Profiled Side-Channel Analysis in the Efficient Attacker Framework

Picek, Stjepan; Heuser, Annelie; Perin, Guilherme; Guilley, Sylvain

DOI
10.1007/978-3-030-97348-3_3

Publication date
2022

Document Version
Accepted author manuscript

Published in
Smart Card Research and Advanced Applications

Citation (APA)
Picek, S., Heuser, A., Perin, G., & Guilley, S. (2022). Profiled Side-Channel Analysis in the Efficient Attacker Framework. In V. Grosso, & T. Pöppelmann (Eds.), *Smart Card Research and Advanced Applications : 20th International Conference, CARDIS 2021, Revised Selected Papers* (Vol. 13173, pp. 44-63). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 13173). Springer. https://doi.org/10.1007/978-3-030-97348-3_3

Profiled Side-channel Analysis in the Efficient Attacker Framework

Stjepan Picek[1], Annelie Heuser[2], Guilherme Perin[1], and Sylvain Guilley[3]

[1] Delft University of Technology, The Netherlands
[2] Univ Rennes, Inria, CNRS, IRISA, France
[3] Secure-IC S.A.S., France

Abstract. Profiled side-channel attacks represent the most powerful category of side-channel attacks. There, the attacker has access to a clone device to profile its leaking behavior. Additionally, it is common to consider the attacker unbounded in power to allow the worst-case security analysis. This paper starts with a different premise where we are interested in the minimum power that the attacker requires to conduct a successful attack. We propose a new framework for profiled side-channel analysis that we call the Efficient Attacker Framework. With it, we require attacks to be as powerful as possible, but we also provide a setting that inherently allows a more objective analysis among attacks. To confirm our theoretical results, we provide an experimental evaluation of our framework in the context of deep learning-based side-channel analysis.

1 Introduction

Side-channel analysis (SCA) is a threat that exploits weaknesses in physical implementations of cryptographic algorithms rather than the algorithms themselves [1]. Profiled SCA performs the worst-case security analysis by considering the most powerful side-channel attacker with access to an open (the keys can be chosen or are known by the attacker) clone device. Additionally, the SCA community considers an attacker in the setting with unbounded power, e.g., the attacker can obtain any number of profiling or attack traces and has unlimited computational power.

In the last two decades, besides template attack and its variants [2, 3], the SCA community started using machine learning to conduct profiled attacks. Those results proved to be highly competitive compared to template attack, and, in many scenarios, machine learning methods surpassed template attack performance [4–6]. Unfortunately, in these scenarios, the experimental setup is often arbitrarily limited, and no clear guidelines on the limitation of profiling traces or the hyperparameter tuning phase are offered or discussed.

More recently, the SCA community started to experiment with deep learning where such methods bested both template attack and other machine learning methods [7–9]. Again, no clear guidelines on the number of profiling traces were given or investigated. Simultaneously, the researchers started to give more attention to the hyperparameter tuning, but the results are still far from definitive ones, see, e.g., [10, 11]. Consequently, there is an evident lack of evaluation guidelines/frameworks in the context of profiled analysis to understand various attacks' performance or how they compare. This gap is highly important as state-of-the-art results with deep learning successfully and efficiently break publicly available targets.

This paper aims to extend the currently used evaluation techniques to a framework that determines the least powerful attacker that can still reveal secret information. To achieve this, we evaluate the limit on 1) the number of measurements the attacker can collect in the training phase and 2) the number of hyperparameter tuning experiments. It could sound counter-intuitive to make such limitations as one can argue there is no reason why an attacker cannot collect a large number of measurements or run hyperparameter tuning as long as needed (or select an algorithm that has no hyperparameters to tune). We claim that there are several reasons for that:

1. By considering a scenario where an unlimited number of measurements are available, we "allow" less powerful attacks. More precisely, the attacker can use a larger set of measurements to compensate for less powerful profiling models.
2. By considering a scenario where a computationally unbounded attacker runs the analysis, one assumes the attacker can always find the best possible attack while that seldom happens in practice.
3. The target device may include a countermeasure that limits the number of exploitable measurements. The experimental setup can have constraints that limit the allowed length of the hyperparameter tuning phase.
4. Although taking measurements or running more experiments is "cheap", there is always a point where this is more effort than the target/secret is worth.
5. Having more measurements does not guarantee better results, especially in realistic scenarios. Consider the case where one device is used for profiling and the other for the attack, i.e., the portability setting (a realistic case that is usually simplified in research works where only a single device is used [7–9, 12]). Then, adding more measurements to the profiling phase can cause machine learning methods to overfit [4] [12]. The same issue can happen due to a too detailed tuning phase.

As far as we know, there are no previous works considering profiling and realistic attacker evaluation frameworks. When the attacker is restricted, it is usually set as one of several tested scenarios (e.g., testing a classifier's performance with specific hyperparameters or a different number of measurements in the training phase). Alternatively, it is motivated by some limitations in the data acquisition or evaluation process.

In this paper, we present the following main contributions:

1. We propose a new framework for profiled side-channel analysis where we evaluate the minimum power of an attacker in the profiling phase to still be successful in the test phase. We also introduce a new threat model that differs

[4] Overfitting occurs when the learning model learns the data too well and cannot adapt to previously unseen data.

from a common one by considering a more realistic attacker. The attacker in our threat model is still powerful from the computational perspective and the perspective of the learning models that can be built. In other words, we move from the problem of simply breaking the target (which is well-explored and with strong results, especially when considering deep learning) to a problem where we break the target with a minimal number of measurements and minimal hyperparameter tuning. We consider our framework to be intuitive and easily adaptable to many realistic scenarios.

2. We strengthen our results with an experimental evaluation conducted on publicly available datasets protected with masking countermeasures. We explore two commonly used leakage models and two neural network types.

The code is publicly available at
`https://github.com/AISyLab/EfficientAttackerFramework`.

