When Theory Meets Practice: A Framework for Robust Profiling Side-channel Analysis

Stjepan Picek[1], Annelie Heuser[2], Lichao Wu[1], Cesare Alippi[3,4], and Francesco Regazzoni[5,4]

[1] Delft University of Technology, The Netherlands
[2] CNRS, IRISA, Rennes, France
[3] Politecnico di Milano, Milano, Italy
[4] Università della Svizzera Italiana, Lugano, Switzerland
[5] University of Amsterdam, The Netherlands

Abstract. Profiling side-channel attacks are considered the most potent form of side-channel attacks. They consist of two steps. First, the adversary builds a leakage model using a device similar to the target one. This leakage model is then exploited to extract the secret information from the victim's device. These attacks can be seen as a classification problem, where the adversary needs to decide to what class (and consequently, the secret key) the traces collected from the victim's device belong. The research community investigated profiling attacks in-depth, primarily by using an empirical approach. As such, it emerges that a theoretical framework to analyze profiling side-channel attacks comprehensively is still missing.

In this paper, we propose a theory-grounded framework capable of modeling and evaluating profiling side-channel analysis. The framework is based on the expectation estimation problem that has strong theoretical foundations. We quantify the effects of perturbations injected at different points in our framework through the robustness analysis, where the perturbations represent sources of uncertainty associated with measurements, non-optimal classifiers, and countermeasures. Finally, we use our framework to evaluate the performance of different classifiers using publicly available traces.

1 Introduction

Embedded and cyber-physical devices, connected to form the Internet of Things (IoT), are pervading every aspect of our lives. Even though they provide fundamental services, their use of sensitive data and access to critical infrastructure brings new security challenges. It follows that security becomes one of the most important extra-functional requirements that the designer should grant to the devices. Designing secure embedded devices is extremely challenging for two main reasons. First, the limited area and energy budget available in these devices are often insufficient to implement full flagged and robust cryptographic primitives. Second, these devices should be resistant to physical attacks. The

pervasive diffusion of these devices makes them physically available to adversaries willing to exploit the implementation's physical weaknesses to extract the stored secret information (typically, the secret key).

It is necessary to understand the adversary's capabilities to achieve resistance against physical attacks. Side-channel attacks (SCAs), in particular profiling ones, are by far the most studied (and the most powerful) physical attacks since they have been proved to be very effective both in the lab and in real-world applications [31]. In profiling attacks, the adversary first profiles a device identical (or, at least similar) to the one that will be attacked. In a second phase, using this profile and the victim device's traces, the adversary attempts to recover the secret key. Common examples of such profiling attacks are template attack [7,9] as well as supervised machine learning-based attacks [6,19,23,34,35,49].

It is well-known that a template attack is the most powerful one from the information-theoretic perspective (given that certain assumptions hold) [7]. Simultaneously, numerous related works experimentally showed how machine learning techniques perform extremely well in many realistic scenarios, see, e.g., [6,49]. However, there is no guarantee that machine learning techniques would behave similarly in different scenarios, even if they share characteristics. Still, one would hope to be able to get answers to more specific questions, e.g.:

1. What machine learning method is optimal for a given side-channel scenario?
2. What machine learning method should be preferred for multiple side-channel scenarios?
3. Which machine learning method is the most stable one?

Note that, by answering those questions, we also provide insights into connections among different settings or even countermeasures.

The current state-of-the-art in profiling side-channel analysis progressed tremendously in the last few years. There, deep learning results showed it is possible to break even implementations protected with countermeasures [6,23]. More recently, the SCA community started investigating the explainability of machine learning-based attacks to produce more powerful attacks and novel countermeasures [32,46]. Besides, the robustness of specific classifier-distinguisher combinations has been studied in [47]. However, to the best of our knowledge, the achieved results do not offer strong theoretical insights and theory-based approaches on the evaluation of the performance of general machine learning-based attacks. Unfortunately, the exact values for the simulated noise are not openly available, making a direct comparison with out results not possible. Nevertheless, the authors used an approach rather similar to ours to simulate datasets with added noise, confirming that the simulation-based approach we used in this paper is well justified. Consequently, a general framework capable of modeling (and thus, fairly comparing) all the profiling attacks, and their characteristics, even when used in different conditions, is still missing.

To address this need, in this paper, we propose a framework based on the analysis of problems approached in probability, which models *all* the profiling attacks, allows to analyze their behavior and their performance and gives a rigorous insight into their robustness. We leverage on problems investigated in

probability since, similarly to that class of problems, also in the case of profiling side-channel attacks, we are not searching *the* solution, but a valid answer according to a defined probabilistic figure of merit (which in our case can be the accuracy of the classifier or the guessing entropy). Furthermore, as in the case of problems approached in probability, we are interested in knowing to what extent our result has to be considered correct and what maximum error we could obtain. Besides, a formal approach for the analysis (robustness) and changes in the setup (noise), is still missing. We believe that this is crucial missing information for researchers and evaluation labs.

Our framework is designed to analyze the behavior of profiling SCAs and give insights into the robustness of such methods. By robustness, we consider the system's ability to tolerate perturbations (random changes affecting the system). Thus, robustness (or a lack of it) can be considered as a consequence of adding perturbations to the system. Thus we use robustness analysis as a tool to obtain insights into the general behavior of profiling attacks, since perturbations are also commonly occurring in SCA. Different sources of perturbations occur in profiling SCA due to the presence of:

1. Environment noise.
2. Countermeasures.
3. The differences between the profiling device and the device under attack (portability).

While the first source of perturbation is immediate, the other two deserve further explanation. Consider a system with a countermeasure. The profiling attack does not "know" what the countermeasure is but can only "see" its consequence. This discrepancy can be successfully modeled as a random variable whose realizations are associated with perturbations affecting the system's functionality. Finally, while it is common to use the same device for both training and testing, in reality, there are two devices (the first one to train a model and the second one to attack). This simplification is reasonable but will introduce errors where the attack's performance will be lower than the one measured in experiments with only one device [3]. Additionally, the difference in the measurements in practice can also arise, for instance, from different probes positions when using electromagnetic (EM) SCA.

By considering the robustness paradigm, we evaluate a setting that approximates the realistic one, where perturbations must occur and uncertainty is present. That paradigm is also in the core of the well-known Provably Approximate Correct (PAC) learning [44], where PAC learning theory formalizes the way computation is carried out within an uncertainty affected environment.

The main contributions of this paper are:

1. We propose a framework capable of modeling and evaluating profiling side-channel attacks with strong theoretical foundations.
2. We consider the robustness of profiling attacks in 1) the presence of countermeasures and 2) environment settings like feature selection, dataset size, and hyperparameter tuning.

3. We consider the robustness of profiling attacks for different figures of merit: accuracy, success rate, and guessing entropy.

The rest of this paper is organized as follows. In Section 2, we discuss related works and directions commonly investigated in the profiling side-channel analysis. Section 3 first discusses the threat model, and then it presents an intuitive description of our framework. Afterward, we formally define it through the expectation estimation problem. Next, Section 4 presents details about our experimental setting: profiling methods, datasets, figures of merit, and framework parameters we investigate. In Section 5, we validate our framework by using a technique called stylized facts that compares the behavior of simulations and real-world data. Additionally, we present an experimental evaluation for publicly available datasets. Section 6 presents a discussion about general findings from the experiments and advantages/drawbacks of using our framework. Finally, Section 7 concludes the paper.

2 Related Work

In 1996, Kocher demonstrated the possibility to recover secret data by introducing a method for exploiting the leakages from the device under attack [24]. This is possible because implementations of cryptographic algorithms leak relevant information about the data processed through physical side-channels like timing [24], power consumption [25], EM emanation [36], or sound [14]. The side-channel attacks are usually divided into two groups: direct attack (see, e.g., Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [25]) and profiling attacks (see, e.g., Template Attack (TA) [7], stochastic models [40], or a number of machine learning-based techniques as discussed next).

Profiling attacks are carried out in two stages, profiling and attacking. In the profiling stage, the adversary has full control of cryptographic hardware and can estimate the leaked information's probability distribution. In the second stage, the estimated distribution is used to extract the secret information from a victim device. Profiling attacks are the most powerful form of side-channel attacks. Because of this, they define the worst-case security assumptions. TA is the best (optimal) technique from an information-theoretic perspective if the attacker has an unbounded number of traces, and the noise follows the Gaussian distribution [18, 29]. After the template attack, the stochastic models technique emerged, which uses linear regression in the profiling phase [40]. In years to follow, researchers recognized certain shortcomings of template attacks and tried to modify them in order to deal better with the complexity and portability issues. One example of such an approach is the pooled template attack, where one pooled covariance matrix is used to cope with statistical difficulties [9].

Alongside such techniques, the SCA community realized that a similar approach to profiling is used in other domains in the form of supervised machine learning. Consequently, the researchers started experimenting with different machine learning methods and evaluating their effectiveness in the SCA context. A survey of available works reveals several papers that discuss different machine

learning methods, targets, and experimental settings. When considering attacks on AES (as we do in this paper) and supervised machine learning methods, one can find a number of papers, see, e.g., [15,17,19,21,26–29,33,35]. More recently, deep learning techniques have come into the focus of the SCA community, while most of the attention went to convolutional neural networks [6,23,30,34,49].

