On the Evaluation of Deep Learning-based Side-channel Analysis

Lichao Wu[1], Guilherme Perin[1], and Stjepan Picek[1]

Delft University of Technology, The Netherlands

Abstract. Deep learning-based side-channel analysis already became a de-facto standard when investigating the most powerful profiling side-channel analysis. The results from the last few years show that deep learning techniques can efficiently break targets that are even protected with countermeasures. While there are constant improvements in making the deep learning-based attacks more powerful, little is done on evaluating such attacks' performance. Indeed, what is done today is not different from what was done more than a decade ago.
This paper considers how to evaluate deep learning-based side-channel analysis and whether the commonly used techniques give the best results. To that end, we consider different summary statistics and the influence of algorithmic randomness on the stability of profiling models. Our results show that besides commonly used metrics like guessing entropy, one should also show the standard deviation results to assess the attack performance properly. Our results show that using the arithmetic mean for guessing entropy does not yield the best results, and instead, a geometric mean should be used.

Keywords: Side-channel analysis, Deep Learning, Evaluation, Median

1 Introduction

Side-channel analysis (SCA) encompasses techniques aiming at exploiting weaknesses of algorithms' implementations [11]. One common division of SCA is into direct attacks and profiling attacks. Profiling attacks are more powerful but also require a stronger attacker who has access to a copy of a device to be attacked. The attacker uses that copy to build a model of a device that can be used to attack another similar (identical) device. Since there are two phases to run a profiling attack (profiling phase and attack phase), such attacks are also called two-stage attacks. In the last few years, the most explored profiling attacks are based on machine learning (especially deep learning). Such attacks are very powerful as they can break targets protected with countermeasures [2,7] but also somewhat "easier" to deploy as they do not require a pre-processing stage. Still, there are many open questions that are usually connected with how to find machine learning architectures that perform well [30,26]. Unfortunately, this is just one side of the problem. A perspective that cannot be neglected is how to

assess the performance of such a profiling model. While the state-of-the-art in deep learning attacks progressed tremendously in the last few years, no results consider how to evaluate the performance of deep learning-based SCA.

It is common to use metrics like key rank, success rate, and guessing entropy to evaluate the attack performance in SCA. While the first one is simple, the latter two are run multiple times to counteract the effect of dataset/measurements selection. For direct attacks (e.g., CPA) or simpler profiling attacks like the template attack, this repetition is sufficient as the algorithms are deterministic, so running them multiple times gives the same results (if the measurements do not change). On the other hand, deep learning techniques (i.e., artificial neural networks) have multiple sources of randomness (randomness due to the initialization, randomness due to the regularization, randomness due to the optimization procedure), making those algorithms stochastic. As such, it is common to expect different results if running those algorithms multiple times. This makes the evaluation of the performance of such algorithms more challenging. The problem becomes even more challenging if accounting for the differences among various neural network architectures.

To the best of our knowledge, there are not many works assessing the evaluation performance of side-channel attacks. Martin et al. investigated how to estimate key rank distribution for SCA [12]. Interestingly, they concluded that the guessing entropy should be easier to estimate through the arithmetic mean. What is more, the authors concluded that repeated experiments are needed for a stable estimate. Whitnall and Oswald considered robust profiling setting [25], which can also be connected with the stability of a profiling model, as intuitively, a more robust profiling model also provides more stability. Picek et al. considered the robustness issue through the expectation estimation problem and provided strong theoretical foundations to assess the robustness of deep learning-based SCA [20]. The authors concluded that deep learning algorithms are quite robust, but they did not consider how to improve the evaluation process.

This paper investigates how to evaluate the attack performance of deep learning-based SCA. Our main contributions are:

- We investigate the influence of algorithmic randomness on the attack performance. More precisely, we use the standard deviation to showcase that running experiments multiple times can result in a significantly different assessment of the attack performance. This is confirmed for scenarios using different random models and when using the same profiling model and training it independently several times.
- We investigate the most appropriate summary statistic for the evaluation of the attack performance. We consider the arithmetic mean, geometric mean, and median and show that the median is the most appropriate metric. Furthermore, our results indicate that deep learning-based SCA often results in skewed distributions of the attack performance, so the arithmetic mean is not appropriate statistics, which is highly relevant as it is commonly used.
- We investigate how a different number of independent experiments in the attack phase influences attack performance. Our results show that this value

does not play a significant influence, so much smaller values can be safely used.

We conduct an extensive experimental evaluation including three datasets, two leakage models, and different types of neural networks to confirm our observations.

2 Preliminaries

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

A dataset represents a collection of side-channel measurements denoted as $\mathbf{T}$, with each trace $\mathbf{t}_i$ associated with an input value (plaintext or ciphertext) $\mathbf{d}_i$ and a key $\mathbf{k}_i$. The dataset is commonly divided into a training set of size N, a validation set of size V, and an attack set of size Q. We denote with k a key candidate that takes its value from the keyspace $\mathcal{K}$, while k^* represents the correct key.

Finally, $\boldsymbol{\theta}$ denotes the vector of parameters learned in a profiling model (e.g., the weights in neural networks). Finally, $\mathcal{H}$ denotes the set of hyperparameters defining the profiling model.

2.1 Machine Learning-based Side-channel Analysis

In the rest of this work, we concentrate on supervised machine learning and the multi-class classification task, as commonly done in related works (see Section 3). Supervised machine (deep) learning classification represents the machine learning task of learning a function f that maps an input to the discrete output ($f : \mathcal{X} \rightarrow Y$)) based on examples of input-output pairs. We consider the multi-classification task (thus, with more than two classes), where c denotes the number of classes. The function f is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$, where n denotes the number of trainable parameters.