2 Existing Frameworks for Side-channel Evaluation

2.1 Scientific Metrics

The most common evaluation metrics in the side-channel analysis are success rate (SR) and guessing entropy (GE) [13]. GE states the average number of key candidates an adversary needs to test to reveal the secret key after conducting a side-channel analysis. In particular, given Q traces in the attack phase, an attack outputs a key guessing vector $g = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ 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. The guessing entropy is the average position of k_a^* in g over multiple experiments. The success rate is defined as the average empirical probability that g_1 equals the secret key k_a^*.

In practice, one may consider leakage models $Y(\cdot)$ that are bijective functions. Thus, each output probability calculated from the classifiers for $Y(k)$ directly relates to one key candidate k. When $Y(\cdot)$ is not bijective, several key candidates k may get assigned with the same output probabilities, which is why a single trace attack ($Q = 1$) may not be possible in the case of non-bijective leakage models. Further, to calculate the key guessing vector g over Q attack traces, the (log-)likelihood principle is used.

Remark 1. SR and GE are used for practical evaluations in both non-profiling and profiling scenarios. Typically, they are given over a range of traces used in the attack phase (i.e., for $q = 1, 2, \ldots, Q$). If these metrics are used in profiling scenarios, there are no clear guidelines for evaluating attacks. Most of the time, the number of training measurements N in the profiling stage is (arbitrary) fixed, making comparisons and meaningful conclusions on profiled side-channel attacks or resistance of implementations hard and unreliable in most scenarios.

Whitnall and Oswald introduced a more theoretical framework that aims at comparing distinguishing powers instead of estimators of attacks [14, 15]. Accordingly, the profiling dataset N size does not play any role in this framework.

The most popular metrics of the framework are the relative and absolute distinguishing margins in which the correct key's output score and the value for the highest-ranked alternative are compared.

Another approach to compare side-channel attacks uses closed-form expressions of distinguishers [16], enabling conclusions about distinguishers without the requirement of actual measurements. Unfortunately, only a few closed-form expressions of distinguishers have been achieved so far.

Regarding masking countermeasures, Duc et al. defined information-theoretical bounds on the success rate depending on the number of measurements, shares, and independent on the concrete estimated side-channel attack [17]. In [18], the authors provided information-theoretic tools to bound the model errors in side-channel evaluations concerning the choice of the leakage model.

Typically, to assess the performance of machine learning classifiers, accuracy is used [19]. A detailed comparison between accuracy (but also other machine learning metrics like precision, recall, F1) and guessing entropy/success rate is given in [6], which details that such metrics may not always be a proper choice for assessing the attack performance in side-channel analysis.

2.2 Practical Evaluation Testing

While most of these previous metrics are relevant in some contexts and scenarios, a different approach is required to make research statements in the context of profiled attacks. This issue becomes even more evident when looking at practical evaluation used in standardization processes. In practice, there are two main practical schemes:

1. Test-based schemes, such as NIST FIPS 140 [20] and its application to the mitigation of other attacks (part of Appendix F, in particular, non-invasive attacks ISO/IEC 17825 [21]).
2. Evaluation-based schemes, such as Common Criteria (CC, ISO/IEC 15408 [22]).

Interestingly, both FIPS 140 and CC pay attention to the limited amount of resources spent. When considering FIPS 140 / ISO/IEC 17825, the requirement is more on the attack traces, but regarding CC, the evaluation of attacks is considered under two phases: identification (which matches with the training phase in the context of profiled side-channel attacks) and exploitation (which matches with the attack phase in the context of profiled side-channel attacks). Strictly speaking, the distinction is for CC version 2, but it still implicitly holds for version 3. Several factors are considered for the evaluations of attacks, namely: elapsed time, expertise, knowledge of the Target Of Evaluation (TOE), access to TOE, equipment, open samples. The first factor, elapsed time, directly connects with the acquisition of traces in the profiling phase and the hyperparameter tuning. Indeed, according to the guidance "Application of Attack Potential to Smartcards" [23], the score is considered:

- 0 if the profiling of the traces can be performed in less than one hour,
- 1 if the profiling of the traces can be performed in less than one day,
- 2 if the profiling of the traces can be performed in less than one week,
- 3 if the profiling of the traces can be performed in less than one month,

– 5 if the profiling of the traces cannot be performed in less than one month.

Accordingly, we see that the CC guidance favors attacks, realized with as little profiling effort as possible. This profiling effort can go in the direction of the number of required measurements, the number of experiments in the hyperparameter tuning phase, or both.

2.3 Practical Observations and Effects of Aging

Besides overfitting (see details in Section 1), another difficulty for profiled attacks is that the collection of side-channel traces becomes less reliable after a long period. Due to temperature and environmental conditions evolution over time, some trend noise must be added to the side-channel traces. For instance, this has been characterized by Heuser et al. in [24], where it is proven that trend noise drastically impedes SCA. Similar findings are confirmed by Cao et al. [25]. Efficient distinguishing situations, such as that depicted in Figure 1 shows that the best number of traces to estimate a distinguisher is not always "the maximal". This is illustrated on a simple "difference of means" attack representing side-channel attack on DPA contest 4.2 traces [26] (the second implementation (v4.2) is based on an improved version of the first version - v4).

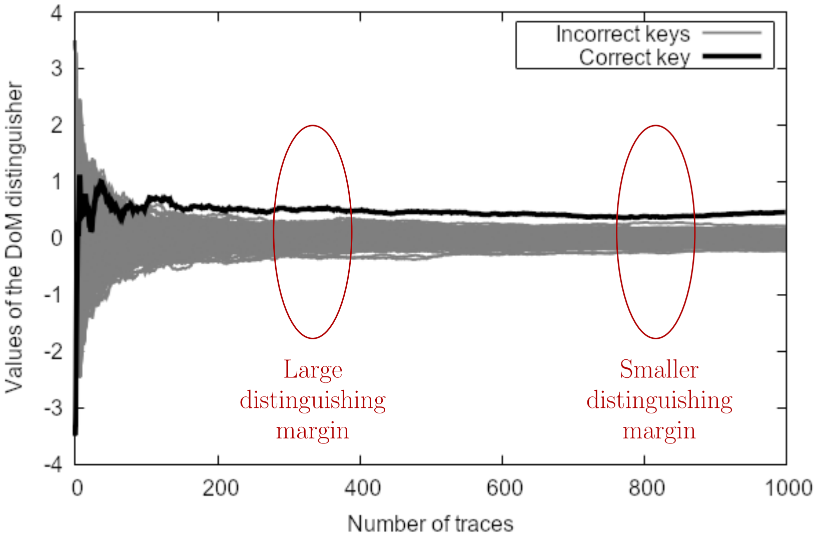


Fig. 1: Difference of Means (DoM) distinguisher estimation for all key bytes (the correct one and all incorrect ones).