When trying to find a common denominator for those works, the most obvious one is that they report machine learning being able to reach high attack performance and often outperforming template attacks. Additionally, deep learning can commonly break the implementations protected with countermeasures (at least those available in the publicly available datasets). Still, all of these works have "only" experimental results with sporadic attempts in explaining why such results are obtained. Besides works that consider how to improve the attack performance, more recently, there have been several works considering the topics of explainability and interpretability of deep learning in side-channel analysis [32,45,46].

Other aspects of the profiling side-channel attacks have been studied already in the past. The problem of comparing profiling side-channel attacks has been tackled by Standaert et al. [42] They proposed to use the information theory-based methodology to investigate the performance of the different phases of profiling side-channel attacks on exemplary simulated leakages [43]. Their results show that the quality of the profiling phase can be captured by an information-theoretic metric, while the key recovery phase is better measured with a security metric. The analysis of profiling side-channel attacks can also benefit from simplification and optimization. Among such works, a relevant one is from Bronchain et al., where the authors propose a quantitative tool for leakage certification [5]. Quantitative analysis is carried out by bounding the unknown mutual information metric on perceived information and on the hypothetical information. Durvaux et al. addressed the problem of bias in the security evaluation caused by estimation and assumption errors using sound statistical techniques to quantify the leakage of a device and guarantee that the amount of information extracted is close to the maximum achievable with a perfect model [10].

3 General Framework

In this section, we start by we introducing the threat model. Afterward, we provide intuitive description of our framework, followed by the discussion on the expectation estimation problem and robustness analysis. Finally, we discuss how to introduce the robustness analysis into the profiling SCA framework.

3.1 Threat Model

We investigate a typical profiling side-channel setting. We consider this to be a de-facto standard as a number of certification laboratories are evaluating hundreds of security-critical products with this model daily.

In this setting, the adversary has access to a clone device running the target cryptographic algorithm. The clone device can be queried with a known key and plaintext while the corresponding leakage trace is stored. Ideally, the adversary can perform an infinite number of queries and can construct the corresponding database of side-channel leakage traces to characterize a precise profiling model. After the characterization is done, the adversary queries the victim device with known plaintext to obtain the secret key. The key recovery is carried out by comparing the side-channel leakage traces from the victim device with the profiling model previously characterized.

Since the focus of this paper is power and electromagnetic attacks, in the rest of the paper, when discussing profiling attacks, we consider profiling attacks that use power or electromagnetic radiation as side-channel. Furthermore, in this paper, we consider only those scenarios where the paradigm is supervised learning, and the task is classification, i.e., to learn a mapping f between a set of input variables X an output variable Y ($f : X \; ! \; Y$).

3.2 Intuitive Description of the Framework

The previous works on profiling side-channel attacks discussed in Section 2 provide a detailed overview of the attacks and methodology used. Still, they do not abstract the specificity of the attack to a higher-level framework. As a result, a complete analysis of the attack characteristics is empirical, and so is the direct comparison of different profiling techniques.

Figure 1 depicts the generic framework we propose to model profiling side-channel attacks. Recall, during such attacks, the adversary attempts to recover the secret (typically, the key) of a cryptographic device in two phases, commonly known as the profiling and attacking phase. The profiling of the device D (the physical realization of a cryptographic primitive), depicted on the right part of the figure, consists of collecting several leakage traces $x_i; 1 \quad i \quad n$ where n is the number of traces taken corresponding to the encryption of a certain number of plaintexts PT and a number of known keys K. For each trace x_i, we obtain T time samples (also known as features or attributes): $x_i = x_{i,1}; \ldots; x_{i,T}$. During the profiling phase, for each of the traces x_i, we additionally obtain the label of the trace y_i, which denotes the actual value the trace has (where the connection between the label and the key/plaintext includes the sensitive operation like the S-box and an appropriate leakage model). The leaked traces are typically altered by a random perturbation $_1$ that can be caused by measurement or algorithmic noise but also by the operation of a side-channel countermeasure. The leakage trace and the corresponding plaintext are used to derive an estimate of the secret key $\hat{K}$. These estimates of the secret key, along with the traces, are used to train a classifier C, depicted in the left part of Figure 1. The specific classifier can be arbitrarily selected from the corpus of all possible classifiers suitable for side-channel recovery. Each classifier is trained on the training set in the sense that different training sets of the same size will generate different classifier (profiling) models. To model this phenomenon, we consider the given classifier's output influenced by a specific perturbation $_2$, which implies we

are assuming that the intensity (variance) of two perturbations can be the same (i.e., different points are characterized by the same signal-to-noise ratio (SNR) or values affected by perturbations insist on the same bounded interval). Without loss of generality, we consider $_1$ and $_2$ to be equal to since here, we do not differentiate between the perturbations coming from one or the other source. The estimated values of the labels $\hat{y}_i$ are compared to the actual ones and used to estimate the classifier's robustness.

The robustness problem we consider here can be solved by considering the robustness of randomized algorithms [1]. In the particular randomized algorithm setting, we are interested in quantifying the robustness of an algorithm utilizing a suitable figure of merit. In our setting, we aim to quantify the robustness of a profiling side-channel attack (first seen as a supervised machine learning problem) to the perturbations caused by the measurement noise or countermeasures but also the intrinsic noise of a specific classifier. The framework we consider is general, which means it supports any profiled/supervised method and model, as well as any figure of merit. To validate our framework, in Section 5, we select to work with a number of common SCA classifiers and figures of merit.

Our framework proposes one theoretical interpretation for an analysis that is currently mostly done empirically. Consequently, we can achieve two goals:

1. The connection with well-understood problems (expectation estimation problem and robustness problem) allows us to model each profiling side-channel attack and thus compare, in an objective and rigorous way, the performance of different classifiers in a specific scenario.
2. A quantitative estimate of the confidence of our results. Ultimately, we will be able to answer the questions like "Which classifier family behaves better in some specific scenario?" and "How to compare different profiling side-channel attacks?".

3.3 Expectation Estimation Problem and Robustness

This section aims to answer two questions that will be used in subsequent derivations and provide some common nomenclature. First, how to estimate the expected value of a function by identifying the minimum number of samples granting to build an approximation with an arbitrary level of accuracy and confidence (Section 3.3). Second, how to estimate the robustness of a system once affected by perturbations. To achieve this, we introduce the basic theory behind the expectation estimation problem and robustness analysis.

The Cherno Bound The framework for expectation estimation we use is built starting from the Chernoff bound. In a nutshell, the Chernoff bound is a way to estimate the probability distribution of an unknown variable. The main advantage of the Chernoff bound is that it has extremely mild and quite generic assumptions that are fulfilled by an extremely large number of problems. Informally, the requirement of the Chernoff bound we use here is simply that the variable we are considering is bounded.

