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

Stjepan Picek, Annelie Heuser, Cesare Alippi, Francesco Regazzoni

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

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

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

Abstract. Profiled side-channel attacks are the most powerful attacks and they consist of two steps. The adversary first builds a leakage model, using a device similar to the target one, then it exploits this leakage model 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 (corresponding to the secret key) the traces collected from the victim's devices belong to. For a number of years, the research community studied profiled attacks and proposed numerous improvements. Despite a large number of empirical works, a framework with strong theoretical foundations to address profiled side-channel attacks is still missing.

In this paper, we propose a framework capable of modeling and evaluating all profiled analysis attacks. This framework is based on the expectation estimation problem that has strong theoretical foundations. Next, we quantify the effects of perturbations injected at different points in our framework through robustness analysis where the perturbations represent sources of uncertainty associated with measurements, non-optimal classifiers, and methods. Finally, we experimentally validate our framework using publicly available traces, different classifiers, and performance metrics.

1 Introduction

Embedded and cyber-physical devices, connected together to form the Internet of Things (IoT), are pervading every aspect of our lives. They make our lives simpler, but their use for handling sensitive data and for managing critical infrastructure poses new challenges. Because of this, security becomes one of the most important extra-functional requirements that designer should provide to the units. Designing secure embedded devices is extremely challenging because of two main reasons. First, the limited area and energy budget available in these devices are often not sufficient to implement full flagged and robust cryptographic primitives. Second, these devices should be resistant against physical attacks. In fact, the pervasive diffusion of these devices makes them physically available to adversaries willing to exploit the physical weaknesses of the implementation to extract the stored secret information (typically, the secret key).

To achieve resistance against physical attacks, it is necessary to have a complete understanding of the adversary's capabilities. Side-channel attacks (SCAs), in particular profiled 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. In the rest of this paper, when we talk about side-channel attacks, i.e. profiled side-channel attacks, we consider those attacks that use power or electromagnetic radiation as side-channel. In profiled attacks, the adversary first profiles the power consumption of a device identical (or, at least similar) to the one that will be attacked. In the second phase, using this profile and the power traces measured from the victim device, the adversary attempts to recover the target secret key. Well-known examples of such profiled attacks are template attack [1, 2] as well as supervised machine learning-based attacks [3–6].

While it is well-known that template attack is the most powerful one from the information theoretic point of view (given that certain assumptions hold) [2], for machine learning techniques the full characterization of their performance is open. From one perspective, a large number of works experimentally show how machine learning techniques perform extremely well in many realistic scenarios and even surpass template attack [4]. At the same time, there is no theoretic guarantee how machine learning techniques would behave is some different scenario, even if it is a similar one. As such, the results we can obtain are always somewhat "lacking" and need to be taken *cum grano salis*. Naturally, giving general theoretic findings is very difficult, even impossible. For instance, the "No Free Lunch" theorem, which states that when averaged over all classification problems, all supervised machine learning behave the same [7] gives us an insight about the difficulty of the problem. Still, one could hope to give answers to some more specific questions of the form (in the increasing order of difficulty):

1. What machine learning algorithm is the best one for a given side-channel scenario?
2. What machine learning algorithm is the best one for multiple side-channel scenarios?
3. What machine learning algorithm is the best one for side-channel scenarios?

As already said, the current state-of-the-art does not offer us any results giving theoretical insights into the performance of machine learning-based attacks nor their robustness. In this paper, we aim to fill in this gap and offer the first results considering the general performance of machine learning attacks (and in fact, all profiled attacks). To that end, we propose a general framework intended to analyze the behavior of profiled SCAs and give insights into the robustness of such algorithms. Here, by robustness, we understand the ability of a system to tolerate perturbations (random changes inserted into the system). Although we use the robustness analysis as a tool to obtain insights into the general behavior of profiled attacks, we note perturbations are also an integral part of such attacks. In fact, there are several sources of perturbations that occur in profiled SCA due to:

– Environment noise,

– Countermeasures,
– The differences between the profiling device and the device under attack.

While the first source of perturbation is clear, the other two deserve further explanation. Consider a system with a countermeasure. There, the profiling attack method does not "know" what is the countermeasure but can only "see" its consequence in the form of noise, i.e., perturbation causing the problems in system's functionality. For this setting, we only require that the generated perturbation has a Gaussian distribution (which does not present a limitation in practical settings). Finally, in a realistic setting, the measurements for profiled attacks we obtain from two devices: profiling device and the device under attack, Yet, usually we omit this fact in the experiments and then all the measurements come from the same device. This simplification is a reasonable one but still bound to introduce some error where the real behavior of profiled attack will be somewhat lower than the one measured in experiments with only one device. Additionally, the difference in the measurements in practice can also arise for instance from different probes positions when we use electromagnetic (EM) SCA.

By considering the robustness problem, we actually consider 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 [8] where PAC learning theory formalizes the way a computation is carried out within an uncertainty affected environment.

The main contributions of this paper are:

– We propose a framework capable of modeling and evaluating all profiled side-channel attacks where we provide theoretical foundations for the framework.
– We consider the robustness of profiled attacks in 1) the presence of countermeasures and 2) when there are two devices: one for profiling and second to be attacked.
– We consider the robustness of profiled attacks for all commonly used figures of merit: accuracy, success rate, and guessing entropy.

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 [9]. In other words, implementations of cryptographic algorithms leak relevant information about the data processed through physical side-channels like timing [9], power consumption [10], EM emanation [11], or sound [12]. Today, when considering SCA and symmetric-key cryptography, there are two main directions one could follow:

1. direct attack, see e.g., Simple Power Analysis (SPA) and Differential Power Analysis (DPA).
2. profiled attacks, see e.g., Template Attack (TA) or a number of machine learning-based techniques.

Differing from the direct attacks, profiled attacks require a profiling stage, i.e., a step during which the cryptographic hardware is under a full control of the adversary to estimate the probability distribution of the leaked information. Such attacks have received a lot of attention in the last years due to the fact they define the worst case security assumptions. There, template attack is de-facto standard in the SCA community. TA is the best (optimal) technique from an information theoretic point of view if the attacker has an unbounded number of traces and the noise follows the Gaussian distribution [13,14]. After the template attack, the stochastic attack emerged using linear regression in the profiling phase [15]. 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 only one pooled covariance matrix is used in order to cope with statistical difficulties [1].

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, some researchers started experimenting with different machine learning (ML) techniques and evaluating their effectiveness in the SCA context. There, a survey of available works reveals a number of papers discussing different machine learning algorithms, targets, experimental settings, etc. When considering attacks on AES (as we do in this paper) and machine learning techniques, one can find a number of papers, see e.g., [3, 5, 14, 16–22]. More recently, deep learning techniques started to capture the attention of the SCA community. Accordingly, the first results confirmed that expectations where most of the attention went to convolutional neural networks [4, 23].