3 The Efficient Attacker Framework

3.1 Threat Model

The adversary has access to a clone device running the target cryptographic algorithm. This device can be queried with a known key and plaintext while the corresponding leakage measurement is stored. Commonly, the adversary can have infinite queries to characterize a precise profiling model. There are no limits on how many experiments he can do to find such a profiling model. Next, the adversary queries the attack device with known plaintext to obtain the unknown key. The corresponding side-channel leakage measurement is compared to the characterized profiling model to recover the key.

In our threat model, the adversary has a limited number of queries to characterize a profiling model. Additionally, he has a limited number of experiments to conduct hyperparameter tuning. Note, while our framework allows various machine learning tasks, we concentrate on the classification task in this paper, as it is common in the profiled SCA [7–9].

3.2 Components of a Successful Attack

Current evaluations for profiled SCA mostly assume that the attacker is unbounded in his computational power. This assumption aims to provide the worst-case scenario for the designer, which should help assess the risk properly. Although the attacker is considered unbounded, he is always bounded, with bounds set ad-hoc, and there are no clear directions one should follow when modeling the realistic attacker.

First, we discuss two core assumptions we make in this research. These need to be fulfilled so that general meaningful comparisons between profiled attacks can be made, and our framework can provide exploitable results:

1. Attack must be possible. While our framework does not require the attacker always to succeed, the attack must be possible. For instance, having measurements completely uncorrelated with the labels (set of variables defined from a leakage model) will make our framework not useful. Still, no side-channel attack can succeed if there is no statistical connection between the measurements and labels. Consequently, this is not a drawback of our framework.
2. We consider only profiled (supervised) attacks, and therefore, profiling measurements need to allow learnability about the problem. Profiling measurements that are completely uncorrelated with the attack measurements would make our framework unusable. The hyperparameter tuning (if possible) must allow reaching a useful profiling model. Again, the profiled attacks cannot work if the previous conditions are not fulfilled, which does not represent our framework's disadvantage.

Next, we examine the three components of a successful attack. The worst-case (strongest) attacker will be unbounded in all three components. Simultaneously, fulfilling only one or two of those components accounts for more realistic settings one encounters in practice:

1. Quantity (the number of measurements) - there must be sufficient measurements in the profiling/test phase to conduct the attack, i.e., to build a reliable profiling model that generalizes to the unseen data. This criterion is a natural one and is already well-known in SCA as researchers usually report the attack's performance concerning a different number of measurements. There is much less research to determine the minimum number of measurements for a successful attack.
2. Quality (based on the available measurements, it must be possible to find the mapping f between the input (measurements) and output (labels)) - the measurements need to be of sufficient quality to conduct the attack. This condition could be translated into the requirement that the SNR should be sufficiently high or that the data need to have all information required to model the leakage correctly. Finally, this component includes the leakage model's quality, i.e., the considered leakage model provides sufficient information and the distribution of leakages. Again, like the previous component, this one is well addressed in the SCA community as researchers usually conduct various pre-processing steps, e.g., to select/transform features or align traces.
3. Learnability (hyperparameter tuning) - the attacker needs to learn the profiling model. This perspective also accounts for finding the best possible hyperparameters for the profiling model. The learnability is naturally connected with the quantity and quality components. This component is significantly less addressed, but more recent works show the SCA researchers becoming more interested in it [9–11, 27], confirming our claims about the learnability importance. We note that while the researchers usually conduct various tuning procedures, they rarely report how difficult it was to find the hyperparameters used in the end.

We should not limit the quality component: if the attacker can obtain measurements, those measurements should be of the best possible quality. When discussing the quantity and learnability components, we can (and we must) evaluate the limit of the number of profiling measurements and experiments in the tuning phase since:

1. If always considering the extreme case of unbounded measurements in the profiling phase, we "allow" to utilize weaker attack, which may only work in this extreme scenario. On the other hand, if we consider the minimum number of available traces in the profiling phase while still succeeding in the attack phase, we promote efficient attacks.
2. Theoretically, the attacker who is unbounded in his capabilities could break cryptographic implementations even with a single measurement as he can always find the optimal attack. This reasoning suggests that ultimately, the designer could do nothing to stop the attack.

Remark 2. Having a limited number of measurements or time to conduct hyperparameter tuning is a realistic occurrence in practical scenarios, as the attacker may be limited by time, resources, and also face implemented countermeasures, preventing him from taking an arbitrarily large number of side-channel measurements while knowing the secret key of the device.

To conclude, we need to consider an attacker who can perform a successful attack with the smallest possible number of profiling measurements N, where success is defined over a performance metric ρ with a threshold of δ. To reach that success, the attacker should use the smallest possible number of tuning experiments H (where h represents a specific set of hyperparameters, i.e., a specific profiling model).

Example 1. Consider ρ being the guessing entropy < 20, which is a common threshold value in the side-channel analysis, see, e.g., [6]. Then, the measure of the attacker's power is 1) the number of profiling traces N he needs to train a profiling model, which is then used on attack traces (of size Q) to break the implementation, 2) the number of experiments conducted before finding the hyperparameters resulting in a strong attack, or 3) both the number of profiling traces and hyperparameter tuning experiments.