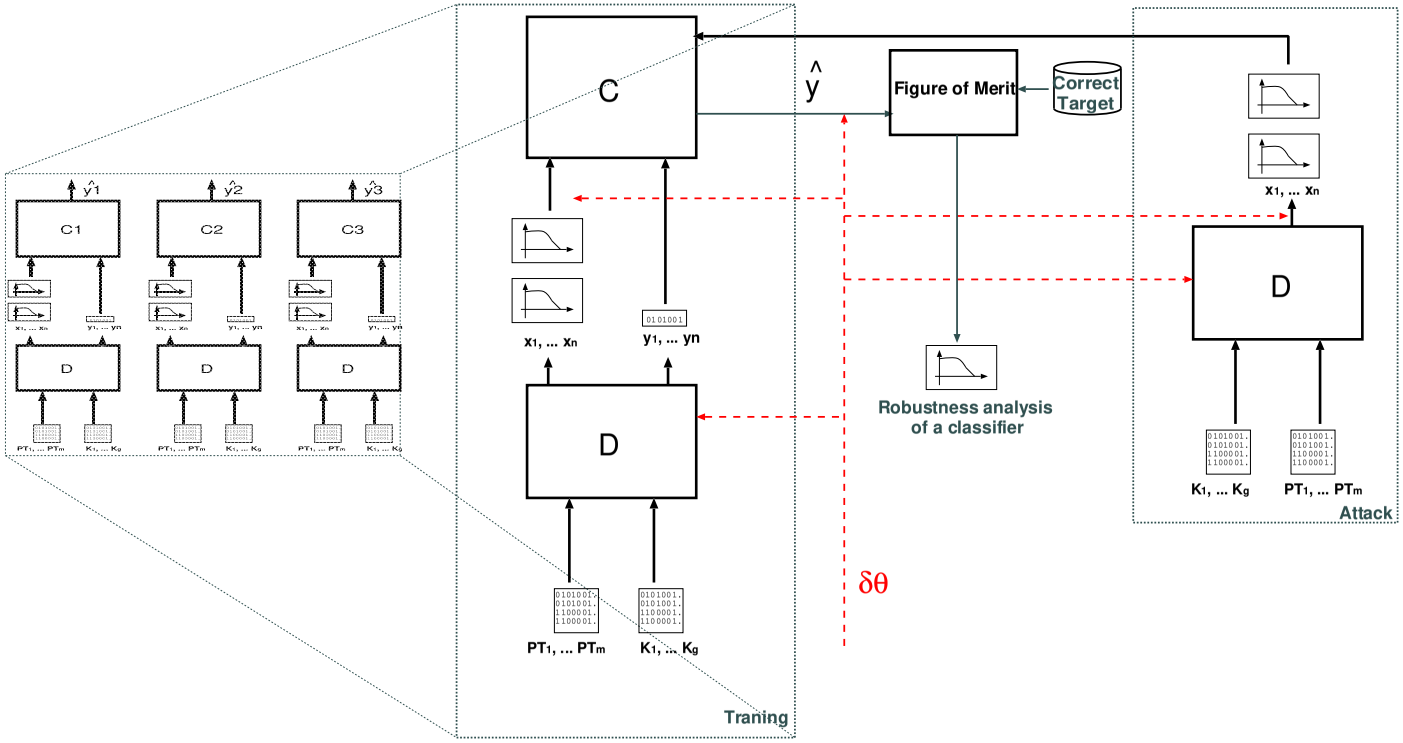


Fig. 1: The framework for the profiling side-channel analysis. The red line denotes the locations of possible perturbations. There are two phases of the attack. Profiling phase (left part of the figure in bold) and attack phase (right part of the figure in bold). Since the framework is generic, it supports any profiling attack, as denoted by several different classifiers on the far left side. This framework supports any: 1) profiling attack, 2) leakage model, 3) dataset (considering, e.g., side-channel information, level of noise, countermeasures used, dataset size).

More precisely, the Chernoff bound allows determining the number of samples needed to estimate a probability with arbitrary accuracy in the estimate approximation and confidence in the made statement. The Chernoff bound can also be used to estimate the expected value of random variable according to estimation accuracy and confidence levels set by the designer [8]. We now define the Chernoff bound formally. The Chernoff bound for a generic probability density function and continuous variable can be derived from the Hoeffding inequality for the empirical mean [20].

Let $x_1; \quad x_n$ be a sequence of independent random variables so that each x_i is almost surely bounded by the interval $[a_i; b_i]$, i.e., $\Pr(x_i \ 2 \ [a_i; b_i]) = 1$. Then, defining the empirical mean $\hat{E}_n = \frac{1}{n} \sum_{i=1}^{n} x_i$, we have that for any value the Hoeffding inequality for the empirical mean:

$$\Pr \quad j\hat{E}_n \quad E[\hat{E}_n]j \qquad 2e^{\frac{-2 \ ^2 n^2}{\sum_{i=1}^{n}(b_i - a_i)^2}} \tag{1}$$

holds where E is the expectation operator. Note that in the rest of the paper, we use $u(\ )$ notation to denote a generic function accepting any number of arguments, and we specify the parameters and the role of the function in the text. Eq. (1) can be rewritten as:

$$\Pr \quad j\hat{E}_n \quad E[\hat{E}_n]j < \quad > 1 \quad 2e^{\frac{-2 \ ^2 n^2}{\sum_{i=1}^{n}(b_i - a_i)^2}}: \tag{2}$$

If $\hat{E}_n$ is the estimate $\hat{p}_n(\ )$ of a probability [6], e.g., $(p(\ ) = \Pr(u(\ ) \quad)$ for a given positive scalar and a loss function $u(\ ))$, we have that for a generic random variable $_i$ the indicator function:

$$x_i = I \ (u(\ _i) \quad) = \begin{array}{l} 1 \text{ if } u(\ _i) \\ 0 \text{ if } u(\ _i) > \end{array}$$

assumes values in $f0; 1g$. As a consequence, $a_i = 0; b_i = 1$ and Eq. (2) becomes

$$\Pr \quad j\hat{E}_n \quad E[\hat{E}_n]j < \quad > 1 \quad 2e \ ^{2n\epsilon^2}:$$

Since, $\hat{p}_n(\ ) = \hat{E}_n$ and $E[\hat{p}_n(\ )] = p(\ )$ we derive

$$\Pr \ (j\hat{p}_n(\ ) \quad p(\ )j < \ ) > 1 \quad 2e \ ^{2n\epsilon^2}: \tag{3}$$

Finally, we derive the Chernoff bound by requesting confidence $2e \ ^{2n\epsilon^2}$:

$$n \quad \frac{1}{2 \ ^2} \ln \frac{2}{-}: \tag{4}$$

The Chernoff bound states that if we sample from the domain of random variable according to its probability density function, then the Eq. (4) holds with confidence . In other words, we can build an approximation $\hat{p}_n$ of unknown probability $p(\ )$ with accuracy , and the statement will hold with confidence .

[6] Note that may not always be the case, but we consider such "interesting" cases in this paper.

The Expectation Estimation Problem The expectation estimation problem we consider is based on the Chernoff bound. In its generic formulation, the expectation estimation problem consists of identifying the minimum number of samples needed to achieve an arbitrary level of accuracy and confidence in approximating the expected value of a given function $u(\ )$.

Let $u(\ ) \in [0;1]$ be a Lebesgue measurable function (as defined in Appendix A) over $\mathbb{R}^l$ and f_ψ be the probability density function of a random variable defined over . The expectation estimation requires evaluation of the expected mean:

$$E[u(\ )] = \int_{\Psi} u(\ ) f_\psi(\ ) d \ : \tag{5}$$

Since the evaluation of the expected mean defined in Eq. (5) is generally computationally hard problem for a generic function, an approximation is built starting from n i.i.d. samples $_1; \ ; \ _i; \ ; \ _n$ drawn from according to f_ψ. We call

$$\hat{E}_n(u(\ )) = \frac{1}{n} \sum_{i=1}^{n} u(\ _i) \tag{6}$$

the empirical mean. It should be commented that $\hat{E}_n(u(\ ))$ is a random variable depending on the particular realization of the n samples. The Chernoff bound can be used to build an accurate approximation of Eq. (5) through Eq. (6) at accuracy level and confidence .

$\hat{E}_n(u(\ ))$ is the estimate of the figure of merit. By assuming that the condition $u(\ ) \in [0;1]$ we immediately derive the bound on n thanks to the Hoeffding's inequality. In general, it is enough to require $u(\ _i)$ to be bounded, e.g., to the same $a_i = a; b_i = b; i = 1; : : : ; n$. If that is the case, the bound on the number of samples becomes:

$$n \quad \frac{(b \quad a)^2}{2 \ ^2} \ln \frac{2}{} : \tag{7}$$

Note, there is a connection between the expectation estimation problem we consider here and the probability estimation problem commonly considered in SCA. In fact, the expectation estimation problem (including the Chernoff bounds) is computed starting from the expected values, ultimately relying on values obtained through the probability estimation problem.

Robustness Robustness of a system refers to the ability to tolerate perturbations that might affect its structural parameters and, in turn, its performance, measured utilizing a given figure of merit. In practice, what is desirable is that the system continues to work even in the presence of a perturbation, and what is to be estimated is the capability of correctly operating when perturbations occur. In our case, the system is the profiling attack, and we want to evaluate the robustness of such attacks for all possible effects that can affect the success of an attack (measurement noise, portability of the classifier, countermeasures). Ultimately, we want to understand under which conditions the attack is still

possible. More formally, a system/function $g(\ ;x)$ is robust with respect to perturbations $\ \in\ \in R^l$ at level $\ \in \mathbb{R}^+$ when, given a discrepancy function $u(g(\ ;x);g(\ ;\ ;x)) \in U \quad \mathbb{R}$ the system experiences a degradation in performance within $\ :$

$$u(\ \ ) = u(g(\ ;x);g(\ ;\ \ ;x)) \quad ;\ \ \forall \ \ \in \ ;\forall x \in \tilde{X}: \tag{8}$$

In the rest of the paper, we assume that the perturbation level can become arbitrarily large so that no small perturbation theories are viable. $u(\ \ )$ represents the perturbation impact on the system's behavior, as observed through the figure of merit. Note that $u(\ \ )$ does not have an explicit function in the inputs in the sense that if inputs are in there, they are finite in number and fixed and belong to the set $\tilde{X}$, which is the discrete set containing a finite number of input instances.

In this setting, we need to determine the smallest $\ $ satisfying the previous expression. Since this can be computationally intractable, we move to a probabilistic setting. There, a computation is robust at level $\ $ with probability $1 \quad$ for the perturbation space $\ $ when $\ $ is the smallest value such that $\Pr(u(\ \ ) \quad) \quad 1 \quad ;\ \forall \ \ \in \ $. Here, $\ $ is a small positive value in $[0;1]$ and $1 \quad$ is the confidence level.

Once defined $p(\ )$ to be the probability that $u(\ \ ) \quad$ for an arbitrary but given $\ $ value:

$$p(\ ) = \Pr(u(\ \ ) \quad) \ \textit{for each} \ \ \in \ : \tag{9}$$

Eq. (9) can be estimated with Chernoff and the minimum performance level (or degradation of that level, depending on if the smaller value is better or vice versa) identified through the performance level set $\ = \{\ _1;\ \ ;\ _k\}$. More precisely, we use the Chernoff bound to build an approximation of unknown probability $p(\ )$ with accuracy $\ $ for a confidence $\ $ as detailed in the example below.