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 performance and often outperform template attack. Additionally, all of these works have "only" experimental results with some attempts (ranging from obvious ones to purely speculative ones) in explaining why such results are obtained. At the same time, as far as we are aware, there are no papers discussing on a more general level what are the limitations when using machine learning.

3 General Framework

In this section, we present a generic approach for modeling supervised machine learning algorithms and we show how, from this, we derive a framework to model all profiled side-channel attacks. First, we introduce the framework in an intuitive way, showing the motivation and goals. Then, we introduce the background definitions and the numerical problems on which our framework is based. Finally, we formally present the framework for modeling profiled side-channel attacks and show how this is tightly connected with the PAC learning concept.

3.1 Intuitive Description of the Framework

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

Figure 1 depicts the generic framework we propose in order to model profiled 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 $x_i, 1 \leq i \leq n$ where n is the number of measurements taken corresponding to the encryption of a certain number of plaintexts PT and a number of known keys K. For each measurement x_i, we obtain a 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 measurements x_i we additionally obtain the label of the measurement y_i, which denotes the actual value the measurement has. The leaked traces are typically altered by a perturbation δ_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 models. In order to model this, we consider the output of the given classifier influenced by a certain perturbation δ_2. Without loss of generality, we consider δ_1 and δ_2 to be equal to δ 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 robustness of the classifier.

The problem we consider here can be connected to the one of analyzing the robustness of randomized algorithms [24]. In the randomized algorithm setting, we are interested in quantifying the robustness of an algorithm by means of a suitable figure of merit. In our setting, we aim to quantify the robustness of a profiled 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 algorithm and model as well as any figure of merit. To validate our framework, in Section 4, we select to work with a number of common SCA classifiers and figures of merit.

Our framework proposes one theoretical interpretation for an analysis which, to date, was mostly done empirically. Consequently, we are able to achieve two goals:

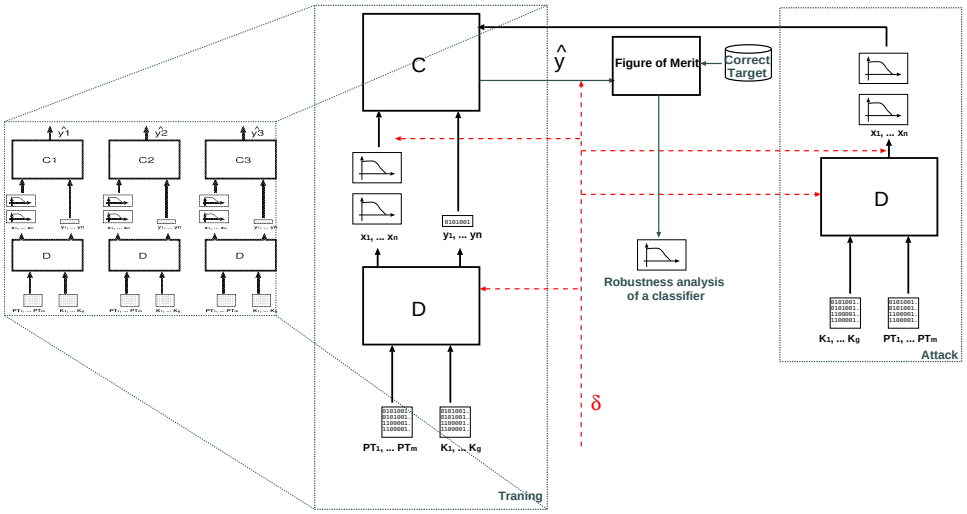


Fig. 1: The framework for the profiled side-channel analysis. The red line denotes the possible perturbations. There are two phases of the attack. Profiling phase (left part of the figure in bold) and attack phase (right part of the figure in bold). Since the framework is generic, it supports any type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

1. The connection with well-understood problems (expectation estimation problem and robustness problem) allows us to model each profiled side-channel attack and thus compare, in a sound 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 in a sound way to questions such as "Which classifier behaves better in some specific scenario?" and "How to compare different profiled side-channel attacks?".

3.2 Expectation Estimation Problem and Robustness

In this section, we aim to answer two questions which will be used in subsequent derivations and provide some common nomenclature. First, how to estimate the expected value of a function, i.e., the minimum number of samples granting to build an approximation with an arbitrary level of accuracy ϵ and confidence δ (Section 3.2). Second, how to estimate the robustness of a system once affected by perturbations (Section 3.2).

Lebesgue Measurability A generic function $u(\psi), \psi \in \Psi \subseteq \mathbb{R}^l$ is Lebesgue measurable with respect to Ψ when its generic step-function approximation S_N obtained by partitioning Ψ in N *arbitrary* domains grants that

$$\lim_{N\to\infty} S_N = u(\psi)$$

on set $\Psi - \Omega$, $\Omega \subseteq \mathbb{R}^l$ being a null measure set [25]. Basically, no engineering-related mathematical computations are Lebesgue non-measurable.

The Chernoff Bound 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 be used also to estimate the expected value of random variable according to estimation accuracy and confidence levels set by the designer [26]. The Chernoff bound for a generic probability density function and continuous variable ψ can be derived from the Hoeffding inequality for the empirical mean [27].

Let $x_1, \cdots x_n$ 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., $\Pr(x_i \in [a_i, b_i]) = 1$. Then, defining the empirical mean $\hat{E}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$, we have that for any ϵ value the Hoeffding inequality for the empirical mean:

$$\Pr\left(|\hat{E}_n - E[\hat{E}_n]| \geq \epsilon\right) \leq 2e^{\frac{-2\epsilon^2 n^2}{\sum_{i=1}^{n}(b_i-a_i)^2}} \tag{1}$$

holds where E is the expectation operator. Eq. (1) can be rewritten as:

$$\Pr\left(|\hat{E}_n - E[\hat{E}_n]| < \epsilon\right) > 1 - 2e^{\frac{-2\epsilon^2 n^2}{\sum_{i=1}^{n}(b_i-a_i)^2}}. \tag{2}$$

If $\hat{E}_n$ is the estimate $\hat{p}_n(\gamma)$ of a probability e.g., ($p(\gamma) = \Pr(u(\psi) \leq \gamma)$ for a given positive scalar γ and a loss function $u(\psi)$), we have that for a generic random variable ψ_i the indicator function

$$x_i = I\left(u(\psi_i) \leq \gamma\right) = \begin{cases} 1 \text{ if } u(\psi_i) \leq \gamma \\ 0 \text{ if } u(\psi_i) > \gamma \end{cases}$$

assumes values in $\{0, 1\}$. As a consequence, $a_i = 0, b_i = 1$ and Eq. (2) becomes

$$\Pr\left(|\hat{E}_n - E[\hat{E}_n]| < \epsilon\right) > 1 - 2e^{-2n\epsilon^2}.$$

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

$$\Pr\left(|\hat{p}_n(\gamma) - p(\gamma)| < \epsilon\right) > 1 - 2e^{-2n\epsilon^2}. \tag{3}$$

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

$$n \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}. \tag{4}$$

The Chernoff bound states that if we sample from the domain of random variable ψ according to its probability density function then the (3) holds with confidence δ. In other words, we can build an approximation $\hat{p}_n$ of unknown probability $p(\gamma)$ with accuracy ϵ; the statement holds with confidence δ.