3.3 Framework Description

The goal for machine learning classification task is to learn a mapping (model) f from $\mathcal{X}$ to $\mathcal{Y}$, i.e., $Y \leftarrow f(X, \theta)$ where X are samples drawn i.i.d. from set $\mathcal{X}$ and where the cardinality of X equals N. Let θ be the profiling model's parameters that result in the best possible approximation from h hyperparameter combinations. Additionally, let $\mathbf{g}_{Q,f} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ be the guessing vector from the profiled side-channel attack using Q traces in the attack phase, and the profiling model f built in the profiling phase as an input. In practice, the estimation of f depends on hyperparameters h, which we denote by f_h when the dependency is emphasized. Then, $\rho(g_{Q,f}, k_a^*)$ represents the performance metric of the profiled side-channel attack using the secret key k_a^* to evaluate the success.

For a given number of attack traces Q and $h_1, \ldots, h_H$ hyperparameter tuning selections (H being the number of different hyperparameter sets), the Efficient Attacker Framework aims at minimizing the number of profiling traces N to model the function f_{h_i} with hyperparameter selection h_i ($1 \leq i \leq H$), such that the performance metric is still below (or above) a certain threshold δ:

$$\min\{N \ : \ \rho(g_{Q,f_{h_i}}, k_a^*) < \delta\}, \text{ where } N, i \geq 1 \text{ and } i \leq H. \tag{1}$$

Algorithm 1 gives the procedure of the evaluation in the Efficient Attacker Framework, and a motivating example is given in Example 2. Note that the framework allows conducting experiments in parallel to the data acquisition phase. Indeed, one can start with evaluating the performance regardless of the number of already acquired measurements. For example, the attacker can assume the regime where he downloads new measurements every hour and repeats the experiments with an always-increasing number of measurements.

Algorithm 1 increases the number of profiling traces until the stop condition (statically defined) is satisfied. As a secondary objective, it attempts to reduce

```
Static parameters: Maximum size $H$ of hyperparameter models to consider,
                   a performance metric $\rho$ and a threshold value $\delta$, e.g., GE
                   $< 20$
Input            : Profiling and attacking device to collect traces from
Output           : Minimum number of profiling traces $N$

Capture a test dataset (with secret key $k_a^*$). Its size $Q$ depends on the
  expected performance of the attack. For instance, this test dataset can be as
  small as one trace!
Training_set $\leftarrow \varnothing$
$N \leftarrow 0$
while True do
    Capture one trace        // A speed-up can be obtained by advancing
      faster, e.g., 10 by 10 traces
    Append them to Training_set, $N \leftarrow N + 1$
    for $i = 1;\ i \leq H;\ i++$ do
        (Randomly) select hyperparameters $h$
        Perform Training with selected hyperparameters and obtain a model
          $f_h$
        Receive $\rho(g_{Q,f_{h_i}}, k_a^*)$
        if $\rho < \delta$ then                    // The model is good enough
            store hyperparameter selection $h$
            break
return Minimum number of profiling traces $N$
```

Algorithm 1: Conceptual evaluation procedure in the Efficient Attacker Framework.

the search space for the hyperparameters models, with the learning phase to be as computationally efficient as possible.

Remark 3. Algorithm 1 considers both the number of profiling traces and hyperparameter tuning experiments, but this can be easily adjusted for only one of those options, extended or replaced by other performance evaluations. For instance, if using a template attack, there are no hyperparameters to tune, which means that only the number of profiling traces is relevant. On the other hand, if facing a setting where one cannot obtain enough measurements to reach δ, then the natural choice is not to limit the number of measurements even more but to consider the number of hyperparameter tuning experiments. While we consider the number of hyperparameter tuning experiments from the learnability perspective in this paper, this could be easily cast, for instance, to the selection of points of interest with template attack.

Example 2. A standard performance metric used in the side-channel analysis is guessing entropy with, e.g., a threshold $\delta = 20$. In the Efficient Attacker Framework, one would find the minimum number of profiling traces N and hyperparameter experiments H to reach a guessing entropy below 20 for a fixed number of Q attack traces. This setting ensures that key enumeration algorithms [28] (when attacking several key bytes, as in AES-128 where there are 16 bytes of the key that needs to be recovered simultaneously for a full key recovery attack) are efficient. Typically, Q ranges over a set of values. Experimental results are discussed in Section 4.

Remark 4. In practice, Algorithm 1 shall be evaluated several times to get an empirical estimation $\hat{\mathbb{E}}(N)$ of the minimum number of profiling traces. This can be achieved by averaging several evaluations of Algorithm 1 (as done in non-profiled side-channel attack-oriented frameworks, see [13, §3.1]).

Remark 5. The Efficient Attacker Framework is evaluator-oriented and aims at unleashing profiled attacks even with frugal learning constraints. This reflects some situations where the number of interactions with the device is limited:

- by design, e.g., owing to enforcement of countermeasures such as limited number of cryptographic executions until system end-of-life, or
- by certification constraints such as limited “elapsed time” in the Common Evaluation Methodology (CEM [29, B.4.2.2]) of the Common Criteria.

Remark 6. If two profiling models exhibit very similar performance but require a radically different amount of resources, then a Pareto front of solutions (i.e., a set of non-dominated solutions) needs to be given where the designer can decide on a proper trade-off.

We reiterate that our framework is not designed to force the attacker to use a small number of measurements in the profiling phase or limit the number of experiments in the hyperparameter tuning phase. Instead, it forces the attacker (evaluator) to find the smallest number of traces and tuning experiments to attack the target successfully.

4 Experimental Evaluation

4.1 Datasets

The first dataset we consider is the ASCAD with a fixed key dataset. The measurements are obtained from an 8-bit AVR microcontroller running a masked AES-128 implementation, where the side-channel is electromagnetic emanation [30]. This dataset has the same key for the profiling and attack phase. There are 50 000 traces for profiling and 10 000 for the attack. We use a pre-selected window of 700 features for the raw trace, and we attack key byte 3, which is the first masked key byte, as commonly done in the literature [30].

The second dataset is a version of the ASCAD dataset with random keys (denoted ASCAD random keys dataset) in the profiling set. The dataset consists of 200 000 traces for profiling and 100 000 for the attack. We use a pre-selected window of 1 400 features for this dataset and attack key byte 3 (the first masked key byte).