Next, we provide an example for profiling SCA.

Example 1. Let us consider a setup with accuracy $\ = 0.1$ and confidence $\ = 0.1$. The Chernoff bound (Eq. (4)) gives $n \quad 149.7$ to achieve the desired level of accuracy and confidence. This means that we need to repeat the experiment 150 times to understand if the algorithm (classifier) is robust. Next, let us consider a setting where the perturbation impact $u(\ \ )$ is 10% of the signal. Then, we can ask what the degradation of the performance $\ $ considering guessing entropy is. More precisely, what will be the change in the guessing entropy value caused by perturbing the system?

Remark 1. We note that the experiment as discussed in Example 1 can be done for one or more perturbation levels and one or more performance levels $\ \in \ $.

3.4 Profiling SCA Framework

We now express profiling side-channel attacks as the expectation estimation problem. The starting point is to map the steps of profiling analysis to the

framework depicted in Figure 1. The first phase of profiling side-channel attacks is the training phase, which is represented by the bold part of Figure 1. The training phase begins with the collection of the traces corresponding with the encryption of several plaintext and keys.

Formally, during the encryption, the secret key k is processed with t plaintexts or ciphertexts of the cryptographic algorithm, while the attacker collects a set of traces x. In the case of AES, typically k and t are processed in bytes, which reduces the attack complexity. The mapping y maps the plaintext or the ciphertext $t \; 2 \; T$ and the key $k \; 2 \; K$ to a value that is assumed to relate to the deterministic part of the measured leakage x. We denote the output of y as the label, which is coherent with the terminology used in the machine learning community. For profiling, there are two common models to define $y(t; k)$ so to calculate the labels of the measurement traces:

- *intermediate value (ID) model*: in this leakage model, the attacker considers an intermediate value of the cipher or the distance between two consecutive values processed. When considering the AES cipher, this leakage model results in 256 labels.
- *Hamming weight (HW) model*: in this leakage model, the attacker assumes the HW of the intermediate value model. When considering the AES cipher, this leakage model results in 9 labels.[7]

Considering the intermediate value, the profiling model may be more accurate but requires more resources. In particular, to gain stable estimations for each possible value, an attacker needs a sufficient number of measurements per value. Additionally, as the attacker needs to iterate through all values in the profiling phase as well as for each measurement in the attacking phase, the computational complexity may become high - especially when targeting ciphers operating on more than 8-bit.

The HW model's preference is related to the underlying device (e.g., for some devices, the power consumption is assumed to be roughly proportional to the number of bit transitions) and the lower complexity. In our analysis, we consider both leakage models for all datasets and profiling methods.

In the attack phase, the goal is to make predictions about the occurring labels

$$y(t_{a_1}; k_a); : : : ; y(t_{a_N}; k_a);$$

where k_a is the secret unknown key on the device under the attack.

4 Experimental Setting

This section discusses the datasets, machine learning classifiers, and the framework's experimental settings we consider.

[7] One more common leakage model would be the Hamming distance leakage model, which would also result in 9 labels.

4.1 Datasets

In our experiments, we consider several publicly available datasets representing a typical sample of commonly encountered scenarios and one simulated traces dataset (used for the framework validation). The datasets we use are the de-facto standard for SCA and are commonly considered in machine learning-based SCA research.

ASCAD Datasets The target platform is an 8-bit AVR microcontroller (ATmega8515) running a masked AES-128 implementation, and traces are made using electromagnetic emanation [2]. There are two versions of the ASCAD dataset: one with a fixed key with 50 000 traces for profiling/training and 10 000 for testing. The second version has random keys, and the dataset consists of 200 000 traces for profiling and 100 000 for testing. For both versions, we attack the key byte 3, which is the first masked byte. For the version with the fixed key, we use a pre-selected window of 700 features, while for the version with random keys, the window size equals 1 400 features. These datasets are available at `https://github.com/ANSSI-FR/ASCAD`.

CHES CTF Dataset This contains AES-128 measurements, released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces consist of masked AES-128 encryption running on a 32-bit STM microcontroller. In our experiments, we consider 43 000 traces for the training set, which contains a `xed key`. The validation and test sets consist of 1 000 traces each. The key used in the training and validation set is different from the key for the test set. We attack the first key byte. Each trace consists of 2 200 features. This dataset is available at `https://chesctf.riscure.com/2018/news`.

Simulated Dataset The circuit we simulated to obtain the simulated traces is a reduced portion of the AES algorithm, composed of a key addition followed by an S-box lookup. The obtained data are then stored in a register. The flow used to generate the simulated traces is implemented using state-of-the-art commercial electronic design automation commodities and is derived from the simulation flow presented by Regazzoni et al. [37]. The test circuit is designed using HDL language, synthesized with a synthesis tool (Synopsys design compiler), and placed and routed with an automated tool (Cadence Encounter). The final circuit, together with the parasitic extracted using the extractor build in the place and route tool, are simulated at SPICE level using Synopsys Nanosim, where the simulation resolution has been set to 1*ps*, thus producing, for each input-output pair, a trace of 5 000 data points. The target technological library used in the process is the Nangate 45 *nm* library. At the end of the simulation process, this dataset contains 256 simulated traces of execution, one for each S-box output and, being obtained from simulation, free from environmental or measurement noise.

4.2 Figures of Merit

We consider two standard metrics when conducting SCA: success rate and guessing entropy. Additionally, we consider one common machine learning metric: accuracy.

Most of the time, in side-channel analysis, an adversary is not only interested in predicting the labels $y(\cdot, k_a)$ in the attacking phase for which accuracy is a good metric but aims at revealing the secret key k_a. For this, common measures are the success rate (SR) and the guessing entropy (GE) of a side-channel attack [43]. In particular, let us assume, given Q amount of traces in the attacking phase, an attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability with $|\mathcal{K}|$ being the size of the keyspace. So, g_1 is the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate.

- **Guessing entropy**. It is the average position of k_a in $\mathbf{g}$.
- **Success rate**. It is defined as the average empirical probability that g_1 is equal to the secret key k_a. In this paper, we calculate the number of required trace to reach a success rate greater than 90%, which is denoted as $T_{SR>0.9}$. Aligned with the common implementations in the literature, guessing entropy and success rate are calculated by averaging the key rank results for 100 times.
- **Accuracy**. It is defined as the ratio of the correctly classified examples and the total number of examples.

4.3 Profiling methods – Classifiers

It is not a trivial task to select the best classifier for the given problem. Some classifiers can still be regarded as a usual choice when a highly accurate classification is sought [12]. To provide relevant experiments, we select several classifiers that are a common choice in SCA, as discussed in Section 2.

Template Attack The template attack (TA) relies on the Bayes theorem and considers the features as dependent [7]. In the state-of-the-art, template attack relies mostly on a normal distribution. Accordingly, a template attack assumes that each $P(X = x|Y = y)$ follows a (multivariate) Gaussian distribution parameterized by its mean and covariance matrix for each class Y. The authors of [9] propose to use only one pooled covariance matrix averaged over all classes Y to cope with statistical difficulties and thus lower efficiency. In our experiments, we use the pooled template attack.

Naive Bayes The Naive Bayes (NB) classifier is based on the Bayesian rule but is labeled "Naive" as it works under a simplifying assumption that the predictor features are mutually independent among the features, given the class value [13]. The existence of highly correlated features in a dataset can influence the learning process and reduce the number of successful predictions. NB assumes a normal distribution for predictor features and outputs posterior probabilities as a result of the classification procedure [13].

Radial Kernel Support Vector Machines Radial Kernel Support Vector Machines (denoted SVM in this paper) is a kernel-based machine learning family of methods that are used to classify both linearly separable and linearly inseparable data accurately. The idea for linearly inseparable data is to transform them into a higher dimensional space using a kernel function, wherein the data can usually be classified with higher accuracy. The scikit-learn implementation we use considers libsvm's C-SVC classifier that implements SMO-type algorithm [11]. The multi-class support is handled according to a one-vs-one scheme.

Random Forest Random Forest (RF) is a well-known ensemble decision tree learner [4]. Decision trees choose their splitting attributes from a random subset of k attributes at each internal node. The best split is taken among this randomly chosen attributes.

Multilayer Perceptron The multilayer perceptron (MLP) is a feed-forward neural network that maps sets of inputs onto sets of appropriate outputs. MLP consists of multiple layers (at least three) 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 [16].

Convolutional Neural Networks Convolutional neural networks (CNNs) commonly consist of three types of layers: convolutional layers, pooling layers, and fully connected layers. The convolution layer computes the output of neurons that are 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. The fully-connected layer (the same as in MLP) computes either the hidden activations or the class scores.

4.4 Points of Interest

For NB, SVM, RF, MLP, and TA, we use the 50 most important features, as commonly done in related works. To select those features, we use the Pearson correlation coefficient [22]:

$$Pearson(x;y) = \frac{\sum_{i=1}^{N}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}; \quad (10)$$

For the CNN classifier, we do *not* conduct any feature selection but instead use the full set of available features (raw traces or window of traces).