The Expectation Estimation Problem The expectation estimation problem consists in 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 $u(\psi)$.

Let $u(\psi) \in [0, 1]$ be a Lebesgue measurable function over $\Psi \subseteq \mathbb{R}^l$ and f_ψ be the probability density function of a random variable ψ defined over Ψ. The expectation estimation requires evaluation of the expected mean

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

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 $\psi_1, \cdots, \psi_i, \cdots, \psi_n$ drawn from ψ according to f_ψ. We call

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

the empirical mean. It should be commented that $\hat{E}_n(u(\psi))$ 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 ϵ and confidence δ.

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

$$n \geq \frac{(b-a)^2}{2\epsilon^2} \ln \frac{2}{\delta}. \tag{7}$$

Robustness Robustness of a system refers to the ability to tolerate perturbations that might affect its structural parameters and, in turn, its performance, measured by means of a given figure of merit.

More formally, system/function $g(\theta, x)$ is robust with respect to perturbations $\delta\theta \in \Delta \in R^l$ at level $\gamma \in \mathbb{R}^+$ when, given a discrepancy function $u\left(g(\theta, x), g(\theta, \delta\theta, x)\right) \in \mathbb{U} \subset \mathbb{R}$ the system experiences a degradation in performance within γ:

$$u(\delta\theta) = u\left(g(\theta, x), g(\theta, \delta\theta, x)\right) \leq \gamma, \quad \forall \delta\theta \in \Delta, \forall x \in \tilde{X}. \tag{8}$$

In the rest of the paper, we assume that the level of perturbations can become arbitrarily large so that no small perturbation theories are viable.. $u(\delta\theta)$ represents the perturbation impact on the behavior of the system as observed through the figure of merit. Note, $u(\delta\theta)$ 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 γ satisfying the previous expression. Since this can be computationally intractable, we move to a probabilistic setting. There, a computation is robust at level γ with probability $1-\eta$ for the perturbation space Δ when γ is the smallest value such that $\Pr(u(\delta\theta) \leq \gamma) \geq 1-\eta,\ \forall \delta\theta \in \Delta$. Here, η is a small positive value in $[0,1]$ and $1-\eta$ is the confidence level.

Once defined $p(\gamma)$ to be the probability that $u(\delta\theta) \leq \gamma$ for an arbitrary but given γ value:

$$p(\gamma) = \Pr(u(\delta\theta) \leq \gamma)\ for\ each \delta\theta \in \Delta. \tag{9}$$

Eq. (9) can be estimated with Chernoff and the minimum γ identified through set $\Gamma = \{\gamma_1, \cdots, \gamma_k\}$).

3.3 The Profiled SCA Framework

We now express profiled side-channel attacks as the expectation estimation problem. The starting point is to map the steps of profiled analysis attacks to the framework depicted in Figure 1. The first phase of profiled 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 q amount of measurement traces $X_q = [X_1, X_2, \ldots, X_q]$. 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 $t \in \mathcal{T}$ and the key $k^* \in \mathcal{K}$ 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 profiled analysis, there exist two main models to define $y(t, k^*)$ in order to calculate the labels of the measurement traces:

- *intermediate value model*: in this model, the attacker considers an intermediate value of the cipher or the distance between two consecutive values processed, or
- *Hamming weight (HW) model*: in this model, the attacker assumes the HW of the intermediate value model.

Considering the intermediate value, the model may be more accurate but requires more resources. In particular, to gain stable estimations for each possible value an attacker needs a sufficient amount of measurement traces 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 preference for HW model 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 models for all datasets and algorithms.

The adversary first profiles the clone device with the known keys and uses the obtained profiles for the attack. In particular, the attack operates in two phases:

- *profiling phase:* N traces $\boldsymbol{x}_{p_1}, \ldots, \boldsymbol{x}_{p_N}$, plaintext/ciphertext $t_{p_1}, \ldots, t_{p_N}$ and the secret key k_p^*, such that the attacker can calculate the labels $y(t_{p_1}, k_p^*), \ldots, y(t_{p_N}, k_p^*)$.
- *attacking phase:* Q traces $\boldsymbol{x}_{a_1}, \ldots, \boldsymbol{x}_{a_Q}$ (independent from the profiling traces), plaintext/ciphertext $t_{a_1}, \ldots, t_{a_Q}$.

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

$$y(t_{a_1}, k_a^*), \ldots, y(t_{a_N}, k_a^*),$$

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

4 Framework Validation

In this section, we empirically validate the proposed framework using a representative set of profiled algorithms on publicly available datasets.

4.1 Datasets

In our experiments, we use three datasets that we consider to be a representative sample of commonly encountered scenarios. More precisely, the first dataset has no countermeasures and only a small amount of noise, which represents an easy scenario for profiled attack when the number of measurements is sufficient. Next, we consider a dataset without countermeasures but with a large amount of noise. With this dataset, we are approaching more realistic scenarios where profiled techniques have problems reaching high performance. Finally, the last dataset has implemented a countermeasure in the form of random delays which represents a realistic scenario for evaluation.

DPAcontest v4 Dataset The 4th version provides measurements of a masked AES software implementation [28]. As the mask is known, one can easily turn it into an unprotected scenario. As it is a software implementation, the most leaking operation is not the register writing but the processing of the S-box operation and we attack the first round. Accordingly, the leakage model changes to

$$Y(k^*) = \texttt{Sbox}[P_{b_1} \oplus k^*] \oplus \underbrace{M}_{\text{known mask}}, \tag{10}$$

where P_{b_1} is a plaintext byte and we choose $b_1 = 1$. Compared to the measurements from version 2, the SNR is much higher with a maximum value of 5.8577.