4.2 Efficient Attacker Framework Evaluation

The Efficient Attacker Framework enables us to compare side-channel attacks and gives a fair comparison between leakage models. For deep learning-based side-channel attacks, it is often assumed to consider the most accurate leakage model, i.e., using the intermediate value as class variables (the Identity leakage model[5]) [9,27,31], which results in 2^b classes where b is the number of considered bits. In an unsupervised setting (i.e., non-profiled attacks), using the Hamming weight or the Hamming distance leakage model is a common choice, which results in $b+1$ classes only. Using $b+1$ Hamming weight/distance classes to guess a key value in $\{0, \ldots, 2^b-1\}$ cannot result in a single trace attack on average. However, using the Hamming weight/distance leakage models may require fewer traces in the profiling phase to gain good quality estimates of the leakage models (as there are fewer classes to consider). It is, therefore, not straightforward to determine what leakage model is most suitable. Consequently, to give a fair comparison, one should include a dependency on the number of traces in the profiling phase, as done in the Efficient Attacker Framework.

As a metric, we consider guessing entropy (GE), and in particular, we give the minimum number of profiling and attack traces to reach GE < 20. We randomly define hyperparameters for every training procedure for multilayer perceptron (MLP) and convolutional neural networks (CNNs) according to the hyperparameter ranges provided in Table 1. This scenario represents an optimized random hyperparameter search since the hyperparameter ranges are chosen based on the optimized minimum and maximum values (the minimal and maximal values are selected based on related works) [9,11,27,31]. The number of epochs is set to 50 (we observed that the models tend to overfit and degrade the generalization after 50 epochs), and the backpropagation algorithm optimizer is *Adam*. The weights and biases are initialized in a randomly uniform way. We use the batch normalization layer to avoid overfitting, which normalizes the input layer by adjusting and scaling the activations. For CNNs, a pooling layer (with hyperparameters range specified in Table 1) always comes after a convolution layer.

We do not explicitly discuss the time perspective here (e.g., the number of hours or days needed to conduct the experiments). Comparing the number of tuning experiments gives a fair evaluation, regardless of the time needed to run those experiments. We note that the number of tuning experiments up to 50 is low, although we manage to break the target. There is no constraint on the number of experiments one can use with our framework. Additionally, as we work with guessing entropy, each attack is repeated 100 times, which gives much higher computational complexity than one could conclude solely based on the number of tuning experiments. Every figure contains the results for the Hamming weight and Identity (i.e., intermediate value) leakage models, as AES operates on $b = 8$ bits. We select the best neural network model out of 5, 10,

[5] By the “Identity leakage model”, we mean that we do not assume the number of classes can be reduced owing to model degeneracy, as would be the case for instance in the “Hamming weight leakage model”, where it is assumed that the leakage Y depends in X only through $w_H(X)$ (the Hamming weight of X).

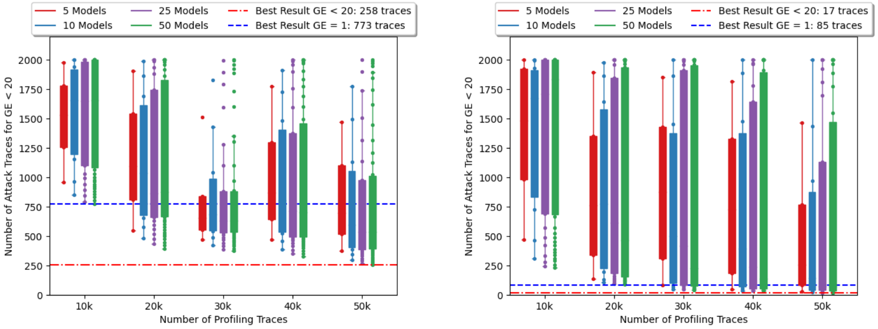
Table 1: Hyperparameter search space for MLP and CNNs.

Hyperparameter	MLP			CNN		
	min	**max**	**step**	**min**	**max**	**step**
Learning Rate	0.0001	0.001	0.0001	0.0001	0.001	0.0001
Mini-batch	100	1 000	100	100	1 000	100
Dense (fully-connected) layers	1	4	1	1	4	1
Neurons (for dense or fc layers)	100	400	100	100	400	100
Convolution layers	-	-	-	1	2	1
Filters	-	-	-	4	16	4
Kernel Size	-	-	-	2	10	2
Stride	-	-	-	1	4	1
Pooling Size	-	-	-	1	4	1
Pooling Stride	-	-	-	1	4	1
Activation function (all layers)	ReLU, Tanh, ELU, or SELU					

25, or 50 trained profiling models for each leakage model and a different number of profiling traces. More precisely, we compare the performance of a different number of profiling models (thus, forming ensembles) as done in [31]. Here, the main idea is to demonstrate that the *learnability* also represents an important dimension in our framework. All the graphs are to be viewed in color.

ASCAD Fixed Key Dataset We depict results for the ASCAD fixed key dataset in Figures 2 and 3, for MLP and CNN, respectively. For the CNN case, we also depict the results by using the architecture from [9]. The results confirm the importance of considering the number of profiling traces and hyperparameter tuning. In particular, for MLP in combination with the HW leakage model: 25 and 50 models behave the same for 30 000 profiling traces, indicating they are "equally" good. Nevertheless, restricting the number of profiling traces, e.g., to 20 000 reveals that 50 models reach better attack performance. Finally, many models perform better for 35 000 than 45 000 profiling traces, indicating that the data cannot fit the current network capacity.