Training. In the training phase, the goal is that the algorithm learns the parameters $\boldsymbol{\theta}'$ minimizing the empirical risk represented by a loss function L on a dataset of size N:

$$\boldsymbol{\theta}' = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} L(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i). \tag{1}$$

Validation. When training a profiling model, we want it to generalize well to previously unseen data. Alternatively, this could be stated as the goal of having a profiling model that shows stability. There, it is common to use cross-validation techniques. Cross-validation is a statistical validation technique used to assess the performance of a machine learning model. It uses a subset of the dataset to train a model and a complementary subset of the dataset not used

for training to assess the model performance. Two commonly employed cross-validation techniques in the machine learning-based SCA are validation and k-fold cross-validation.

With the validation technique, we divide the dataset into training, validation, and test dataset and use the validation dataset to assess the performance of a model trained on the training dataset. Finally, we use the best-obtained model to attack the test dataset. While this technique is simple, it will suffer from a large variance caused by different measurements in disjoint datasets. This technique is commonly used with deep learning-based SCA.

In the k-fold cross-validation, a dataset is divided into k parts. Then, a model is built on $k-1$ folds and evaluated on the k-th fold. This is repeated until each fold serves as the k-th fold (thus, every combination of $k-1$ folds serves to train the model). The choice of k commonly follows the bias-variance trade-off. This technique is commonly used with computationally simpler machine learning techniques.

Testing. In the test phase, the goal is to predict classes (or probabilities that a certain class would be predicted) y based on the previously unseen traces $\mathbf{x}$ (the number of traces equals Q), and the trained model f.

Evaluating the Attack Performance. The outcome of predicting with a model f on the attack set is a two-dimensional matrix P with dimensions equal to $Q \times c$. The probability $S(k)$ for any key byte candidate k is a valid SCA distinguisher, where it is common to use the maximum log-likelihood distinguisher:

$$S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,v}). \tag{2}$$

The value $\mathbf{p}_{i,v}$ denotes the probability that for a key k and input d_i, we obtain the class v. The class v is derived from the key and input through a cryptographic function CF and a leakage model l.

It is common to estimate the effort to obtain the secret key k^* with metrics like success rate (SR) and guessing entropy (GE) from the predictions. Guessing entropy was first defined by Massey, and it represents the expected number of guesses (using an optimal strategy) to correctly guess the value of a random variable [13]. Later, Standaert et al. connected those results with SCA and key rank as the number of guesses an optimal adversary would need to guess the secret key [24].

With Q traces in the attack phase, an attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability where g_1 denotes the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate. Then, guessing entropy is the average position of k^* in $\mathbf{g}$ [1]. The success rate of order o is the average empirical

[1] Averaging is commonly done over 100 independent experiments (attacks) to obtain statistically significant results.

probability that the secret key k^* is located within the first o elements of the key guessing vector $\mathbf{g}$.

Sources of Randomness in Deep Learning-based SCA. When considering deep learning, there are several common sources of randomness that will influence the obtained results. Informally, we can divide those sources into those connected with the dataset (dataset randomness) and those connected with the machine learning algorithm (algorithmic randomness). The randomness connected with datasets is caused by the random selection of the traces to be included in the training/attack dataset. Averaging multiple results is a common way how to reduce the effect of any specific traces. While the choice of measurements can significantly influence the results, we consider it out of scope for this paper, as it influences any side-channel attack and not only the deep learning ones. For more results about attack performance when selecting different traces, we refer interested readers to [29].

Considering the randomness of machine learning algorithms, we will obtain different results even if training/evaluating a neural network on the same set of traces. The common sources of randomness are:

- random initialization of weights and biases,
- randomness in regularization techniques like Dropout,
- randomness in optimization techniques (used to minimize the loss function and, consequently, improve neural networks' performance).

Finally, we can also consider the randomness stemming from the hyperparameter tuning. Since most deep learning-based SCA uses random search to find good hyperparameters, we can expect (radically) different evaluation results based on the used architectures. While there are already results showing that these sources of randomness play a significant role in deep learning-based SCA (as discussed in Section 3), there is no discussion on how to resolve such issues or at least report the results in a more meaningful way.

3 Related Works

The last decade brought significant advances to the profiling side-channel analysis domain. A plethora of good results, first with simpler machine learning techniques and later with deep learning, showed the full potential of such techniques, even when considering protected targets. At the same time, the variety of techniques and choices one could take when using machine learning (or even more deep learning) brought the need for much more detailed analyses, resulting in an abundance of results and papers.

The first developed profiling SCA techniques like template attack [4] or stochastic models [22] have no hyperparameters to tune, making the analysis simpler and deterministic. Then, running the experiments multiple times results in the same solutions, provided that the same measurements are used. While there is no algorithmic randomness for the template attack, there is dataset randomness. Besides selecting the traces to be used in the profiling and attack

phase, it is common to employ a feature selection/engineering before running the attack. This step will influence the final attack performance. Note that the feature selection/engineering step is also a common one for simpler machine learning techniques.

Afterward, machine learning techniques like support vector machines [19], random forest [8], or Naive Bayes [17,6] started to attract more attention in the SCA community as the results were in general favorable compared to template attack. Those techniques have a different number of hyperparameters (expect Naive Bayes that has no hyperparameters) one needs to tune to reach their full potential. Coupled with the complexity stemming from selecting data for training, this can significantly increase the number of required experiments (and thus, time). Still, the evaluation of the attack performance did not account for the algorithmic sources of randomness, and the SCA community continued to report the results in the same fashion as for the template attack (e.g., the average key rank for a specific number of attack traces).