AES HD Dataset This dataset is chosen to target an unprotected implementation of AES-128. The core of AES-128 was written in VHDL in a round based architecture, taking 11 clock cycles for each encryption. A UART module is wrapped around the core in order to enable external communication. The module is designed to allow accelerated measurements in order to avoid any DC shift due to environmental variation over prolonged measurements. The total area footprint of the design contains 1 850 LUT and 742 flip-flops. Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board was used to implement the design. Side-channel traces were measured using a high sensitivity near-field EM probe, which was placed over a decoupling capacitor on the power line. Measurements were sampled on the Teledyne LeCroy Waverunner 610zi oscilloscope. A suitable and commonly used (HD) leakage model, when attacking the last round of an unprotected hardware implementation, is the register writing in the last round [28], i.e.,

$$Y(k^*) = HW(\underbrace{\texttt{Sbox}^{-1}[C_{b_1} \oplus k^*]}_{\text{previous register value}} \oplus \underbrace{C_{b_2}}_{\text{ciphertext byte}}), \tag{11}$$

where C_{b_1} and C_{b_2} are two ciphertext bytes, and the relation between b_1 and b_2 is given through the inverse ShiftRows operation of AES. $b_1 = 12$ was chosen,

which resulted in $b_2 = 8$, as it is one of the easiest bytes to attack. The obtained measurements that form the dataset are relatively noisy and the resulting model-based SNR (signal-to-noise ratio), i.e., $\frac{var(signal)}{var(noise)} = \frac{var(y(t,k^*))}{var(x-y(t,k^*))}$, has a maximum value of 0.0096. In total, 500 000 traces were captured corresponding to 500 000 randomly generated plaintexts, each trace with 1 250 features. Since this implementation leaks in the HD model, we denote it as AES HD. This dataset is publicly available at https://github.com/AESHD/AES_HD_Dataset.

Random Delay Dataset As our third use case, we use an actual protected implementation. Our target is a software implementation of AES on an 8-bit Atmel AVR microcontroller with implemented random delay countermeasure as described by Coron and Kyzhvatov in [29]. We mounted our attacks against the first AES key byte by targeting the first S-box operation. The dataset consists of 50 000 traces of 3 500 features each. For this dataset, the SNR has a maximum value of 0.0556. This dataset is publicly available at https://github.com/ikizhvatov/randomdelays-traces.

4.2 Feature Selection

For all the experiments, we use only the 50 most important features. To select those features, we use the Pearson correlation coefficient [30]:

$$Pearson(x, y) = \frac{\sum_{i=1}^{N}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}. \tag{12}$$

4.3 Profiled Algorithms

It is not a trivial task to select the best classifier for the given problem. Still, there are some classifiers that can be regarded as a usual choice when a highly accurate classification is sought [31]. In order to provide relevant experiments, we select several classifiers that are a common choice in SCA as given in Section 2.

Naive Bayes The Naive Bayes (NB) classifier [32] is also based on the Bayesian rule but is labeled "Naive" as it works under a simplifying assumption that the predictor features (measurements) are mutually independent among the D features, given the class value. 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. NB classifier outputs posterior probabilities as a result of the classification procedure [32]. The Bayes' formula is used to compute the posterior probability of each class value y given the vector of N observed feature values x.

Radial Kernel Support Vector Machines Radial Kernel Support Vector Machines (denoted SVM) is a kernel based machine learning family of methods that are used to accurately classify both linearly separable and linearly inseparable data. The idea for linearly inseparable data is to transform them to a higher dimensional space using a kernel function, wherein the data can usually be classified with higher accuracy. Radial kernel based SVM that is used here has two significant tuning parameters: the cost of the margin C and the kernel parameter γ. The scikit-learn implementation we use considers libsvm's C-SVC classifier that implements SMO-type algorithm [33]. 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 [34]. Decision trees choose their splitting attributes from a random subset of k attributes at each internal node. The best split is taken among these randomly chosen attributes and the trees are built without pruning, RF is a parametric algorithm with respect to the number of trees in the forest. RF is a stochastic algorithm because of its two sources of randomness: bootstrap sampling and attribute selection at node splitting. The most important parameter to tune is the number of trees I (note, we do not limit the tree size.)

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 [35]. Since we have more than one hidden layer in our experiments, this implementation of MLP represents a deep learning algorithm. We investigate the behavior of MLP with various activation functions, solvers, number of layers, and the number of nodes.

Template Attack The template attack (TA) relies on the Bayes theorem and considers the features as dependent. In the state-of-the-art, template attack relies mostly on a normal distribution. Accordingly, template attack assumes that each $P(\boldsymbol{X} = \boldsymbol{x}|Y = y)$ follows a (multivariate) Gaussian distribution that is parameterized by its mean and covariance matrix for each class Y. The authors of [1] propose to use only one pooled covariance matrix averaged over all classes Y to cope with statistical difficulties and thus a lower efficiency. In our experiments, we use the version of the attack that uses only one pooled covariance matrix.

Hyperparameter Tuning First, we conduct a tuning phase to select hyperparameters for which classifiers perform well over datasets. We emphasize that the tuned parameters represent a reasonable choice that exhibits good behavior but they should not be considered as the best possible ones. Indeed, if concentrating on any specific scenario, a more detailed tuning could probably result in a somewhat improved performance. Yet, since we consider scenarios where we

introduce perturbations in the testing phase, there is no guarantee that initial good parameters would be still good for the measurements with the added noise. At the same time, by having the suboptimal parameters we could prevent the classifiers from overfitting in the training phase, which can be beneficial when considering the testing phase with the added noise.

For template attack and Naive Bayes, there are no parameters to tune. For radial SVM, we conduct a grid search for $C = [0.001, 0.01, 0.1, 1]$ and $\gamma = [0.001, 0.01, 0.1, 1]$ and we select $C = 0.001, \gamma = 0.001$. For Random Forest, we experiment with a different number of trees $I = [10, 50, 100, 200, 500, 1\,000]$ and we select $I = 500$. Finally, for multilayer perceptron, we investigate activation functions $tanh$ and $ReLU$, solvers $lbfgs$ and $adam$, and number of layers/nodes $[(50, 10, 50), (50, 30, 20, 50), (50, 25, 10, 25, 50)]$. In the end, we select to use $tanh$ activation function, $adam$ solver, and $(50, 30, 20, 50)$ configuration of layers/nodes.

4.4 Figures of Merit

We consider three standard metrics when conducting SCA: accuracy, success rate, and guessing entropy. After running profiled attacks, we first obtain accuracy as the measure of performance. The accuracy is defined as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{13}$$

TP refers to true positive (correctly classified positive), TN to true negative (correctly classified negative), FP to false positive (falsely classified positive), and FN to false negative (falsely classified negative) instances. TP, TN, FP, and FN are well-defined for hypothesis testing and binary classification problems. In the multiclass classification, they are defined in one class–vs–all other classes manner and are calculated from the confusion matrix.

Most of the time, in side-channel analysis, an adversary is not only interested to predict 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. In particular, let us assume, given Q amount of samples in the attacking phase, an attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in a 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.

The success rate is defined as the average empirical probability that g_1 is equal to the secret key k_a^*. The guessing entropy is the average position of k_a^* in $\mathbf{g}$. As SCA metrics, we report the number of traces needed to reach a success rate SR of 0.9 as well as a guessing entropy GE of 10.

4.5 Experiments

We conduct our experiments for a scenario where $\epsilon = 0.1$ and $\delta = 0.1$. According to the Chernoff bound (Eq. (4)), we need to use $n \geq 149.7$ to achieve the

Algorithm 1 Algorithm to solve the probabilistic robustness evaluation problem.