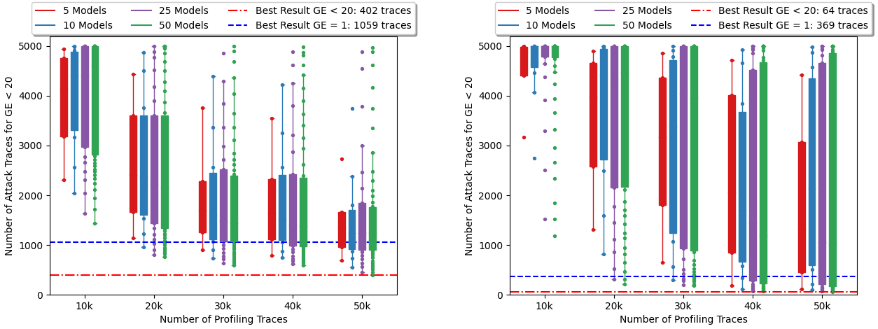
ASCAD Random Keys Dataset Figures 4 and 5 show results for the ASCAD with the random keys dataset. In Figure 4, we give results for MLP with hyperparameters defined per Table 1. Notice that considering a different number of profiling traces shows radically different behaviors. The more important is to observe that the profiling traces component becomes not as relevant as increasing the number of searched MLP models, especially for the Identity leakage model in Figure 4b. For example, by keeping 40 000 profiling traces, the best number of attack traces after searching for five models is around 3 100 traces, while the



(a) MLP, HW leakage model.

(b) MLP, Identity leakage model.

Fig. 2: Profiled SCA on the ASCAD fixed key dataset with MLP.



(a) CNNs, HW leakage model.

(b) CNNs, Identity leakage model.

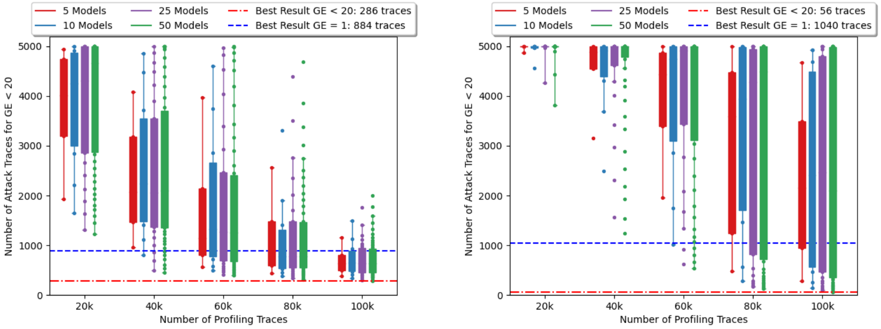
Fig. 3: Profiled SCA on the ASCAD fixed key dataset with CNNs.

minimum number of attack traces to reach GE< 20 with 50 models is close to 1 000 traces.

Figure 5 depicts the results for CNN architectures confirming the previous observations. In this particular example, we can immediately see how important it is to keep increasing the number of profiling traces as well as the number of searched models. This is expected as, due to the larger number of hyperparameter options, CNNs are more difficult to tune compared to MLP. In this case, the Efficient Attacker Framework reveals that increasing both components (profiling traces and learnability) makes the attack stronger.

4.3 Strong Adversary in the Efficient Attacker Framework

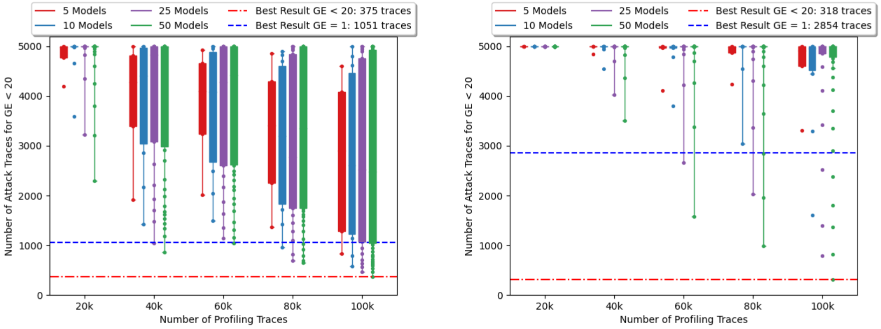
In the previous section, we evaluated our framework under the perspective of an adversary with strong side-channel capabilities (a profiled attack is mounted over optimal trace interval containing leaky points-of-interests). However, this



(a) MLP, HW leakage model.

(b) MLP, Identity leakage model.

Fig. 4: Profiled SCA on the ASCAD random keys dataset with MLP.



(a) CNNs, HW leakage model.

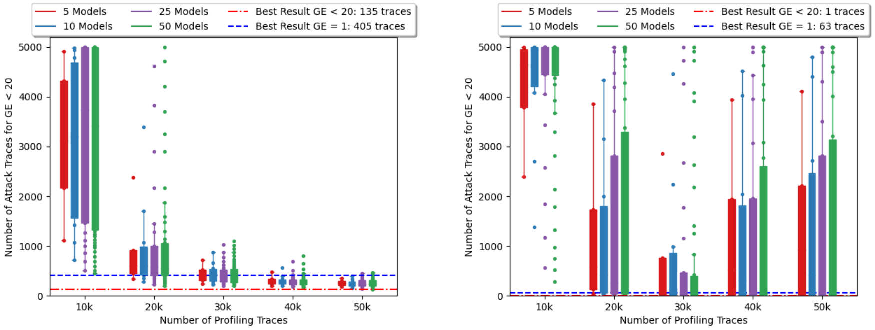
(b) CNNs, Identity leakage model.

Fig. 5: Profiled SCA on the ASCAD with random keys dataset with CNNs.

same adversary executes a random search and does not possess an optimal neural network model. In this section, we consider state-of-the-art models from [9] and [27], which provide carefully tuned CNN models for the ASCAD dataset. This way, an adversary is considered strong from both side-channel and deep learning perspectives. As the hyperparameters are already chosen, we again run 50 models for each fixed number of profiling traces by only randomly varying the batch size (from 50 to 400, with steps of 50 traces).

Figure 6 provides the results for the *cnn_architecture* [32] proposed in [9] for the Hamming weight and Identity leakage models (for the HW leakage model, we use the same learning model as for the Identity leakage model, but we set the number of output classes to 9). The framework indicates that increasing the number of profiling traces is not very relevant when possessing an "optimal" profiling model. Indeed, in Figure 6b, the best results are achieved for 30 000 profiling traces, and adding more profiling traces increases training time and does not improve attack results. In this example, we observe with a real-world dataset that GE< 20 can be achieved with a single attack trace.