Finally, in the last few years, we see a trend of using deep learning for profiling SCA. The first significant step was done by Maghrebi et al. as they showed that CNNs could efficiently break targets [10]. Additionally, they showed that deep learning works well with raw traces, removing the need for various feature selection techniques [16]. Cagli et al. demonstrated that deep learning could also break implementations protected with jitter countermeasures and introduced the concept of data augmentation in the profiling SCA [3]. Picek et al. evaluated several machine learning metrics and showed a discrepancy between those and the side-channel metrics [18]. The authors showed that the metrics problems also happen for deep learning techniques. Kim et al. designed a deep learning architecture capable of achieving excellent results on several publicly available datasets [7]. To further improve the attack performance, the authors regularized input with Gaussian noise.

Benadjila et al. provided an investigation into the importance of hyperparameter tuning [1]. Zaid et al. proposed the first methodology to select hyperparameters related to the size (number of learnable parameters, i.e., weights and biases) of layers in CNNs [30]. Wouters et al. [26] improved upon the work from Zaid et al. [30] where they showed how to reach similar attack performance with significantly smaller neural network architectures. Perin et al. investigated deep learning model generalization and showed that output class probabilities represent a strong SCA metric [14]. Wu et al. introduced Bayesian optimization for hyperparameter tuning [27]. With this approach, the authors managed to find small neural network architectures that perform well (surpassing the architectures' performance obtained by the previous methodologies). Rijsdijk et al. used reinforcement learning to find small convolutional neural networks that perform well, surpassing the previous state-of-the-art [21]. *These works showed the importance of hyperparameter tuning but did not consider the influence of algorithmic randomness. What is more, the ever-increasing number of hyperparameters to test resulted in a simpler (faster) validation process but also a larger variance among results.*

Li et al. investigated the influence of randomness caused by the weight initialization for MLP and CNN architectures and showed that, depending on the choice of the weight initialization method, SCA attack performance could vary significantly [9]. Perin and Picek explored the impact of the optimizer choice for deep learning-based SCA [15]. Their results indicated that some commonly used optimizers could easily overfit, thus requiring more effort during the training process.

All these works have in common that they do not question whether the estimation of the attack performance can be improved and if the arithmetic mean is the best choice for estimating the attack.

4 Experimental Setup

4.1 Datasets

We consider three datasets: two versions of the ASCAD dataset and the CHES_CTF dataset. The first two datasets are obtained from an 8-bit AVR microcontroller running a masked AES-128 implementation, where the side-channel is electromagnetic emanation [1]. The first version of the dataset comprises 50 000 traces for profiling and 10 000 for the test. Commonly, researchers attack the first masked byte, which is key byte three, and for it, the authors provided the pre-selected window of 700 features. This dataset has the same key for both training and test sets. We denote this dataset as ASCAD_F. This dataset is available at `https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key`.

The second version of the dataset is obtained from the same target as the first one but has random keys in the profiling set, which has 200 000 traces, and a fixed key in the test set that has 100 000 traces. The pre-selected window to attack the first masked key byte (key byte three) has 1 400 features. We denote this dataset as ASCAD_R. This dataset is available at `https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key`.

Finally, we investigate a dataset obtained from a 32-bit STM microcontroller running a masked AES-128 implementation. There are 45 000 traces for the training set [2], which contains a fixed key. The attack set consists of 5 000 traces. The key used in the training and validation set is different from the key configured for the test set. Each trace consists of 2 200 features. This dataset is available at `https://chesctf.riscure.com/2018/news`.

4.2 Leakage Models

We investigate two leakage models:

1. The Hamming weight (HW) leakage model: the attacker assumes the leakage proportional to the sensitive variable's Hamming weight.
2. The Identity (ID) leakage model: the attacker considers the leakage in the form of an intermediate value of the cipher.

[2] At the moment, there are only 10 000 traces available to download

4.3 Machine Learning Algorithms

Our experiments include two commonly used neural network types used in the profiling side-channel analysis. The first one is the multilayer perceptron (MLP), and the second one is the convolutional neural network (CNN).

Multilayer Perceptron. The multilayer perceptron is a feed-forward neural network mapping input sets onto sets of appropriate outputs. MLP consists of multiple layers of nodes in a directed graph, where each layer is fully connected to the next one, and training of the network is done with the backpropagation algorithm [5].

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

4.4 Environment

The machine learning model was implemented in python version 3.6, using TensorFlow library version 2.0. The model training algorithms were run on a cluster made out of Nvidia GTX 1080 and GTX 2080 graphics processing units (GPUs), managed by Slurm workload manager version 19.05.4.

4.5 Summary Statistics

Once we obtained the information bout key rank from z independent experiments over space $\mathcal{S}$, we need to find the most appropriate estimator for the expected value of $\mathcal{S}$. A common way to do this is to use the arithmetic mean, where the arithmetic mean of z examples equals

$$\overline{x} = \frac{1}{z} \sum_{i=1}^{z} x_i. \tag{3}$$

While a common way to calculate guessing entropy, arithmetic mean has a drawback as it is dominated by numbers on a larger scale. This happens due to a simple additive relationship between numbers where scales do not play a role.

An alternative to arithmetic mean that takes into account the proportions is the geometric mean:

$$\check{x} = \left(\prod_{i=1}^{z} x_i \right)^{\frac{1}{z}}. \tag{4}$$

We can also consider the middle value of the dataset, which is called median:

$$\tilde{x} = \frac{x_{\frac{z}{2}} + x_{\frac{z}{2}+1}}{2}. \tag{5}$$

The median is less affected by outliers and skewed data compared to the arithmetic mean.

The standard deviation is a measure of the amount of variation or dispersion of a set of values. In the SCA context, a large standard deviation means that the adversary will have a high probability to be "lucky" (or "unlucky") in the choice of measurements or hyperparameters.

$$\sigma_x = \sqrt{\frac{1}{z}\sum_{i=1}^{z}(x_i - \overline{x})^2}. \tag{6}$$

5 Experimental Results

For the ASCAD_F and ASCAD_R datasets, we used 50,000 traces for profiling, while for CHES_CTF, we used 10 000 profiling traces. The attack set size is set to 5 000 for all three datasets. We investigate two settings in our experiments: in the first one, we consider random profiling models, while in the second one, we use state-of-the-art profiling models from related works.