Identify the perturbation space Δ and the random variable $\delta\theta$ with pdf $f_{\delta\theta}$ over Δ;
Select the accuracy ϵ and confidence δ
Identify the interested performance level set $\Gamma = \{\gamma_1, \cdots, \gamma_k\}$
$\hat{p}_{n,\Gamma}(\gamma)$= verification-problem$(\Delta, f_{\delta\theta}, u(\delta\theta), \Gamma, \epsilon, \delta)$
use $\hat{p}_{n,\Gamma}(\gamma)$

function verification-problem $(\Delta, f_{\delta\theta}, u(\delta\theta), \Gamma, \epsilon, \delta)$:
Draw $n \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$ samples $\delta\theta_1, \cdots, \delta\theta_n$ from $\delta\theta$ according to $f_{\delta\theta}$
For each $\gamma \in \Gamma$ estimate

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

Return $\hat{p}_{n,\Gamma}$

desired level of accuracy and confidence. Consequently, we set $n = 150$ in our experiments. Next, we select to work with the noise α equal to 0.005 that goes in the range $[-\alpha \cdot f, \alpha \cdot f]$, where f denotes the factor going in the range $[1, 50]$. Consequently, we can consider our scenario as working with 50 different noise intensities. Note, by using this noise level and factor range, we evaluate the effect of perturbations of magnitude up to the 25% of the signal, which represents a significant perturbation. Indeed, consider a realistic profiled attack where the profiling device and the device under the attack differ in measurements magnitude up to 25%. For sure, this represents a system that should be difficult to attack. Recall, since the $\delta = 0.1$, this means the probability equals 0.9. In all figures, we display 3 lines for each classifier. The full line depicts the behavior of the classifier while dotted lines depict $\pm\epsilon$. To assess the performance for a specified intensity level, we need to compare the values for the behavior of the classifier and $\pm\epsilon$. That range gives us how robust is the classifier, i.e., what kind of performance we can expect. Finally, in order to improve the readability of graphs, when success rate or guessing entropy are not able to reach the defined performance level for any number of test measurements, we say the value equals -1 and we denote it with lines plotted with $*$ symbol.

In Algorithm 1, we give the pseudocode of the procedure we follow in order to assess the robustness of a certain classifier. Note, since $n = 150$ and the number of intensities f equals 50, this means that for each scenario we need to run the attack phase 7 500 times. Note, due to the lack of space, all figures for success rate are given in Appendix A.

DPAv4 Results Figures 2a until 2f give results assuming the HW model. One can see a different behavior of algorithms with respect to the change in intensities. In particular, Naive Bayes and SVM stay nearly constant, while MLP, RF, and TA decrease slowly with the increase in the intensities. It is also possible to determine the intensity level up to which it seems better to use MLP/TA/RF

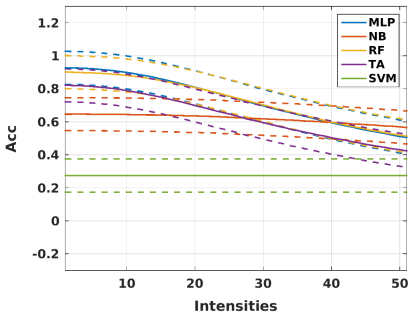
while after it, we see that SVM/NB result in better performance. This is interesting since it gives an intuition to the evaluator what kind of classifier should select with respect to the expected level of noise or the worst-case scenario. When looking at the guessing entropy and success rate, it is obvious that TA is performing the worst in the presence of the increasing intensities. We give a zoom in into guessing entropy in Figure 2c, showing that although all algorithms experience the decrease in the performance with the increasing intensities, the slope of RF and MLP is steeper than for SVM and NB. Accordingly, SVM and NB appear more resistant to noise addition.

Additionally, in this context, we evaluate one instance of SVM with suboptimal parameters and one instance with optimally trained parameters on the original datasets. Our experiments indicate that the optimal parameters for DPAv4 are $C = 1, \gamma = 1$. Figures 6a and 6b in Appendix B show their accuracies and guessing entropies. When considering accuracy, SVM with optimal parameters is performing better for small intensities as one would expect. Additionally, even for larger noise intensities, we can observe that accuracy of the optimally tuned SVM to be better than for SVM with suboptimal hyperparameters. For guessing entropy, SVM with optimal parameter values is performing better than SVM with suboptimal parameter values, but one can observe that when the noise is large, both SVM versions have almost the same performance. This points us that if the perturbations are very large, it could happen that there is no benefit from detailed tuning (in fact, since such tuning takes extra time, a reasonable choice would be to run only a coarse tuning).

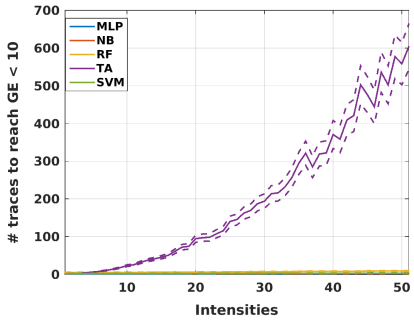
The results for the intermediate value model are given in Figures 2d until 5b. First, we observe that the accuracy is much lower than for the HW model, which is natural as now there are 256 classes while in the HW model there are only 9. SVM is the only algorithm that stays constant for all intensities (a consequence of the selected hyperparameters), while all the other methods experience a slow decrease in accuracy with the increase in intensity. In terms of guessing entropy, one can see that RF does not succeed in any scenario in the intermediate value model, while SVM and NB are relatively stable for differing intensities. For MLP, we observe a very graduate increase in the guessing entropy value with the increase in intensity. Again, TA is highly affected when increasing intensities. Finally, MLP, NB, and SVM are slightly more efficient considering the intermediate value model (256 classes) than for the HW model (9 classes).

AES HD Results The results for the AES HD are depicted in Figures 3a until 3e. Observe how in the HW model, accuracy stays relatively constant over intensities (see Figure 3a) and is naturally much lower than for DPAv4 dataset (since SNR is much lower).