The Identity leakage model results from Figure 6b indicate one more interesting phenomenon, which is, to the best of our knowledge, not before reported in deep learning-based SCA. We can notice for one model setting the behavior called deep double descent [33]. This behavior describes a phenomenon where the test loss first decreases with the increase in the architecture size. Then, the loss starts to increase and finally decreases again. When the loss increases, this is connected with an effect called "sample-wise non-monotonicity". Interestingly, this effect describes a behavior where more training traces damages the test phase's performance. While there is no definitive answer to why this behavior happens, one explanation could be that the model does not have enough capacity to fit the data. Adding more data requires the model to drastically "change" its parameters, improving attack performance.



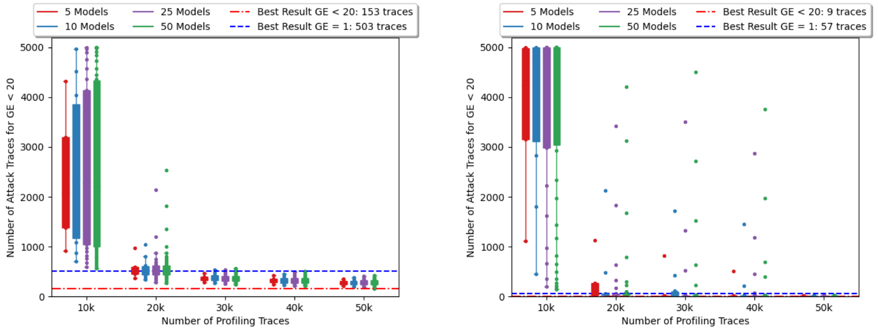
(a) CNN from [32], HW leakage model. (b) CNN from [32], Identity leakage model.

Fig. 6: Profiled SCA on the ASCAD with fixed keys dataset with CNN architecture from [9].

Figure 7 shows results for *noConv1_ascad_desync_0* [34] proposed in [27]. As this neural network architecture is an optimization built on top of [32], results for the ASCAD fixed key dataset indicate an even smaller minimum number of profiling traces to reach successful results, which is 20 000 profiling traces. Nevertheless, we can also observe the differences in model performance with the Efficient Attacker Framework when selecting different leakage models.

4.4 General Observations

On a general level, while not the core research point in this work, we note that the Identity leakage model requires fewer attack traces to reach GE < 20, which is expected. MLP exhibits better performance than CNN for a smaller number of profiling traces, which is again in line with related works. It is important



(a) CNN from [34], HW leakage model. (b) CNN from [34], Identity leakage model.

Fig. 7: Profiled SCA on the ASCAD with fixed keys dataset with CNN architecture from [27].

to observe how the learnability constraint directly influences the required combination of the number of profiling and attack traces to reach a low guessing entropy. Moreover, one can choose a trade-off between profiling traces N and attack traces Q while still performing a successful attack.

While our framework aims to find the minimal number of profiling traces and keep the number of tuning experiments to mount a successful attack as low as possible, we never state what those numbers should be. Indeed, the experiments showcase radically different behaviors for various numbers of profiling and attack traces (coupled with the influence of the number of tuning experiments). Providing actual values makes sense only when the whole experimental environment is considered (datasets, algorithms, environmental settings) and, even more importantly, when one compares experiments on the same targets but with different settings. All our experiments strongly confirm that the number of profiling traces and the number of experiments (complexity) play a paramount role and should be included in proper performance analysis for deep learning-based SCA.

4.5 Advantages of the Efficient Attacker Framework

Usually, an attacker is expected to make use of the maximum possible number of profiling traces to build a model (templates, deep neural networks). Similarly, the number of attack traces tends to be maximized to better estimate the model exploitation capability. In cases when the learning model is inefficient (i.e., unable to fit the existing leakage) and all available side-channel measurements are used, the attacker or evaluator has a limited view of what component has a significant impact on the attack results, which can lead to overestimating the security of the target.

In this case, the reference metric would be the guessing entropy of a single experiment, which says nothing about the influence of the number of measurements and tuning experiments on the security of the assessed target. Therefore, the Efficient Attacker Framework usage provides a better representation of the influence of the number of profiling traces, attack traces, and tuning experiments. We analyze an attack's efficiency with GE < 20 as a reference metric. Of course, the framework can be adapted to any metric that describes the attack's efficiency, such as success rate, or extended to more dimensions that may influence the attacker's strength, for example, by including resource requirements. While the benefits of depicting the results with our framework are evident, one can ask whether we lost some information when compared to the traditional result depiction.We claim this not to be true due to two reasons. First, all relevant information is kept so the attacker can still depict traditional results. Second, once the appropriate performance level is set (e.g., guessing entropy value equal to δ), it is less relevant to observe how that value is reached (as values above the threshold are out of the attacker's reach).

As a common scenario for deep learning side-channel evaluation, our experiments concentrated on the concept of divide-and-conquer strategies for symmetric ciphers. However, the Efficient Attacker Framework is not limited to this scenario, and depending on the threat model of the attack, the framework can be extended, for example, to rank estimation strategies [35] or even to recursive recovering strategies like Extend and Prune (EP) [36]. Instead of using metrics on subkey bytes, an evaluator would choose a rank estimation strategy and depict the number of attack traces to reach a certain estimated rank within the complete keyspace as a performance metric. As in our experiments, this may be evaluated in terms of the number of training traces. Naturally, the Efficient Attacker Framework would allow us to compare different rank estimation strategies. EP techniques are required when estimating models for the entire keyspace is not feasible as for many asymmetric ciphers. While the estimation of key recovery differs, the application of the Efficient Attacker Framework is similar. Depending on the chosen cryptographic primitive, an evaluator could again depict the minimum number of traces in the attack phase, depending on the amount of information (bits or chunks of information).