The number of random profiling models is set to 100 for all experiments. We set the maximum size for architectures for the random model generation to the ones from the ASCAD paper, which we denote as 'MLP_best' and 'CNN_best'. Aligned with the optimal settings provided by the ASCAD paper, we use RMSProb as the optimizer with a learning rate of 1e-5. The training epoch number is set to 75. The detailed model implementation are listed in Table 1. To generate the random models from the base model (MLP_best and CNN_best), for CNN models, we randomized the kernel size of the convolution layer and the number of neurons in the dense layer. The latter one is also randomized in MLP models. Specifically, the range is from the $half$ of the original parameter to the original parameter. For instance, the kernel range of the first convolution layer in the CNN model is from 32 to 64. For MLP, the range of the neurons is from 100 to 200. We aim to provide diverse architectures, but they should still perform relatively well as they are based on well-performing architectures that we do not change radically.

Test models	Convolution (filter_number, size)	Pooling (size, stride)	Dense layer	Activation
MLP_best	-	-	200*5	ReLU
CNN_best	Conv (64, 128, 256, 512, 512)	avg(2,2)*5	4 096*2	ReLU

Table 1: Base MLP and CNN architectures used in the experiments.

In terms of attacks with the state-of-the-art models, we used the MLP models obtained through the Bayesian Optimization [28]. The implementation details are listed in Table 1. The CNN models we used are developed with the reinforcement learning approach [21]. The details about the architectures are listed in Tables 2 and 3. All of the training hyperparameters are aligned with the original papers [28,21]. Specifically, the learning rate of CNNs is handled by OneCycleLR policy [23] with the maximum learning rate of 5e-3. While there are other state-of-the-art models we could use (e.g., from [30,26]), we opted for these as the related works did not run experiments for the HW leakage model but only the identity leakage model. Finally, to reduce the effects of algorithmic randomness (initialization, regularization, optimization), we train each model 20 times and average the results.

Test models	Dense layer	Activation	lr
$ASCAD\_F_{HW}$	1 024, 1 024, 760, 8, 704, 1 016, 560	ReLU	1e-5
$ASCAD\_F_{ID}$	480,480	ELU	5e-3
$ASCAD\_R_{HW}$	448, 448, 512, 168	ELU	5e-4
$ASCAD\_R_{ID}$	664, 664, 624, 816, 624	ELU	5e-4
$CHES\_CTF_{HW}$	192, 192, 616, 248, 440	ELU	1e-3

Table 2: MLP architectures used in the experiments [28].

Test models	Convolution (filter_number, size)	Pooling (size, stride)	Dense layer	Activation
$ASCAD\_F_{HW}$	Conv(16,100)	avg(25,25)	15+4+4	selu
$ASCAD\_F_{ID}$	Conv(128,25)	avg(25,25)	20+15	selu
$ASCAD\_R_{HW}$	Conv(4, 50)	avg(25, 25)	30+30+30	selu
$ASCAD\_R_{ID}$	Conv(128, 3)	avg(75, 75)	30+2	selu
$CHES\_CTF_{HW}$	Conv(4, 100)	avg(4, 4)	15+10+10	selu

Table 3: CNN architectures used in the experiments [21].

In all the experiments, we conduct the following steps to obtain the results:

1. For each of the profiling models (20 for state-of-the-art and 100 for random models), we calculate summary statistics (arithmetic mean, geometric mean, median) for evaluation metrics (GE, SR) for a different number of attacks (independent experiments).
2. Once we obtain results summary statistics for all experiments, we average them (arithmetic mean).
3. We plot the average and standard deviation over experiments. Note that since we average over profiling models, we show the influence of algorithmic

randomness and not dataset randomness (as then, we should show standard deviation over a number of attacks).
4. Since all of the models effectively retrieve the key or converge to close to zero guessing entropy, we use T_{GE0} to evaluate the attack result (i.e., the number of attack traces to reach GE of zero).

Moreover, we also investigated the success rate but observed it commonly does not change regardless of the averaging methods and thus offers limited information. Therefore, we omit these results and only present the success rate results that contain more information.

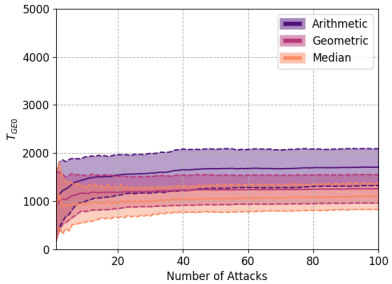
5.1 Results for the ASCAD_F Dataset

The results for random models are shown in Figure 1. The solid lines represent the average of the T_{GE0} metric (i.e., arithmetic mean, geometric mean, or median), while the dashed lines of the same color indicate the upper and lower bound of the standard deviation ($\pm\ \sigma$). The spaces in-between are filled with the corresponding but lighter color. First, one can observe all the results to indicate rather stable behavior: the median is a statistic indicating the best attack performance while the worst is the arithmetic mean. Interestingly, we can observe that the upper deviation value for the median gives similar results as the lower deviation value for the arithmetic mean, indicating that the median is a significantly better evaluation statistic. The differences in the number of attack traces are also significant: from around 700 to 2 000 attack traces. These results indicate that if using the arithmetic mean (and to a smaller extent, geometric mean), it is easy to get outlier results, skewing the distribution.