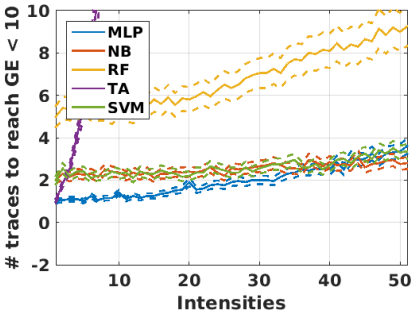
For guessing entropy, Figure 3c shows that TA and NB are performing the best and stay nearly constant over intensities. Recall that on the DPAv4 dataset, TA was nearly exponentially increasing with the increasing intensities. Interestingly, we see that MLP is failing for nearly all intensities. From the algorithms that manage to break the implementation, SVM performs the worst and we see



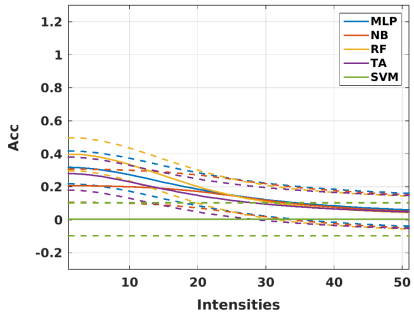
(a) DPAv4, HW model, accuracy



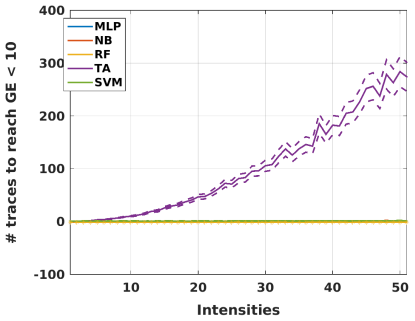
(b) DPAv4, HW model, guessing entropy



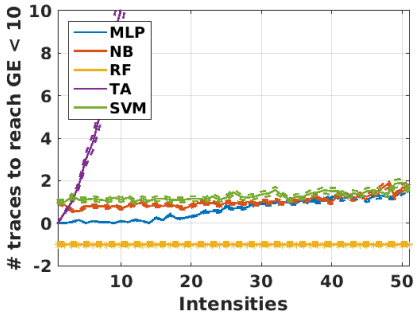
(c) DPAv4, HW model, guessing entropy (zoom in)



(d) DPAv4, value model, accuracy



(e) DPAv4, value model, guessing entropy



(f) DPAv4, value model, guessing entropy (zoom in)

that its performance deviates slightly with the change in the intensity of the noise. Differing from that, for RF we see large deviations over intensities. In Figure 5c, we see that only TA and NB reach the success rate level of 0.9 and TA stays constant while NB deteriorates slightly with the increase in intensities.

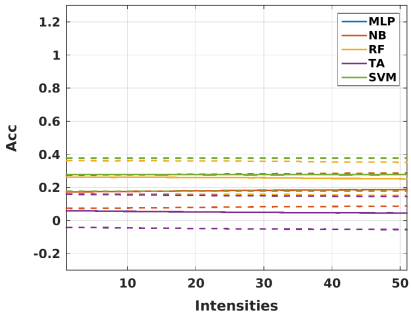
When considering the intermediate value model, we see a significantly different behavior of the algorithms. Accuracies decrease to almost 0 for all the algorithms (see Figure 3d and the zoom in in Figure 3b). At the same time, Figures 3e and 5d show that it is possible to reach the threshold levels for guessing entropy and success rate. In particular, MLP performs the best and stays nearly constant over intensities. NB is the second best with a slight decrease in the performance for increasing intensities. SVM is the third best with a constant behavior. TA shows high deviations which differs from the behavior seen for the HW model. Finally, RF fails for nearly all intensities.

Random Delay Results In terms of accuracy, all algorithms show a similar behavior for the Random Delay dataset (the performance is basically the same). For the guessing entropy and success rate, we see in Figures 4b and 5e that TA and NB perform constant over intensities and are better than RF for smaller intensities. For medium intensities, RF becomes much more efficient but it starts to fail for higher intensities. This interesting and novel behavior is discussed in more detail in Section 4.5.

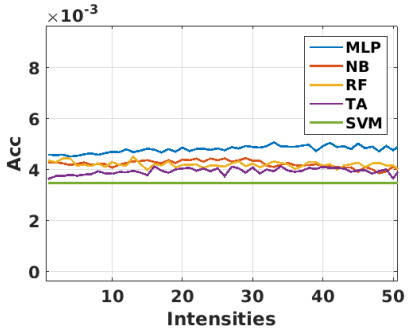
Considering the value model we see that SVM, RF, and TA are failing. MLP is failing for small intensities but succeeding to reach the guessing entropy threshold level for medium and higher intensities. NB is succeeding throughout all intensities but displays large deviations. When considering the success rate, only NB is reaching the success rate threshold while decreasing over intensities. We concentrate on explaining this behavior in Section 4.5.

General Observations and Remarks With our framework, we are able to observe four different types of classifier behavior over intensities, i.e., levels of perturbations. The most natural (and maybe expected behavior) is a slow decrease of performance with increasing level of noise. This behavior can be noticed for accuracies for the DPAv4 dataset for both models and all algorithms (except SVM with the intermediate value model). For guessing entropy and success rate, TA is nearly exponentially decreasing in performance for DPAv4, while others show a much slower decrease for these metrics. Thus, we state as the first finding that in case of high SNR scenarios, TA is not robust in the presence of perturbations. Differing from that, machine learning-based attacks like SVM, MLP, and NB perform well and are robust in the presence of perturbations.

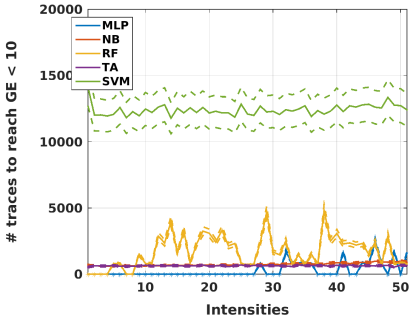
The second type of behavior is the constant performance in the presence of various levels of perturbations. We emphasize that such a behavior is highly desirable for side-channel algorithms in practical settings. This behavior can be seen for the Random Delay and AES HD datasets when considering accuracy and all algorithms. Thus, for datasets where the noise level is rather high, which actually can point that the constant behavior comes from the fact that the



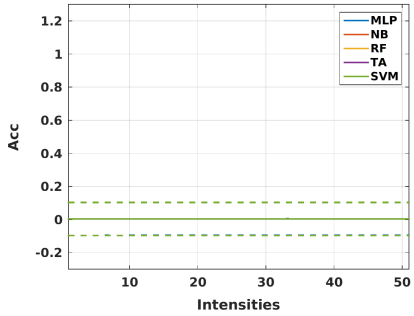
(a) AES HD, HW model, accuracy



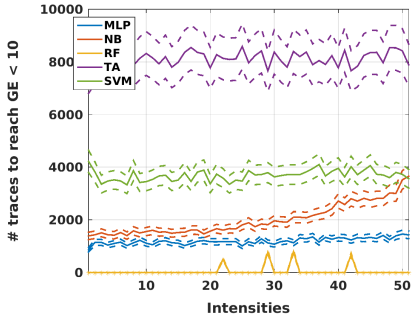
(b) AES HD, value model, accuracy (zoom in)