Hyperparameter Tuning We conduct a tuning phase to select hyperparameters for which classifiers perform well over considered datasets. We emphasize that the tuned parameters represent a reasonable choice that exhibits good behavior but should not be considered the best possible ones. A more detailed tuning could result in a somewhat improved performance if concentrating on any specific scenario. Nevertheless, since we consider scenarios where we introduce perturbations in the testing phase, there is no guarantee that good initial hyperparameters would still be suitable for the added noise traces. In fact, the hyperparameter tuning does not affect our framework: not-so-careful parameter tuning is considered as one of the noise settings considered in our framework (Figure 1). For TA and NB, there are no hyperparameters to tune. For SVM, there are two significant tuning parameters: the cost of the margin C and the kernel parameter . We conduct a grid search for $C = [0.001; 0.01; 0.1; 1]$ and $= [0.001; 0.01; 0.1; 1]$. In the end, we select $C = 1$ and $= 1$. For RF, we experiment with a different number of trees $I = [10; 50; 100; 200; 500; 1\,000]$ and select 200 traces for our architecture. For MLP with feature selection, we investigate *tanh* and *ReLU* activation functions, *sgd* and *adam* optimizers, and architectures of shape $(50; 25; 10; 50); (50; 30; 50); (50; 30; 20; 50)$. After the tuning phase, we decide to use the *tanh* activation function, *adam* optimizer, and $(50; 30; 20; 50)$ configuration of layers/nodes. For RF, MLP, and SVM, we use 5-fold cross-validation.

For CNN, we consider the optimized networks and hyperparameters from [38] for both ASCAD and CHES CTF datasets. Note that these networks are optimized for both HW and ID leakage models, and we directly apply them to our experiments. The detailed architectures are listed in Appendix B. For all tests, we use a validation dataset of 5 000 traces.

4.5 Framework Setting

We conduct our experiments for a scenario where $= 0.1$ and $= 0.1$. The Chernoff bound (Eq. (4)) gives $n \quad 149.7$ to achieve the desired level of accuracy and confidence. Consequently, we set $n = 150$ in our experiments, which means that every experiment is repeated 150 times to obtain statistically significant results. Next, we select to work with noise equal to 0.005 that goes in the range $[\quad if; \quad if]$, where if denotes the intensity factor in the range $[1; 30]$. By doing so, we will evaluate noise levels up to 15% of the signal, which would capture many realistic settings (more noise is, of course, possible). More precisely, we can consider our scenario as working with 30 different noise intensities (perturbations). Recall, the noise we inject could arise from various sources in practical applications, for example:

- variations between the profiling and attacking device [3],
- environmental noise between acquisition campaigns,
- (minor) changes in the experimental setup (probes, devices, hyperparameters, points of interest).
- effect of countermeasure randomness.

We emphasize that we do not claim that all countermeasures can be modeled with Gaussian noise. Our claim is that all the countermeasures can be modeled as noise (trivially, as they make the model more difficult to fit), then a robustness framework allows fair and insightful comparison. Gaussian noise was the model used for the settings we considered, but our framework can analyze any noise used to model countermeasures.

5 Framework Validation and Application

In this section, we first use our framework with the simulated traces to validate its correctness. Next, we use it to evaluate the profiling attacks' performance on the publicly available datasets.

Note that our framework considers the setting where we add noise to the attack traces only. This simulates the behavior one would encounter in, e.g., 1) portability due to differences between profiling device and the device under attack, 2) real-world applications due to the changes in the environment setup (e.g., changes in the probe location for EM acquisition [3], environment noise, etc.), 3) targets with countermeasures as they would cause differences between profiling and testing device, and 4) changes in the hyperparameter setting as suboptimal hyperparameter choices result in less powerful profiling models that cannot fit the leakage well. We do not add perturbation to both training and attack traces, as this would simulate different devices (when, e.g., testing the influence of a specific countermeasure, one needs to consider it both for the profiling and attack phases). Finally, we do not add perturbation to the training phase only as it would serve as a regularization factor that helps the classification procedure [23]. Additionally, we emphasize that this evaluation aims not to find the best performing methods but to confirm our framework's validity and obtain information about the robustness of profiling methods.

In Algorithm 1, we present the pseudocode of the procedure we follow to assess the robustness of a certain classifier against perturbations in the attack phase. Since the number of experiments (samples) equals 150 ($n = 150$) and the number of intensities equals 30 ($f = 30$), it means that for each scenario, we need to run the attack 4 500 times.

Remark 2. By following Algorithm 1, we observe we need to define appropriate , which denotes the performance (degradation) level, and perturbation $u(\ )$. Then, the function `verification-problem` returns 1 if the performance of the classifier due to the perturbation causes degradation no larger than . Finally, only if all n experiments return 1, we consider the classifier robust concerning a specific perturbation and performance level.

Example 2. Let us consider guessing entropy as a metric and performance level $= 5$. This means that guessing entropy can be up to value 5. Furthermore, we take the perturbation of 10%. Finally, let $= 0.1$ and $= 0.1$, which results in $n = 150$. This means we need to run Algorithm 1 150 times. If for all 150 times the function `verification-problem` returns 1, then we say that the classifier

Algorithm 1 Algorithm to solve the probabilistic robustness evaluation problem [1].

Identify the perturbation space and the random variable with pdf f over ;
Select the accuracy and confidence
Identify the interested performance level set $= \{ _1; \cdots ; _k\}$
$\hat{p}_{n;}$ ( )= verification-problem(; f ; u( ); ; ;)
use $\hat{p}_{n;}$ ( )

Function verification-problem (; f ; u( ); ; ;):
Draw $n \geq \frac{1}{2^2} \ln \frac{2}{}$ samples $_1; \cdots ; _n$ from according to f
For each $\in$ estimate

$$\hat{p}_n( ) = \frac{1}{n} \sum_{i=1}^{n} I\,(u(_i) \leq); \; I\,(u(_i) \leq) = \begin{cases} 1 \text{ if } u(_i) \leq \\ 0 \text{ if } u(_i) > \end{cases}$$

Return $\hat{p}_{n;}$

is 5 GE-robust for perturbation of 10%. If the function returns 0 in any of the experiments, then the classifier is not robust for that specific performance level and perturbation.

5.1 Framework Validation on Simulated Traces

In general, it is not an easy problem to validate a theoretical framework for machine learning as one needs to consider all possible scenarios. As this is impossible in practice, we first use the concept of stylized facts, which is a generalization that summarizes data. More precisely, we examine the framework's behavior with the simulation data and compare it with real data. If the simulation data behaves in the same manner as the real data, we can assume that the framework can indeed be used with real data. Naturally, this approach must handle certain inaccuracies as simulation data is far from a perfect representation of the real data. Here, we use the simulated traces where we add Gaussian noise, as discussed in the next section.

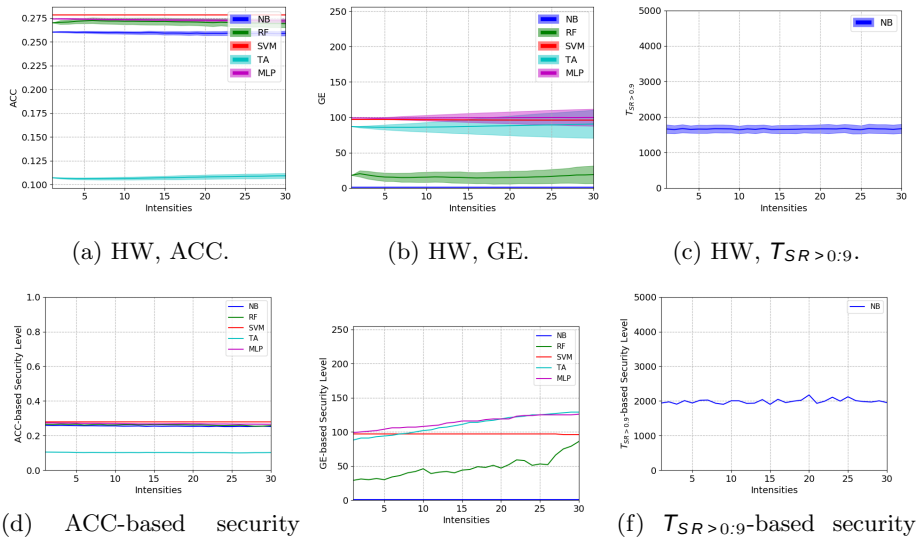
Simulated Traces - Data Augmentation and Noise Addition Recall, our simulated dataset contains 256 traces, one for each possible S-box output value and 5 000 features. We select 50 features that correspond to the highest correlation between the traces and the S-box output to match our other datasets. Next, we extend this dataset with additional traces to build a reliable profiling model. We do this by adding Gaussian noise to the simulation traces:

$$X^{0} = X + N: \tag{11}$$

We make the simplified but still commonly used assumption that the noise is univariate Gaussian distributed with zero-mean, i.e., $N \quad N\,(0; \ ^2)$ where is the standard deviation. We repeat this procedure 15 000 times to create 15 000

noisy traces with $\quad = (0; 0{:}5]$. Note that our final dataset contains approximately the same number of traces for each S-box output value. We divide the simulated and data augmented dataset into 10 000 traces for training and 5 000 for testing. We denote this dataset as AES_SIM.