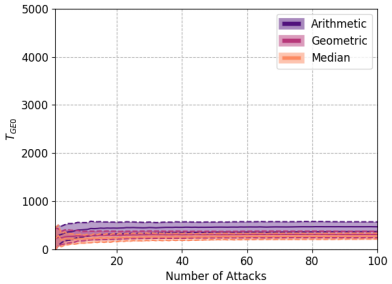
Next, the behavior for a different number of attacks remains stable where we do not see differences for more than 40 attacks. This indicates that averaging 100 times, as commonly done in the literature, does not bring significant advantages, so this can be potentially reduced to make the attack evaluation faster. Finally, for all three averaging methods, the standard deviation results are comparable regardless of the number of attacks. This indicates that random models indeed perform well for this dataset and that more elaborate tuning mechanisms are not needed (a similar conclusion is given in [28]. Interestingly, MLP for the ID leakage model shows the best results and smallest standard deviation. We postulate that this happens as the model's capacity is well aligned with the difficulty of the dataset, so most of the experiments end up with a rather similar attack performance.

We also show averaged success rate results in Figure 2. The rest of the results are omitted as the success rate results are the same for the three averaging methods. Interestingly, we see a drop for both geometric mean and arithmetic mean with more attack results being averaged, while the median remains stable. This indicates that the influence of outliers when considering more attacks becomes more significant, as it skews the distribution.

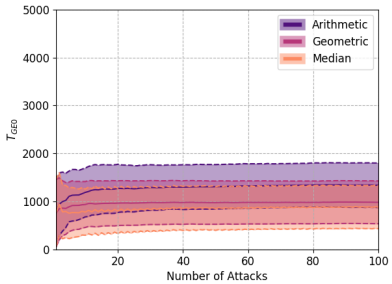
Next, we investigate the performance of four state-of-the-art models. The results are shown in Figure 3. The green dashed line represents the attack performance reported in the original papers [28,21]. For MLP, observe that the



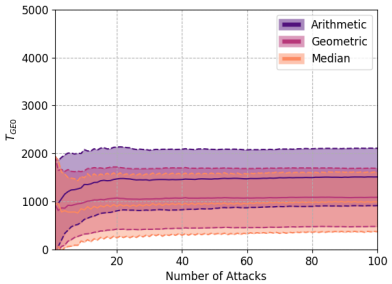
(a) Random MLP with the HW leakage model.



(b) Random MLP with the ID leakage model.

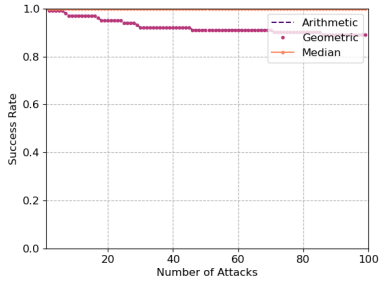


(c) Random CNN with the HW leakage model.

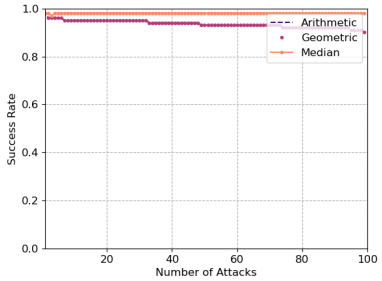


(d) Random CNN with the ID leakage model.

Fig. 1: T_{GE0}: attack on ASCAD_F with random MLP and CNN models.



(a) Random MLP with the HW leakage model.



(b) Random CNN with the ID leakage model.

Fig. 2: Success Rate: attack on ASCAD_F with random MLP and CNN models.

results as reported in related works are better. This indicates that the models are not very stable, and to reliably report their performance, averaging is a mandatory step. What is more, we see that getting results on the level of those reported in related works requires a significant number of experiments (until

the appropriate weights of a model are found). Large standard deviation values also confirm this. We can again observe that the median gives the best results, while the arithmetic mean indicates significantly worse behavior (around twice as many traces required to reach GE of zero). For all the cases except MLP for the ID leakage model, we also see that the increased number of attacks does not indicate differences in the performance.

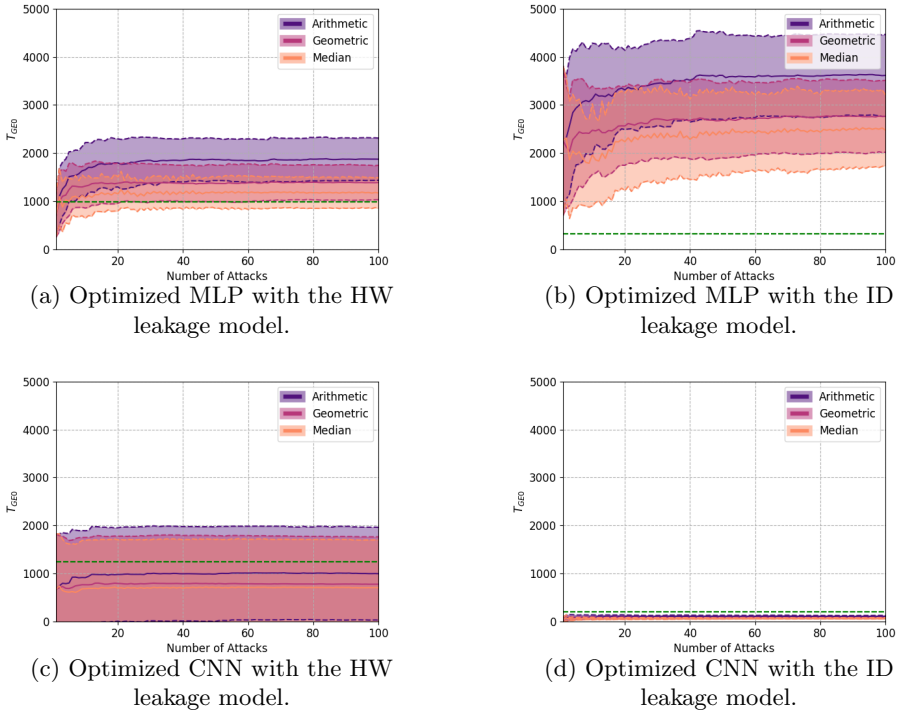
For CNN results, the median performs the best, which is aligned with the previous results. The number of attacks shows only a marginal influence, and the deviation is large for the HW leakage model while small for the ID leakage model. We postulate this happens as with fewer classes scenario (as it is for the HW leakage model), the profiling model has more capacity (recall that these optimized models are already quite small from the perspective of the number of trainable parameters) and more choice to end up with different performing architectures. The model capacity seems better aligned with the task for the ID leakage model, so most of the experiments end up with similar attack performance. Interestingly, we observe that we can reach an even better performance than reported in related works. We believe this happens as we (in essence) show results for ensembles of classifiers, which is reported to work better than a single classifier [14].