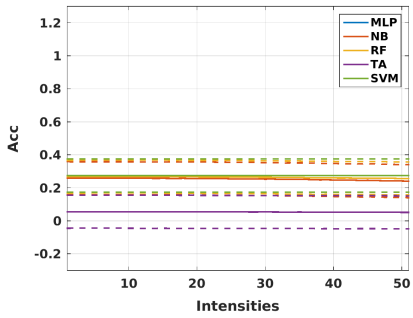
(c) AES HD, HW model, guessing entropy



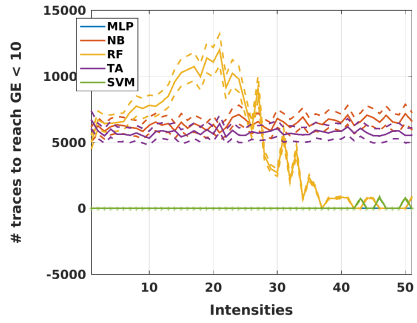
(d) AES HD, value model, accuracy



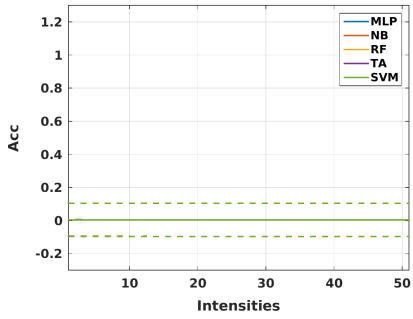
(e) AES HD, value model, guessing entropy



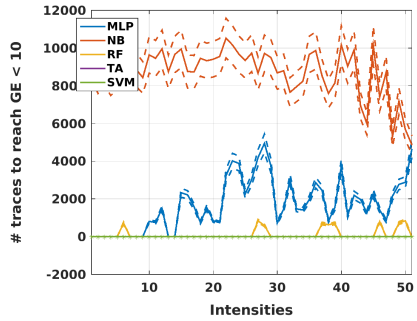
(a) Random Delay, HW model, accuracy



(b) Random Delay, HW model, guessing entropy



(c) Random Delay, value model, accuracy



(d) Random Delay, value model, guessing entropy

classifier works poor even when there is no noise so added noise does not decrease the performance. As two examples of robust classifiers for the HW model, we note TA and NB. In the intermediate value model, TA is performing much worse, while NB still keeps sufficiently good performance. Interestingly, MLP is performing poorly (even failing) considering the HW model but is the most efficient algorithm when considering the intermediate value model. We believe this is mainly a consequence of the imbalancedness of classes. Consequently, for high noise settings, NB is the most robust classifier as indicated by our experiments.

The third (somewhat unexpected) type of behavior is an increase in performance with increasing perturbations. This can be only observed on the Random Delay dataset. More precisely, for the HW model, we observe that RF is drastically improving its performance (both guessing entropy and success rate) with the increase in the perturbation level (until a certain perturbation value occurs when performance drops to -1 indicating there is simply too much noise to perform the attack correctly). Similarly, for the intermediate value model, NB is improving its performance with the increase in the perturbation level (mainly observable for success rate). Since the Random Delay dataset has a countermeasure, we see that adding noise improves the generalization ability of the classifier and allows it to avoid overfitting. Such behavior goes up to a certain level of noise after which the performance quickly drops. Consequently, we see there are even scenarios where robustness is not desirable quality since more robust classifier would tend more to overfit.

Finally, the fourth type of behavior is the one we call statistical anomaly and it can be seen for instance, for MLP classifier and the Random Delay dataset when considering the value model. There, the classifier is unsuccessful already for the scenario without noise and stays unsuccessful for most of the intensity levels. Still, there are some (random) intensity levels where the performance becomes very good. Unfortunately, this behavior cannot be really exploited since we do not know when (and if) will it occur because it happens only due to random changes.

When considering the hyperparameter tuning and perturbations, we see that fine-tuned tuning is beneficial if perturbations are not too big. If we expect to work in scenarios with large perturbations, actually it seems better to just conduct a coarse-grained tuning since the performance will not suffer due to the influence of noise.

5 Conclusions and Future Work

In this paper, we concentrate on the problem of profiled 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 profiled side-channel attack (i.e., classifier). Our framework supports datasets with any type of characteristics as well as different leakage models. To be able to offer such a general behavior, we model our setting as the expectation estimation problem

where we can 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 give an answer to the question which classifier is the best for some scenario. We give robustness analysis for all common figures of merit in SCA: accuracy, success rate, and guessing entropy.

We believe our framework to be a powerful tool that will allow researchers to compare the behavior of various classifiers in a more fair way than it is done up to now. Since profiled SCA in realistic settings should consider different devices for profiling and attacking, we actually see that our framework allows us more than just connecting SCA with problems that have strong theoretical results. More precisely, our framework actually allows us to model the realistic behavior of profiled SCA where because of several different sources of noise, uncertainty must occur. We note that our framework is demanding: to conduct a proper analysis the Chernoff bounds requires a large number of samples, which potentially can be a prohibiting factor for certain computationally intensive classifiers or very large datasets. One easy way how to circumvent this is to parallelize the process for different intensities. Another option, which we plan to explore in the future work is to use Chernoff-Okamoto bound that is less conservative than the Chernoff bound. Since it applies only when $p \leq 0.5$, it remains to be seen how useful this bound can be in the SCA domain.

References

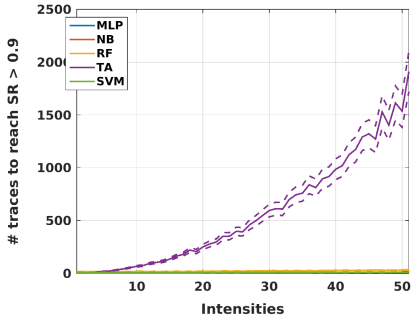
1. Choudary, O., Kuhn, M.G.: Efficient template attacks. In Francillon, A., Rohatgi, P., eds.: Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers. Volume 8419 of LNCS., Springer (2013) 253–270
2. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: CHES. Volume 2523 of LNCS., Springer (August 2002) 13–28 San Francisco Bay (Redwood City), USA.
3. Heuser, A., Zohner, M.: Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines. In Schindler, W., Huss, S.A., eds.: COSADE. Volume 7275 of LNCS., Springer (2012) 249–264
4. Cagli, E., Dumas, C., Prouff, E.: Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Preprocessing. In: Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. (2017) 45–68
5. Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., 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) 4095–4102
6. Picek, S., Heuser, A., Jovic, A., Bhasin, S., 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
7. Wolpert, D.H.: The Lack of a Priori Distinctions Between Learning Algorithms. Neural Comput. **8**(7) (October 1996) 1341–1390

