.i4.389-415

[126] Lichao Wu, Léo Weissbart, Marina Krček, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. 2020. On the Attack Evaluation and the Generalization Ability in Profiling Side-channel Analysis. Cryptology ePrint Archive, Report 2020/899. https://eprint.iacr.org/2020/899.

[127] Lichao Wu, Yoo-Seung Won, Dirmanto Jap, Guilherme Perin, Shivam Bhasin, and Stjepan Picek. 2021. Explain Some Noise: Ablation Analysis for Deep Learning-based Physical Side-channel Analysis. Cryptology ePrint Archive, Report 2021/717. https://ia.cr/2021/717.

[128] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of CNN and RNN for natural language processing. *CoRR* abs/1702.01923 (2017). arXiv:1702.01923. http://arxiv.org/abs/1702.01923.

[129] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. 2021. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021, 1 (2021), 25–55. https://doi.org/10.46586/tches.v2021.i1.25-55

[130] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. 2019. Methodology for efficient CNN architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 1 (Nov. 2019), 1–36. https://doi.org/10.13154/tches.v2020.i1.1-36

[131] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. 2021. Efficiency through diversity in ensemble models applied to side-channel attacks - a case study on public-key algorithms -. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021, 3 (2021), 60–96. https://doi.org/10.46586/tches.v2021.i3.60-96

[132] Fan Zhang, Bin Shao, Guorui Xu, Bolin Yang, Ziqi Yang, Zhan Qin, and Kui Ren. 2020. From homogeneous to heterogeneous: Leveraging deep learning based power analysis across devices. In *57th ACM/IEEE Design Automation Conference, DAC 2020*. IEEE, 1–6.

[133] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. 2020. 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* 2020, 3 (Jun. 2020), 73–96. https://doi.org/10.13154/tches.v2020.i3.73-96

[134] Ziyue Zhang, A. Adam Ding, and Yunsi Fei. 2020. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020, 2 (2020), 26–48.

[135] Yuanyuan Zhou and François-Xavier Standaert. 2020. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized ResNet model for side-channel attacks. *J. Cryptogr. Eng.* 10, 1 (2020), 85–95.

Received 14 January 2022; revised 5 August 2022; accepted 9 October 2022