Finally, a large standard deviation is expected for random models, as we effectively train different profiling models, so it is reasonable to expect some to perform poorly. Still, we see significant deviation even when using a single optimized model, which indicates that 1) those models are still too large for the task, and 2) reporting the attack performance for a single setup can be very misleading.

5.2 Results for the ASCAD_R Dataset

Next, we perform attacks on the ASCAD_R dataset. Recall that the profiling traces for this dataset contain random keys while the attack set contains a fixed but unknown key. This setting is closer to the real attack scenario, but it increases the difficulties in retrieving the correct key from the attack set. Figure 4 presents the attack results for 100 random models. Compares with ASCAD_F, we see performance degradation, especially when attacking the ID leakage model. For instance, when attacking with random MLP for the ID leakage model, 74% of the models failed to converge GE to zero within 5 000 attack traces. Still, even in the worst attack cases, we see median reliability representing the attack result and requiring the smallest number of attack traces to obtain the correct key. Aligned with the previous results, we observe a limited influence of the number of attacks, while standard deviation is large for all cases except one (MLP with the ID leakage model). This indicates that many models perform poorly and n be optimized (which is not surprising as we select them randomly and this dataset is more difficult than the previous one).

Aligned with the previous experiment, there is a significant drop in success rate when the number of attacks increases, indicating the influence of outliers. In



(a) Optimized MLP with the HW leakage model.

(b) Optimized MLP with the ID leakage model.

(c) Optimized CNN with the HW leakage model.

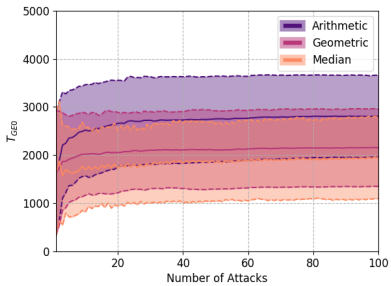
(d) Optimized CNN with the ID leakage model.

Fig. 3: T_{GE0}: attack on optimized MLP and CNN models with the ASCAD_F dataset.

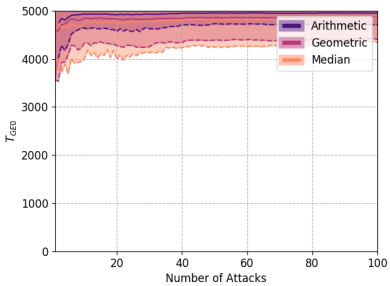
all scenarios, the median reaches the highest success rate of all tested averaging methods.

Moving to the results for the state-of-the-art models, as shown in Figure 5, we see the attack performance is significantly improved compared to the previous result on random models. This means that using random models will not suffice to reach the top attack performance due to a more difficult dataset. Again, we see median performs the best, consistently indicating the superiority of this averaging method. When comparing our results with the one reported in the original paper [28,21] (green dashed line), we again see a slight mismatch between them. Specifically, the reported results for CNN with the HW leakage model act as an outlier in Figure 5c, which again emphasizes the influence of the random weight initialization and the need to provide averaged results over a number of profiling models.

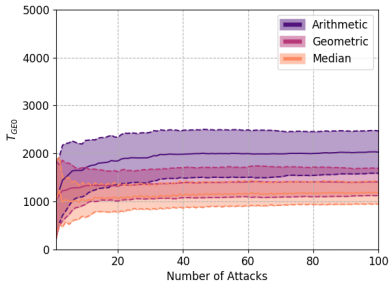
The number of attacks has a small influence, but again, there is no reason to use more than 50 attacks in the experiments. We see a very large standard deviation for CNN and the ID leakage model, indicating that the profiling model



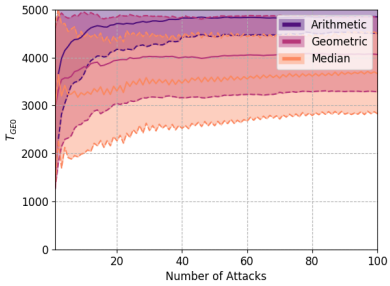
(a) Random MLP with the HW leakage model.



(b) Random MLP with the ID leakage model.



(c) Random CNN with the HW leakage model.



(d) Random CNN with the ID leakage model.