From Figures 2a to 2c, we depict the behavior for the simulated dataset (AES_SIM) when adding perturbations. The solid line represents GE mean of 150 tests, while the faded area represents the corresponding variation. Since almost all the classifiers can break the target for the ID leakage model for all intensity levels, we only present the results for the HW leakage model as it carries more information. First, we notice that while the mean values of all three metrics are different, all classifiers behave the same in general. However, the variation of the metric increases when the intensity increases. This demonstrates that perturbation makes the attack more difficult, even when considering such a simple dataset and a small level of perturbation. Now, in accordance with Algorithm 1, we consider how many times an experiment returned value 1. The first performance value where all 150 experiments return 1 represents the achieved robustness level. We denote it as **{metric}-based security level**. The corresponding figures are presented in Figures 2d to 2f. For instance, the random forest classifier for GE and intensity 5 (corresponding to perturbation of 2.5%) would be GE 37-robust, as the worst value over 150 experiments equals 37. Thus, while GE values indicate good behavior (as in most of the experiments, GE reaches a low value indicating the target is broken), the robustness analysis shows that the algorithm is not robust for this noise level.

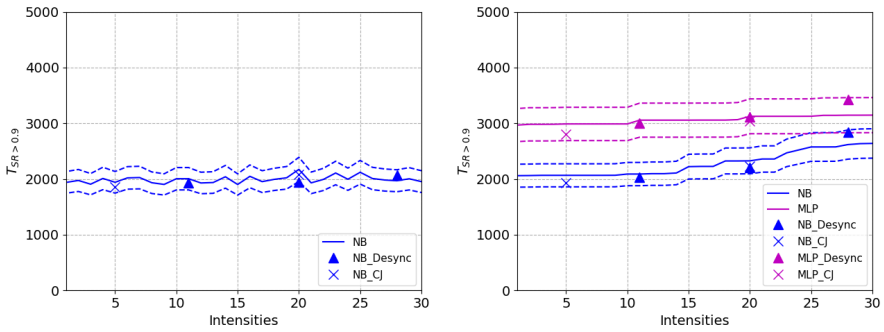


(a) HW, ACC. (b) HW, GE. (c) HW, $T_{SR>0:9}$.

(d) ACC-based security level. (e) GE-based security level. (f) $T_{SR>0:9}$-based security level.

Fig. 2: AES_SIM datasets with HW leakage model.

In the second experiment with the simulated dataset, we aim to confirm that perturbed traces represent a good approximation of various countermeasures. Consequently, we use the same 15 000 traces as before (256 simulated traces data augmented to 15 000 traces with Gaussian noise). On this dataset, we again consider 30 perturbation intensities (Gaussian noise from 0% to 15% of the signal). Additionally, we introduce hiding countermeasures to the dataset, where we consider clock jitter and desynchronization countermeasures [48]. Then, under the assumption that our framework can model the behavior of various countermeasures, we expect that the attack results for datasets with clock jitter or desynchronization countermeasures are similar to the simulated dataset with added hiding countermeasures (for instance, we can estimate signal-to-noise (SNR) ratio for both datasets, and expect the results to be similar for similar SNR values). As a result, we use traces with 1/2 point(s) of desynchronization or 1/2/3 of the clock jitters level. One can expect that with an increased countermeasure level, the SNR of the traces is reduced. We show some characteristic results in Figure 3. Acronym *_Desync* denotes desynchronization added to the traces and *_CJ* clock jitter added. Full lines depict the perturbation experiments, and dotted lines denote the 10% ranges. Crosses and triangles denote the result obtained for the same profiling model if, instead of perturbations, one considers the dataset with countermeasures. The results clearly show that traces with countermeasures are matched with the perturbed traces. This indicates that our framework can model various countermeasures, and consequently, it can be used to estimate the influence of adding countermeasures to a target. Indeed, by considering the results for various intensities, we can extrapolate whether a certain attack method is robust and the influence on the attack performance if adding a specific countermeasure to the target.



(a) HW. (b) ID.

Fig. 3: Results for perturbed traces and traces with added countermeasures.

These experiments con rm the validity of our framework.

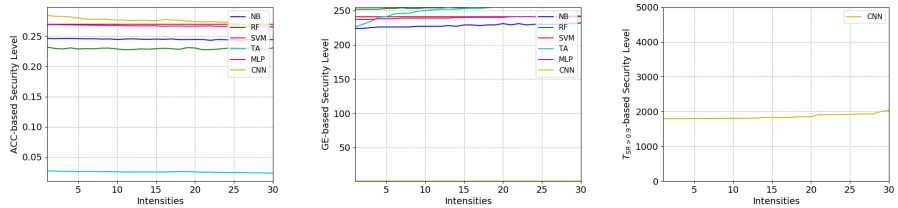
5.2 Publicly Available Datasets and Framework Evaluation

In this section, we present and briefly discuss several characteristic scenarios for publicly available datasets. In these experiments, we consider multiple intensities and performance levels. The figures show results for classifiers with and without feature selection. In Figure 4, we depict results attacking the ASCAD dataset with a fixed key. First, in terms of the ACC-based security level (Figures 4a and 4d). We observe that for both leakage models, the classifiers behave 1) "stable" in the sense that the corresponding figure of merit with the increase in the perturbation level slightly decreases, or 2) "resilient", as we observe no influence of perturbation to the attack performance. Note that CNN performs much better than simpler attacks and deteriorates marginally even for the highest intensity level. The reason accuracy with the ID leakage model is much lower is that we deal with the scenario with 256 labels, so these results are not surprising as the classification task is much more difficult.

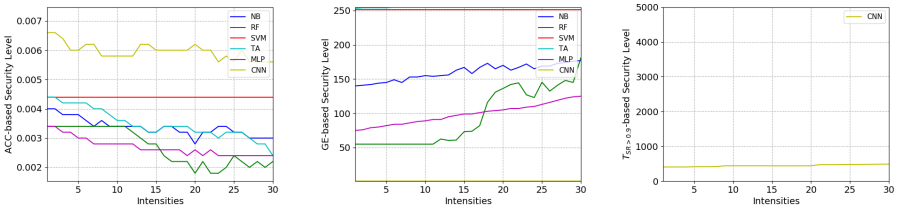
Next, in Figures 4b and 4e, we depict the results for the GE-based security level for both leakage model. When considering the HW leakage model (Figure 4b), the CNN classifier manages to break the target regardless of the perturbation level. Besides that, none of the classifiers can retrieve the key with the given number of attack traces. Meanwhile, we observe that some classifiers perform relatively stable but poorly (i.e., SVM). We can call this type of behavior "no learning" as the architecture is not sensitive to noise since it did not learn to fit the data. When using the ID leakage model, NB, RF, and MLP perform much better, where the influence of introduced perturbation can also be seen. As expected, CNN performs the best, while SVM and TA present "no learning" properties. For instance, for CNN (regardless of the leakage model), we obtain the result that the classifier is GE 0-robust for any perturbation level we considered. The best "stability" of results for CNN happens for GE, which is expected as GE uses the information from the whole key guessing vector.

Finally, in Figures 4c and 4f, we depict the results for the SR-based security level for both leakage model. Note that here the lower the value, the better is attack. As mentioned, only the CNN classifier can break the target regardless of the level of perturbation. Thus the SR-based security level for the rest of the classifiers is larger than 5 000. Therefore, only CNN results are shown here. Interestingly, although $T_{SR>0.9}$ slightly increases with the increasing level of perturbation, the variations are limited in a small range. This indicates the stable performance of the optimized model (and that the best guess is significantly better than other guesses, so perturbation does not cause significant changes). This type of model resilience would be helpful when attacking the traces with variation (i.e., attacking the traces acquired from a different device).

After testing ASCAD with a fixed key, we attack the ASCAD with randoms keys dataset. The results are presented in Figure 5. Similar to the observation from Figure 4, we see that the ACC-based security level decreases with more noise being added. Still, there are classifiers, such as TA and SVM, that perform stable regardless of intensity ("no learning"). In terms of GE-based security level, besides SVM ("no learning") and CNN ("Resilient"), the rest of the clas-



(a) HW, ACC-based security level. (b) HW, GE-based security level. (c) HW, $T_{SR>0.9}$-based security level.



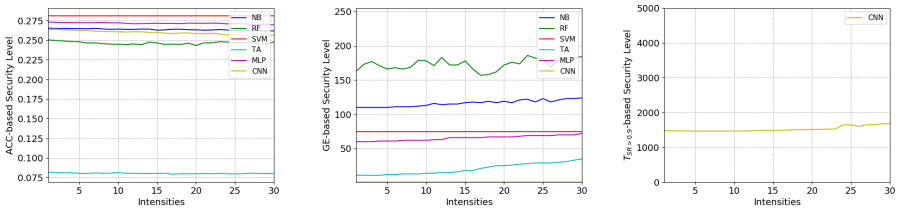
(d) ID, ACC-based security level. (e) ID, GE-based security level. (f) ID, $T_{SR>0.9}$-based security level.

Fig. 4: Results attacking ASCAD (fixed key) dataset.