Some former works also attempted to make the most out of the available information contained within a trace. For instance, soft analytical side-channel analysis [37] aims at leveraging the information collected at different steps in one round (e.g., for AES: `AddRoundKey`, `SubBytes`, `MixColumns`, etc.), and even beyond, from round to round. For such constructive information gathering to occur, the whole secret shall be guessed at once. Belief-propagation algorithms can be used in this respect (to relate all leakage points of interest). However, we notice that such a technique is mostly profitable to exploit as much as possible the online captured side-channel, whereas the scope of our paper is to optimize the usage of the data collected from the learning device.

5 Conclusions

This paper discusses how to evaluate attacks when considering the profiled side-channel analysis. We argue that considering only an unbounded attacker can negatively affect how side-channel analysis is performed while not being realistic. We propose a new framework, denoted as the Efficient Attacker Framework, where we explore the number of measurements and hyperparameter tuning experiments required in the profiling phase such that the attacker is still successful.

We consider our new framework more realistic but also more adept for experimental evaluations since it allows us to compare different results in a more unified way. In particular, our framework will hopefully trigger more research relevant not only for academia but also for evaluation labs. Finally, our framework is relevant beyond profiled side-channel analysis and can be used in any supervised learning setting.

References

1. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer (December 2006) ISBN 0-387-30857-1, `http://www.dpabook.org/`.
2. 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.
3. 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
4. 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
5. 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: COSADE 2015, Berlin, Germany, 2015. Revised Selected Papers. (2015) 20–33
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 Trans. Cryptogr. Hardw. Embed. Syst. **2019**(1) (2019) 209–237
7. 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
8. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(3) (May 2019) 148–179
9. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(1) (Nov. 2019) 1–36

10. Wu, L., Perin, G., Picek, S.: I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/1293 (2020) `https://eprint.iacr.org/2020/1293`.
11. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(3) (Jul. 2021) 677–707
12. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., 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, The Internet Society (2020)
13. Standaert, F.X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: EUROCRYPT. Volume 5479 of LNCS., Springer (April 26-30 2009) 443–461 Cologne, Germany.
14. Whitnall, C., Oswald, E.: A Fair Evaluation Framework for Comparing Side-Channel Distinguishers. J. Cryptographic Engineering **1**(2) (2011) 145–160
15. Whitnall, C., Oswald, E.: A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework. In Rogaway, P., ed.: CRYPTO. Volume 6841 of Lecture Notes in Computer Science., Springer (2011) 316–334
16. Guilley, S., Heuser, A., Rioul, O.: A Key to Success - Success Exponents for Side-Channel Distinguishers. In Biryukov, A., Goyal, V., eds.: Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings. Volume 9462 of Lecture Notes in Computer Science., Springer (2015) 270–290
17. Duc, A., Faust, S., Standaert, F.: Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Oswald, E., Fischlin, M., eds.: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Volume 9056 of Lecture Notes in Computer Science., Springer (2015) 401–429
18. Bronchain, O., Hendrickx, J.M., Massart, C., Olshevsky, A., Standaert, F.: Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Boldyreva, A., Micciancio, D., eds.: Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I. Volume 11692 of Lecture Notes in Computer Science., Springer (2019) 713–737
19. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
20. publication 140-3, N.F.F.I.P.S.: Security Requirements for Cryptographic Modules (Draft, Revised). (09/11 2009) 63 `http://csrc.nist.gov/groups/ST/FIPS140_3/`.
21. ISO/IEC JTC 1/SC 27 IT Security techniques: ISO/IEC 17825:2016 Information technology – Security techniques – Testing methods for the mitigation of non-invasive attack classes against cryptographic modules (January 2016) `https://www.iso.org/standard/60612.html`.
22. ISO/IEC JTC 1/SC 27 IT Security techniques: ISO/IEC 15408-1:2009 Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model (January 2014) `https://www.iso.org/standard/50341.html`.

23. Common Criteria: Supporting Document Mandatory Technical Document Application of Attack Potential to Smartcards (2013) `https://www.commoncriteriaportal.org/files/supdocs/CCDB-2013-05-002.pdf`.
24. Heuser, A., Kasper, M., Schindler, W., Stöttinger, M.: A New Difference Method for Side-Channel Analysis with High-Dimensional Leakage Models. In Dunkelman, O., ed.: CT-RSA. Volume 7178 of Lecture Notes in Computer Science., Springer (2012) 365–382
25. Cao, Y., Zhou, Y., Yu, Z.: On the negative effects of trend noise and its applications in side-channel cryptanalysis. Chinese J. Electron. **23** (2014) 366–370
26. TELECOM ParisTech SEN research group: DPA Contest (1st edition) (2008–2009) `http://www.DPAcontest.org/`.
27. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3) (Jun. 2020) 147–168
28. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.X.: An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In Knudsen, L.R., Wu, H., eds.: Selected Areas in Cryptography. Volume 7707 of Lecture Notes in Computer Science., Springer (2012) 390–406
29. Common Criteria Management Board: Common Methodology for Information Technology Security Evaluation Evaluation methodology, Version 3.1, Revision 4, CCMB-2012-09-004 (September 2012) `https://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R4.pdf`.
30. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptographic Engineering **10**(2) (2020) 163–188
31. Perin, G., Chmielewski, L., Picek, S.: Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(4) (Aug. 2020) 337–364
32. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN-architectures in SCA. `https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA/blob/master/ASCAD/N0%3D0/cnn_architecture.py` (2019) [Online; accessed 20-June-2021].
33. Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., Sutskever, I.: Deep double descent: Where bigger models and more data hurt (2019)
34. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks. `https://github.com/KULeuven-COSIC/TCHES20V3_CNN_SCA/blob/master/src/models.py` (2020) [Online; accessed 20-June-2021].
35. Veyrat-Charvillon, N., Gérard, B., Standaert, F.X.: Security evaluations beyond computing power. In Johansson, T., Nguyen, P.Q., eds.: Advances in Cryptology – EUROCRYPT 2013, Berlin, Heidelberg, Springer Berlin Heidelberg (2013) 126–141
36. 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.
37. Veyrat-Charvillon, N., Gérard, B., Standaert, F.: Soft analytical side-channel attacks. In Sarkar, P., Iwata, T., eds.: Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Volume 8873 of Lecture Notes in Computer Science., Springer (2014) 282–296