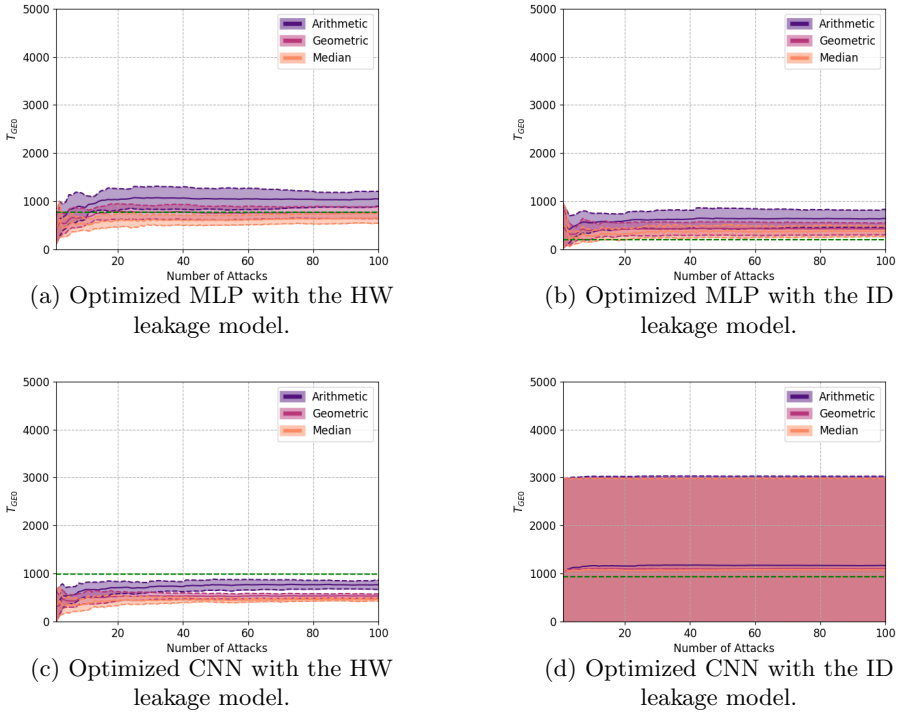
Fig. 4: T_{GE0}: attack on ASCAD_R with random MLP and CNN models.

is not very stable, so multiple experiments should be done to assess the attack performance properly.

Finally, for CNNs, we again see the synergistic effect of using multiple profiling models as we effectively develop an ensemble. An interesting perspective is that we can improve state-of-the-art architectures' results by making ensembles of the same architectures with different trainable parameters. We consider this very interesting as it allows easy constructions of ensembles aligned with the available results from the literature.

5.3 Results for the CHES_CTF Dataset

Finally, we attack the CHES_CTF dataset. Note that CHES_CTF with the ID leakage model always results in attack failure according to [21,28], so we use only the HW leakage model. The results from random model attacks are shown in Figure 6. The performance of the median of the geometric mean is similar, and both of them outperform the arithmetic mean that is commonly used by researchers and evaluators. The results for the random CNNs show not successful attack, which means that random selection of profiling architectures is not a good choice for this dataset. The number of attacks does not show a difference if used



(a) Optimized MLP with the HW leakage model.

(b) Optimized MLP with the ID leakage model.

(c) Optimized CNN with the HW leakage model.

(d) Optimized CNN with the ID leakage model.

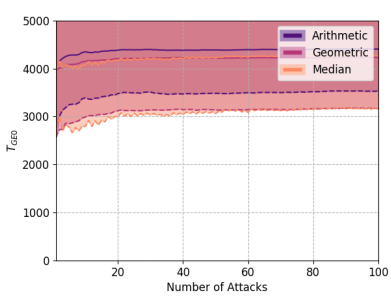
Fig. 5: T_{GE0}: attack on optimized MLP and CNN models with the ASCAD_R dataset.

more than 40, and the deviation for MLP is large, as many profiling models do not succeed in breaking the target.

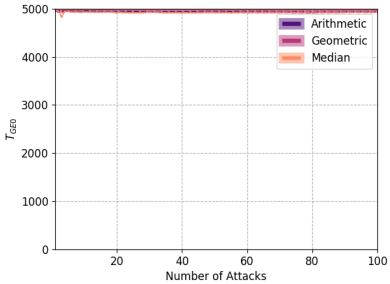
When attacking with state-of-the-art profiling models, the attack efficiency is dramatically improved. As shown in Figure 7, for both MLP and CNN, median performs better than geometric and arithmetic means. Therefore, we can confidently conclude that the median is a better way of calculating GE. Comparing our results and [28,21] (green dashed line), the latter performs significantly better. As mentioned before, since 20-model averaging compensates for the effect of the random weight initialization, we believe that our results reflect the real performance. A large deviation value additionally confirms those observations. Still, using ensembles could be an option to improve the attack performance. Finally, aligned with all previous cases, we do not see a significant impact of the number of attacks.

5.4 General Observations

1. Deep learning-based SCA can show radically different attack performance due to algorithmic randomness, and as such, the attack distribution is skewed.

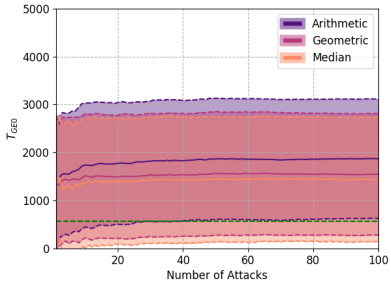


(a) Random MLP with the HW leakage model.

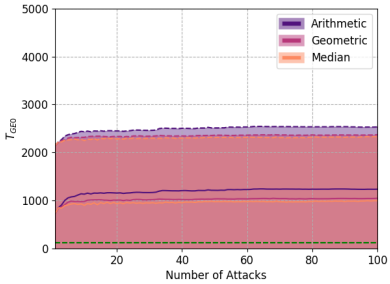


(b) Random CNN with the HW leakage model (most of the attacks failed to converge)

Fig. 6: T_{GE0}: attack on CHES_CTF with random MLP and CNN models.



(a) Optimized MLP with the HW leakage model



(b) Optimized CNN with the HW leakage model

Fig. 7: T_{GE0}: attack on optimized MLP and CNN models with the CHES_CTF dataset.

2. Arithmetic mean should not be used as the average estimate of the attack performance as it suffers due to commonly skewed distribution. Our experiments show that the median is the best choice.
3. Large number of independent experiments to average the attack performance does not increase the stability of results, indicating this as a simple option to speed up the evaluation process.
4. Large standard deviation with random models is expected as we use (radically) different profiling models. For state-of-the-art models, a large standard deviation indicates that the profiling models still have too much capacity.
5. It is necessary to report averaged performance over a number of different profiling models (or trainable parameters) to provide a reliable estimate of the actual attack performance.