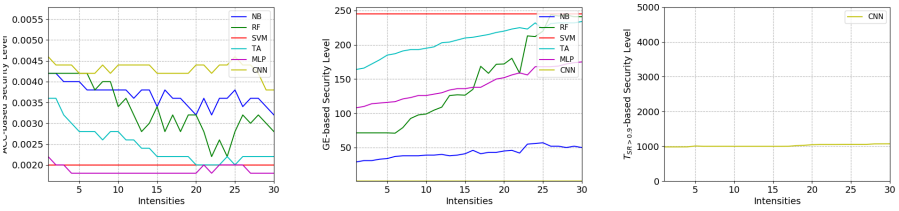
sifiers follow "unstable" behavior. Here, by "unstable" behavior, we mean large oscillations for different perturbation levels. This behavior happens either due to overfitting or underfitting of the model. Finally, $T_{SR>0.9}$-based security level clearly shows the stable performance of the CNN classifier, which again confirms the superior of such an optimized model in fighting with noise addition. In general, we observe that the dataset with random keys is less robust than a dataset with the fixed key. Additionally, since the dataset is more difficult to attack, there is a clear benefit from using a more powerful classifier (CNN) that also uses more features (1 400 vs. 50).

Finally, we test the CHES_CTF dataset. The results are presented in Figure 6. Note that this dataset has limited ID leakage. Thus only the HW leakage model is tested. In terms of the ACC-based security level, we see a significant drop for the TA classifier, indicating its 'inability to deal with large levels of noise, which is also confirmed when looking at the GE-based security level. In terms of GE-based security level, we again see that besides SVM ("no learning") and CNN ("Resilient"), the rest of the classifiers follow a combination of "stable" and "unstable" behavior. For $T_{SR>0.9}$-based security level, CNN shows stable performance, but it requires significantly more traces to reach the same success rate level. This observation is aligned with the observation from the previous datasets.

Based on the obtained results, we see that CNNs (deep learning) perform the best, which adds to them the third advantage compared to simpler machine learning techniques. Indeed, deep learning has an advantage over simpler



(a) HW, ACC-based security level. (b) HW, GE-based security level. (c) HW, $T_{SR>0.9}$-based security level.



(d) ID, ACC-based security level. (e) ID, GE-based security level. (f) ID, $T_{SR>0.9}$-based security level.

Fig. 5: Results attacking ASCAD (random keys) dataset.

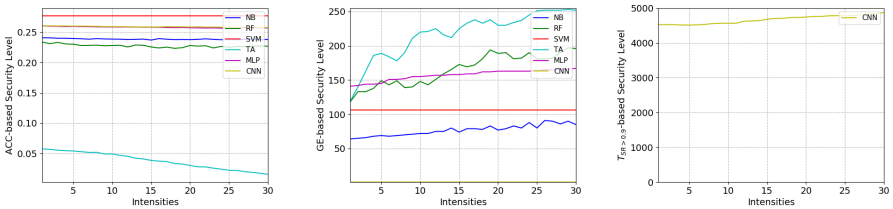
machine learning techniques as it does not require feature engineering and can break targets protected with countermeasures. Now, we additionally observe that it provides more robustness than simpler machine learning techniques. It would be interesting to evaluate the robustness behavior for portability cases, as there, deep learning is known to easily overfit (granted, on simple targets) [3].

These experiments show that our framework can help analyzing the pro ling side-channel attacks in a new way and obtaining more information than commonly used analysis techniques.

6 General Observations and Remarks

There are four main types of behavior one can expect when modeling the system with perturbations. We give the list in the order from the most preferred to the least preferred setting (where preference is aligned with the attacker perspective):

1. "*Resilient*" behavior. In this setting, the classifier is resilient to perturbations, which means that its performance remains practically unchanged even in the presence of noise. Naturally, after some point, the performance starts to deteriorate, and then, the behavior resembles one of the following types.
2. "*Stable*" behavior. In this setting, the classifier is affected by perturbation, and the performance is gradually decreased.



(a) HW, ACC-based security level. (b) HW, GE-based security level. (c) HW, $T_{SR>0.9}$-based security level.

Fig. 6: Results attacking CHES_CTF dataset.

3. "*Unstable*" behavior. Here, a small perturbation results in a significant performance drop or very erratic behavior, i.e., a jump between good and poor performance.
4. "*No learning*" behavior. With this behavior, the perturbation does not influence the performance of the profiling method. This happens because the classifier, even before perturbation, was not working (e.g., classifying all measurements into the Hamming weight 4 [34]), and naturally, making the problem more difficult cannot improve the behavior but also cannot deteriorate it. While this behavior has certain similarities with the first one, the underlying idea is different. Here, the classifier never works, while in the "Resilient" setting, it is stable up to a certain level of perturbation (i.e., no difference between that level of perturbation and no perturbation).

Besides these four types, several subtypes can be recognized by combining the basic behaviors' traits.

The main advantages of using the proposed framework are:

- The framework allows an objective and reliable way to estimate the performance of the profiling side-channel analysis.
- The framework supports considering robustness analysis in different ways. Indeed, here we opted to analyze the classifiers' robustness to the addition of noise in the attack phase. This approach fits different scenarios, from portability, changes in the dataset size, countermeasures, hyperparameter changes [8], classifiers change, etc. Still, we could also evaluate the robustness concerning some other perturbation like the addition or removal of features from traces or pre-processing approaches [48].
- With our framework, it is possible to "map" different setting: for instance, 1) to see what level of perturbation causes the classifiers to behave in the same way, or, 2) at what point the suboptimal hyperparameter results in the same behavior as an added perturbation.

[8] Depending on the scenario, it can be convenient to consider the change in the hyperparameter as a different classifier or as a perturbed classifier.

Ultimately, our framework allows us to directly apply to the profiling side-channel analysis problem the whole body of knowledge developed in years of studying the robustness problem. Based on these advantages, we immediately see one very practical application of our framework: to help designers/manufacturers to make more secure systems. With it, we can observe the added benefit of a certain countermeasure and if it is sufficient to prevent a certain type of attack. Thus, by using this framework, we can design systems that are more secure against various types of attacks.

Our framework is based on robustness analysis. We inherit the advantages of such an approach, but also the limitations. We list two of them, discussing their direct implication to the profiling side-channel attacks problem:

- One needs to know what to evaluate, i.e., what perturbation to take into account.
- Observing the robustness does not necessarily tell us how to make the classifier more robust.

Note that those limitations are not connected with the SCA domain but rather the robustness analysis. Indeed, one needs to know what to evaluate and why that would make sense. Finally, to make the classifier more robust, one would need to conduct perturbations in the sense of the hyperparameter changes.

In this paper, we modeled the perturbations as Gaussian noise added to the attack traces only. This allowed us to reach the following general conclusions based on our experimental results:

- There are four main types of behaviors we can observe with classifiers.
- Easier datasets or leakage models with fewer labels are, in general, more resilient to perturbations.
- Deep learning is, in general, more resilient to perturbations.
- Different types of countermeasures can have the same effect on the classifier, and different types of countermeasures can be approximated with Gaussian noise.

7 Conclusions

In this paper, we concentrate on profiling side-channel attacks and uncertainties stemming from the experimental results. To that end, we propose a general framework that can be used to analyze the behavior of any profiling side-channel attack (i.e., a method performing the classification task). Our framework supports datasets with any characteristics as well as different leakage models. To offer such a general behavior, we model it as the expectation estimation problem to achieve any desired accuracy and confidence level. After we model the classifiers, we use the robustness analysis to estimate their performance in the presence of perturbations. Such an analysis allows us to answer which classifier is the most robust for a specific scenario. We give robustness analysis for several figures of merit: accuracy, success rate, guessing entropy, and the number of traces to reach a specific attack performance.

We believe our framework will be a powerful tool that will allow researchers to compare various classifiers' behavior more fairly than it is done up to now. Since profiling SCA in realistic settings should consider different profiling and attacking devices, we expect that our framework will allow more than just connecting SCA with problems that have reliable theoretical results. More precisely, our framework allows modeling the realistic behavior of profiling SCA, where uncertainty must occur due to several different noise sources.

We note that our framework can be demanding: to conduct a proper analysis, the Chernoff bound requires a large number of experiments, which potentially can be a prohibiting factor for specific computationally intensive classifiers or very large datasets. One easy way how to circumvent this is to parallelize the process for different perturbation levels. Another interesting direction would be to consider different sources of perturbations, e.g., not only adding removing noise but also adding/removing features from a dataset.

References

1. Alippi, C. *Intelligence for Embedded Systems: A Methodological Approach*. Springer Publishing Company, Incorporated, 2014.
2. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., and Dumas, C. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng. 10*, 2 (2020), 163–188.
3. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., and Shrivastwa, R. R. 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* (2020), The Internet Society.
4. Breiman, L. Random Forests. *Machine Learning 45*, 1 (2001), 5–32.
5. Bronchain, O., Hendrickx, J. M., Massart, C., Olshevsky, A., and Standaert, F.-X. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In *Annual International Cryptology Conference* (2019), Springer, pp. 713–737.
6. Cagli, E., Dumas, C., and Prouff, E. 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* (2017), pp. 45–68.
7. Chari, S., Rao, J. R., and Rohatgi, P. Template Attacks. In *CHES* (August 2002), vol. 2523 of *LNCS*, Springer, pp. 13–28. San Francisco Bay (Redwood City), USA.
8. Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist. 23*, 4 (12 1952), 493–507.
9. Choudary, O., and Kuhn, M. G. Efficient template attacks. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers* (2013), A. Francillon and P. Rohatgi, Eds., vol. 8419 of *LNCS*, Springer, pp. 253–270.
10. Durvaux, F., Standaert, F.-X., and Veyrat-Charvillon, N. How to certify the leakage of a chip? In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2014), Springer, pp. 459–476.