8. Valiant, L.: Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World. Basic Books, Inc., New York, NY, USA (2013)
9. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Proceedings of CRYPTO'96. Volume 1109 of LNCS., Springer-Verlag (1996) 104–113
10. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. CRYPTO '99, London, UK, UK, Springer-Verlag (1999) 388–397
11. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Attali, I., Jensen, T., eds.: Smart Card Programming and Security, Berlin, Heidelberg, Springer Berlin Heidelberg (2001) 200–210
12. Genkin, D., Shamir, A., Tromer, E.: Acoustic cryptanalysis. Journal of Cryptology **30**(2) (Apr 2017) 392–443
13. Heuser, A., Rioul, O., Guilley, S.: Good is Not Good Enough — Deriving Optimal Distinguishers from Communication Theory. In Batina, L., Robshaw, M., eds.: CHES. Volume 8731 of Lecture Notes in Computer Science., Springer (2014)
14. Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., Standaert, F.: Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In Mangard, S., Poschmann, A.Y., eds.: Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers. Volume 9064 of Lecture Notes in Computer Science., Springer (2015) 20–33
15. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In LNCS, ed.: CHES. Volume 3659 of LNCS., Springer (Sept 2005) 30–46 Edinburgh, Scotland, UK.
16. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., 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.
17. Lerman, L., Bontempi, G., Markowitch, O.: Power analysis attack: An approach based on machine learning. Int. J. Appl. Cryptol. **3**(2) (June 2014) 97–115
18. Lerman, L., Bontempi, G., 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
19. Lerman, L., Medeiros, S.F., Bontempi, G., Markowitch, O.: A Machine Learning Approach Against a Masked AES. In: CARDIS. Lecture Notes in Computer Science, Springer (November 2013) Berlin, Germany.
20. Picek, S., Heuser, A., Guilley, S.: Template attack versus Bayes classifier. Journal of Cryptographic Engineering **7**(4) (Nov 2017) 343–351
21. Gilmore, R., Hanley, N., 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) 106–111
22. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Lightweight ciphers and their side-channel resilience. IEEE Transactions on Computers **PP**(99) (2017) 1–1
23. Maghrebi, H., Portigliatti, T., 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) 3–26
24. Alippi, C.: Intelligence for Embedded Systems: A Methodological Approach. Springer Publishing Company, Incorporated (2014)

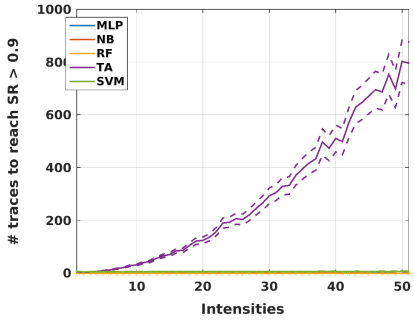
25. Rudin, W.: Real and Complex Analysis, 3rd Ed. McGraw-Hill, Inc., New York, NY, USA (1987)
26. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. Math. Statist. **23**(4) (12 1952) 493–507
27. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**(301) (1963) 13–30
28. TELECOM ParisTech SEN research group: DPA Contest (4^{th} edition) (2013–2014) `http://www.DPAcontest.org/v4/`.
29. Coron, J., Kizhvatov, I.: An Efficient Method for Random Delay Generation in Embedded Software. In: Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings. (2009) 156–170
30. James, G., Witten, D., Hastie, T., Tibsihrani, R.: An Introduction to Statistical Learning. Springer Texts in Statistics. Springer New York Heidelbert Dordrecht London (2001)
31. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Journal of Machine Learning Research **15** (2014) 3133–3181
32. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning **29**(2) (1997) 131–163
33. Fan, R.E., Chen, P.H., Lin, C.J.: Working Set Selection Using Second Order Information for Training Support Vector Machines. J. Mach. Learn. Res. **6** (December 2005) 1889–1918
34. Breiman, L.: Random Forests. Machine Learning **45**(1) (2001) 5–32
35. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016) `http://www.deeplearningbook.org`.

A Supporting Figures

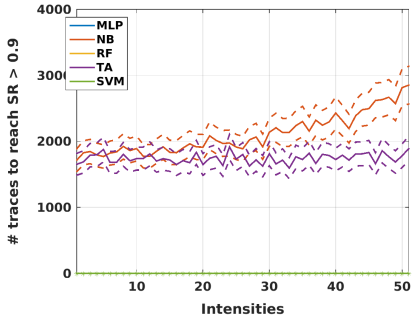
B The Influence of Hyperparameter Tuning



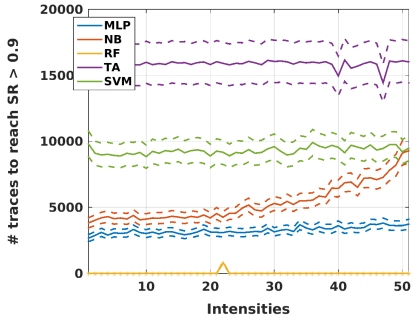
(a) DPAv4, HW model, success rate



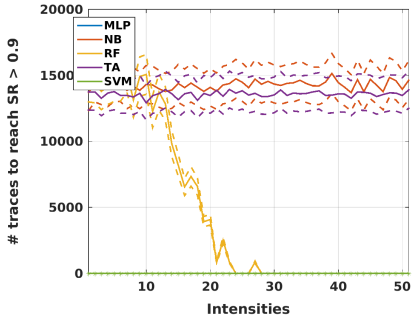
(b) DPAv4, value model, success rate



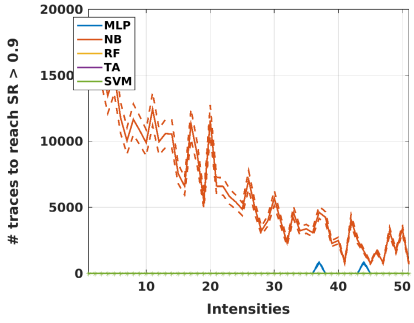
(c) AES HD, HW model, success rate



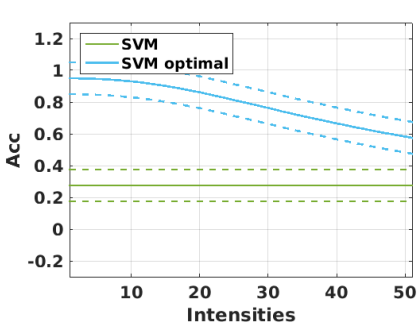
(d) AES HD, value model, success rate



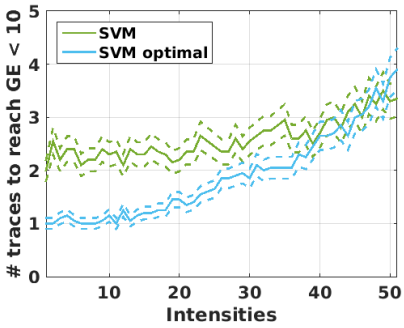
(e) Random Delay, HW model, success rate



(f) Random Delay, value model, success rate



(a) DPAv4, HW model, accuracy, SVM and SVM with optimized hyperparameters for this scenario



(b) DPAv4, HW model, guessing entropy, SVM and SVM with optimized hyperparameters for this scenario