6. It is possible to build strong attacks by using ensembles where we use different profiling models (as done in related works) but also by using a single model trained a number of times.

6 Conclusions and Future Work

This paper investigates the difficulty of assessing the attack performance for deep learning-based side-channel analysis. We experimentally show that the most appropriate summary statistics for evaluating deep learning-based SCA is median and not the arithmetic mean as commonly used. Furthermore, we show that the number of attacks plays only a marginal role where it is enough just not to use a very small number of attacks (e.g., run more than 40 independent attacks) to assess the attack performance properly. Next, we show that algorithmic randomness has a significant effect on the results, and to properly assess them, it is necessary to show averaged results and not only a single one (as commonly done). Finally, we show that ensembles of classifiers can reach better attack performance (as already shown). However, it is not necessary to build those ensembles from different profiling models as the algorithmic randomness connected with the training process suffices to produce different performing profiling models.

This paper dealt only with algorithmic randomness and deep learning. It would be interesting also to consider dataset randomness and use more summary statistics. For instance, while it is common to report average results over multiple experiments, no other summary statistics are reported. We consider reporting standard deviation a good option. Indeed, when comparing several deep learning algorithms, one can often see rather similar results. Nevertheless, the question is how stable those results are and if such additional information can help us to judge better what algorithm performs better.

References

1. 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), 163–188 (2020). https://doi.org/10.1007/s13389-019-00220-8, `https://doi.org/10.1007/s13389-019-00220-8`
2. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 45–68. Springer (2017)
3. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. pp. 45–68. Springer International Publishing, Cham (2017)
4. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: CHES. LNCS, vol. 2523, pp. 13–28. Springer (August 2002), San Francisco Bay (Redwood City), USA
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`

6. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Side-channel analysis of lightweight ciphers: Does lightweight equal easy? In: Hancke, G.P., Markantonakis, K. (eds.) Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10155, pp. 91–104. Springer (2016). https://doi.org/10.1007/978-3-319-62024-4_7, https://doi.org/10.1007/978-3-319-62024-4_7
7. 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 pp. 148–179 (2019)
8. 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
9. Li, H., Krček, M., Perin, G.: A comparison of weight initializers in deep learning-based side-channel analysis. In: Zhou, J., Conti, M., Ahmed, C.M., Au, M.H., Batina, L., Li, Z., Lin, J., Losiouk, E., Luo, B., Majumdar, S., Meng, W., Ochoa, M., Picek, S., Portokalidis, G., Wang, C., Zhang, K. (eds.) Applied Cryptography and Network Security Workshops. pp. 126–143. Springer International Publishing, Cham (2020)
10. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer (2016)
11. 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/
12. Martin, D.P., Mather, L., Oswald, E., Stam, M.: Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 548–572. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
13. Massey, J.: Guessing and entropy. In: Proceedings of 1994 IEEE International Symposium on Information Theory. pp. 204– (1994). https://doi.org/10.1109/ISIT.1994.394764
14. 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), 337–364 (Aug 2020). https://doi.org/10.13154/tches.v2020.i4.337-364, https://tches.iacr.org/index.php/TCHES/article/view/8686
15. Perin, G., Picek, S.: On the influence of optimizers in deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/977 (2020), https://eprint.iacr.org/2020/977
16. Picek, S., Heuser, A., Jovic, A., Batina, L.: A systematic evaluation of profiling through focused feature selection. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **27**(12), 2802–2815 (2019)
17. Picek, S., Heuser, A., Guilley, S.: Template attack versus bayes classifier. J. Cryptogr. Eng. **7**(4), 343–351 (2017). https://doi.org/10.1007/s13389-017-0172-7, https://doi.org/10.1007/s13389-017-0172-7
18. 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), 209–237 (Nov 2018). https://doi.org/10.13154/tches.v2019.i1.209-237, https://tches.iacr.org/index.php/TCHES/article/view/7339
19. 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. pp. 4095–4102 (2017)
20. Picek, S., Heuser, A., Wu, L., Alippi, C., Regazzoni, F.: When theory meets practice: A framework for robust profiled side-channel analysis. Cryptology ePrint Archive, Report 2018/1123 (2018), https://eprint.iacr.org/2018/1123
21. 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), 677–707 (Jul 2021). https://doi.org/10.46586/tches.v2021.i3.677-707, https://tches.iacr.org/index.php/TCHES/article/view/8989
22. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. pp. 30–46. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
23. Smith, L.N.: Cyclical learning rates for training neural networks. In: 2017 IEEE winter conference on applications of computer vision (WACV). pp. 464–472. IEEE (2017)
24. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009. pp. 443–461. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
25. Whitnall, C., Oswald, E.: Robust profiling for dpa-style attacks. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 3–21. Springer (2015)
26. 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), 147–168 (Jun 2020). https://doi.org/10.13154/tches.v2020.i3.147-168, https://tches.iacr.org/index.php/TCHES/article/view/8586
27. 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
28. Wu, L., Perin, G., Picek, S.: I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. IACR Cryptol. ePrint Arch. **2020**, 1293 (2020)
29. Wu, L., Weissbart, L., Krček, M., Li, H., Perin, G., Batina, L., Picek, S.: On the attack evaluation and the generalization ability in profiling side-channel analysis. Cryptology ePrint Archive, Report 2020/899 (2020), https://eprint.iacr.org/2020/899
30. 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), 1–36 (Nov 2019). https://doi.org/10.13154/tches.v2020.i1.1-36, https://tches.iacr.org/index.php/TCHES/article/view/8391