11. Fan, R.-E., Chen, P.-H., and Lin, C.-J. Working Set Selection Using Second Order Information for Training Support Vector Machines. *J. Mach. Learn. Res. 6* (Dec. 2005), 1889–1918.
12. Fernandez-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research 15* (2014), 3133–3181.
13. Friedman, N., Geiger, D., and Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning 29*, 2 (1997), 131–163.
14. Genkin, D., Shamir, A., and Tromer, E. Acoustic cryptanalysis. *Journal of Cryptology 30*, 2 (Apr 2017), 392–443.
15. Gilmore, R., Hanley, N., and O'Neill, M. Neural network based attack on a masked implementation of AES. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (May 2015), pp. 106–111.
16. Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
17. Heuser, A., Picek, S., Guilley, S., and Mentens, N. Lightweight ciphers and their side-channel resilience. *IEEE Transactions on Computers PP*, 99 (2017), 1–1.
18. Heuser, A., Rioul, O., and Guilley, S. Good is Not Good Enough — Deriving Optimal Distinguishers from Communication Theory. In *CHES* (2014), L. Batina and M. Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*, Springer.
19. Heuser, A., and Zohner, M. Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines. In *COSADE* (2012), W. Schindler and S. A. Huss, Eds., vol. 7275 of *LNCS*, Springer, pp. 249–264.
20. Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*, 301 (1963), 13–30.
21. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., and Vandewalle, J. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering 1* (2011), 293–302. 10.1007/s13389-011-0023-x.
22. James, G., Witten, D., Hastie, T., and Tibsihrani, R. *An Introduction to Statistical Learning*. Springer Texts in Statistics. Springer New York Heidelbert Dordrecht London, 2001.
23. Kim, J., Picek, S., Heuser, A., Bhasin, S., and Hanjalic, A. 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.
24. Kocher, P. C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of CRYPTO'96* (1996), vol. 1109 of *LNCS*, Springer-Verlag, pp. 104–113.
25. Kocher, P. C., Jaffe, J., and Jun, B. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1999), CRYPTO '99, Springer-Verlag, pp. 388–397.
26. Lerman, L., Bontempi, G., and Markowitch, O. Power analysis attack: An approach based on machine learning. *Int. J. Appl. Cryptol. 3*, 2 (June 2014), 97–115.
27. Lerman, L., Bontempi, G., and Markowitch, O. A machine learning approach against a masked AES - Reaching the limit of side-channel attacks with a learning model. *J. Cryptographic Engineering 5*, 2 (2015), 123–139.
28. Lerman, L., Medeiros, S. F., Bontempi, G., and Markowitch, O. A Machine Learning Approach Against a Masked AES. In *CARDIS* (November 2013), Lecture Notes in Computer Science, Springer. Berlin, Germany.

29. Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., and Standaert, F. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers* (2015), S. Mangard and A. Y. Poschmann, Eds., vol. 9064 of *Lecture Notes in Computer Science*, Springer, pp. 20–33.
30. Maghrebi, H., Portigliatti, T., and Prouff, E. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings* (2016), pp. 3–26.
31. Mangard, S., Oswald, E., and Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, http://www.dpabook.org/.
32. Masure, L., Dumas, C., and Prouff, E. 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* (2019), I. Polian and M. Stöttinger, Eds., vol. 11421 of *Lecture Notes in Computer Science*, Springer, pp. 145–167.
33. Picek, S., Heuser, A., and Guilley, S. Template attack versus Bayes classifier. *Journal of Cryptographic Engineering 7*, 4 (Nov 2017), 343–351.
34. Picek, S., Heuser, A., Jovic, A., Bhasin, S., and Regazzoni, F. 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.
35. Picek, S., Heuser, A., Jovic, A., Ludwig, S. A., Guilley, S., Jakobovic, D., and Mentens, N. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017* (2017), pp. 4095–4102.
36. Quisquater, J.-J., and Samyde, D. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security* (Berlin, Heidelberg, 2001), I. Attali and T. Jensen, Eds., Springer Berlin Heidelberg, pp. 200–210.
37. Regazzoni, F., Cevrero, A., Standaert, F.-X., Badel, S., Kluter, T., Brisk, P., Leblebici, Y., and Ienne, P. A design flow and evaluation framework for DPA-resistant instruction set extensions. In *CHES09*, C. Clavier and K. Gaj, Eds., vol. 5747 of *LNCS*. Springer, Sept. 2009, pp. 205–19.
38. Rijsdijk, J., Wu, L., Perin, G., and Picek, S. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2021/071, 2021. https://eprint.iacr.org/2021/071.
39. Rudin, W. *Real and Complex Analysis, 3rd Ed.* McGraw-Hill, Inc., New York, NY, USA, 1987.
40. Schindler, W., Lemke, K., and Paar, C. A Stochastic Model for Differential Side Channel Cryptanalysis. In *CHES* (Sept 2005), LNCS, Ed., vol. 3659 of *LNCS*, Springer, pp. 30–46. Edinburgh, Scotland, UK.
41. Smith, L. N. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (mar 2017), IEEE, pp. 464–472.
42. Standaert, F.-X., Koeune, F., and Schindler, W. How to compare profiled side-channel attacks? In *International Conference on Applied Cryptography and Network Security* (2009), Springer, pp. 485–498.

43. Standaert, F.-X., Malkin, T. G., and Yung, M. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009* (Berlin, Heidelberg, 2009), A. Joux, Ed., Springer Berlin Heidelberg, pp. 443–461.
44. Valiant, L. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, Inc., New York, NY, USA, 2013.
45. van der Valk, D., and Picek, S. Bias-variance decomposition in machine learning-based side-channel analysis. Cryptology ePrint Archive, Report 2019/570, 2019. https://eprint.iacr.org/2019/570.
46. van der Valk, D., Picek, S., and Bhasin, S. Kilroy was here: The first step towards explainability of neural networks in profiled side-channel analysis. Cryptology ePrint Archive, Report 2019/1477, 2019. https://eprint.iacr.org/2019/1477.
47. Whitnall, C., and Oswald, E. Robust profiling for dpa-style attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2015), Springer, pp. 3–21.
48. Wu, L., and Picek, S. 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.
49. Zaid, G., Bossuet, L., Habrard, A., and Venelli, A. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2020*, 1 (Nov. 2019), 1–36.

A Lebesgue Measurability

The first requirement that we have to fulfill is that the expected value we want to compute exists. In other words, we require that the integrals we need to compute are well defined, namely that the functions we need to compute are measurable. The most common type of integrals is the integrals of Lebesgue, which are well defined on functions that are Lebesgue measurable.

A generic function $u(\ )$; 2 R^l is Lebesgue measurable with respect to when its generic step-function approximation S_N obtained by partitioning in N *arbitrary* domains grants that

$$\lim_{N\ !\ 1} S_N = u(\ )$$

on set , R^l being a null measure set [39].

The assumptions on Lebesgue measurably are extremely soft. In essence, no engineering-related mathematical computations are Lebesgue non-measurable: every time we integrate, we implicitly assume the function is Lebesgue measurable. The functions we consider in this paper are Lebesgue measurable, so the integrals we compute in Section 3 are well-defined.

B CNN Hyperparameters and Architectures

In Table 1, we present the hyperparameters of the tested CNN architectures from [38].

Table 1: Common hyperparameters for all the reported best architectures.

Convolutional Padding Type	SAME
Pooling Type	Average Pooling
SoftMax Initializer	Glorot Uniform
Initializer for other layers	He Uniform
Activation function	SeLU
Optimizer	Adam
Train Epochs	50
Learning Rate	One Cycle Policy [41]

We present the best obtained architectures in Tables 2. Note that all the tables use the following notation:
Convolutional = C(*Iters*, *kernel_size*, *strides*)
Batch Normalization = BN
Average Pooling = P(*size*, *stride*)
Flatten = FLAT
Fully-connected = FC(*size*)
SoftMax = SM(*classes*)

Table 2: Common hyperparameters for all the reported best architectures.

ASCAD (fixed key) HW	C(2,25,1), P(4,4), FLAT, FC(15), FC(10), FC(4), SM(9)
ASCAD (fixed key) ID	C(128,25,1), P(25,25), FLAT, FC(20), FC(15), SM(256)
ASCAD (random keys) HW	C(4,50,1), P(25,25), FLAT, FC(30), FC(30), FC(30), SM(9)
ASCAD (random keys) ID	C(128,3,1), P(75,75), FLAT, FC(30), FC(2), SM(256)
CHES CTF HW	C(2,2,1), P(7,7), FLAT, FC(10), SM(9)