S-box Pooling: Towards More Efficient Side-Channel Security Evaluations

Yuanyuan Zhou[1,2] and François-Xavier Standaert[1]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
{yuanyuan.zhou,fstandae}@uclouvain.be
[2] SGS Brightsight, Delft, The Netherlands

Abstract. Nowadays, profiled attacks are the standard penetration tests for security evaluations. Often the security evaluators have to perform profiled attacks on each S-box to quantify the security strength of the target symmetric cryptographic algorithm implementations more accurately. The required time to conduct such profiled attacks is very long due to the number of profiling traces (for many certification bodies, at least 1,000,000 are mandated). It is getting even more time-consuming after introducing deep learning profiled attacks. Furthermore, some certification bodies instruct up to 5,000,000 or 10,000,000 profiling traces because modern embedded secure IC products have more and more countermeasures against side-channel attacks. It is a challenge to simultaneously decrease the number of required profiling traces and the required profiling time while retaining the attack performance for profiled attacks. In this work, we propose a simple yet remarkably effective pooling approach to address this problem for security evaluations. That is, pooling over the S-boxes to build a large profiling set and perform the profiling on this large set once. Intensive experiments are conducted with this pooling approach using different profiling tools (template attack and its pooled variant, stochastic model and deep learning) on three different AES implementations (a sequential S-box software AES implementation without masking, a sequential S-box software AES implementation with first-order masking and a parallel S-box hardware AES implementation with first-order masking). The experimental results have shown that the proposed pooling approach can lead to similar attack performance while decreasing both the required number of profiling traces and the required profiling time by a factor of 8 or even 16.

1 Introduction

1.1 The context of this work

In Kocher's seminal work [8], Side-Channel Attacks (SCA) were proposed to extract secret keys of cryptographic algorithms implementations via the timing side channel. Since then, SCA have drawn plenties of attention in the community. On the one hand, it is extended to different side channels, *e.g.*, power consumption [9], electromagnetic radiation [13]. On the other hand, different distinguishers and new approaches are also adapted to SCA. Amongst them,

profiled attacks are considered the most powerful SCA after Chari *et al.* published the novel template attacks [2]. Various research works have focused on improving profiled attacks from different perspectives, *e.g.*, for efficiency purposes [14,3], or portability and robustness. It is worth noting that there are two types of profiled attacks depending on what we are profiling, either profiling the key itself or profiling some intermediate data that depends on it (*e.g.*, S-box output). Generally speaking, profiling the key directly is more studied in the asymmetric cases, profiling key-dependent values (*e.g.*, S-box output) is more studied in the symmetric cases, and key transportation is rarely studied (though applies in all cases).

Concretely, SCA are a pillar for security evaluations of information security products[3], and profiled attacks are a de facto standard penetration test for cryptographic algorithms. For security evaluations, based on the rating policy to gain so-called AVA_VAN.5 security assurance[4], such a standard profiled attacks-based penetration test typically costs 3 to 4 weeks in total for an experienced security evaluator. It is grey-box testing although the security evaluators often can get access to the detailed design information (hardware and/or software) of the implemented symmetric algorithms being evaluated. The outline of time division (in total 3 to 4 weeks) for such a standard profiled attacks-based penetration test starting from scratch is below:

- Stage 1 ($1 \sim 2$ weeks): Understand the target implementation and narrow down the target interval as much as possible through SPA (**S**imple **P**ower **A**nalysis)/SEMA (**S**imple **E**lectro**M**agnetic **A**nalysis) and CPA (**C**orrelation **P**ower **A**nalysis)/CEMA (**C**orrelation **E**lectro**M**agnetic **A**nalysis) (or similar techniques). It also includes the scripting time of the measurement and measurement/analysis time. It is vital to narrow down the target interval as much as possible because the sampling rate for measuring EM (ElectroMagnetic Raditation) traces can be very high (5 GHz to 10 GHz). Hence, the number of sample points within the interesting interval can be the bottleneck of the subsequent profiled attacks. Another critical step in this stage is to choose the appropriate EM signal after surface scans of the chip. It is easier when the location of the symmetric algorithm co-processor is known to the evaluators, but if it is not known, this task can become very time-consuming.
- Stage 2 ($0.5 \sim 1$ week): Measure at least 1,000,000 (recently some certification bodies demand 5,000,000 or up to 10,000,000 because of more and more security countermeasures in the modern secure microcontrollers) profiling traces and at least 150,000 attack traces (50,000 for each of 3 different attack keys).
- Stage 3 ($1 \sim 1.5$ weeks): Preprocess the measured traces (*e.g.*, align them [17]) and conduct template and deep learning profiled attacks on all S-boxes (*e.g.*, 16 in the AES case) in an attempt to recover those three different attack keys (3 sets of attack traces as mentioned in Stage 2). It also includes the report-

[3] https://www.sogis.eu/uk/supporting_doc_en.html
[4] https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-1.pdf

ing time. More precisely, due to the implemented countermeasures such as jitters and random delays, often the evaluators have to re-synchronize (align) the traces step by step to get closer and closer to where the S-box operations are supposed to take place [17]. It commonly costs 1 or 2 days considering the number of traces and the number of sample points of every trace. Afterwards, the evaluators need to perform both template and deep learning attacks on each S-box one by one. Currently, the best practice is to execute profiling and attacking on each S-box one by one [4]. It naturally requires lots of time taking into account the number of sample points (normally a few thousand sample points for one S-box considering the EM traces) and the amount of the traces. With deep learning evaluations now being mandated by certification bodies, this issue further amplifies. Because usually, compared to the classical template attacks, the training of the neural networks requires much more time owing to the many hyperparameters to be tuned.

1.2 Problem to be addressed

To simplify the task of the evaluators, there is not much we can do considering Stage 1 and Stage 2. Also, we cannot skip or shorten the necessary re-synchronization preprocessing for Stage 3 [17]. One may argue that deep learning profiled attacks can be effective to tackle this, however, re-synchronization preprocessing makes deep learning attacks more efficient from security evaluation perspective according to [17] and the same target implementations are considered in this work. Hence, the only option left is to *find a more efficient way to decrease both the number of required profiling traces and the required profiling time while preserving the attack performance*. Somewhat surprisingly, there have been limited attempts to characterize and improve such practical challenges. In this work, therefore, we aim to study it for popular profiled distinguishers and various target implementations.

1.3 Our contribution

The general idea of our work is S-box pooling. It is to first reconstruct a larger set of profiling traces by pooling the profiling traces corresponding to each S-box. The second step is to execute profiling only once on this new profiling traces set. Finally, it is to attack all the S-boxes to disclose all the subkeys (*e.g.*, 16 in the AES case) using the attack traces.

Our contributions in this context are twofold: First, from a data complexity point of view, the proposed S-box pooling approach can decrease the required amount of profiling traces by a factor of 8 or 16 while preserving the attack performance as shown in Table 2. In other words, the evaluators can measure eight or even sixteen times fewer profiling traces during Stage 2. The extra benefit will be reducing the preprocessing time of Stage 3 because 8 or 16 times fewer traces need to be re-synchronized. Second, from the time complexity perspective, the proposed S-box pooling approach can decrease the required profiling time by a factor of 8 or 16 (considering the AES case) while preserving the attack performance.

1.4 Related work

We note that a recent and independent work investigated a similar technique (S-box pooling, which they denote as cross-subkey training) from a different perspective [6]. Namely, their focus is to improve the attack performances when the number of profiling traces is overly limited. They use the accuracy metric while we use the guessing entropy metric for attack performance comparison. In their work, only deep learning profiled attacks are considered so accuracy could be a suitable metric. They have observed significant accuracy improvement with cross-subkey training compared to the case where the amount of profiling traces is overly limited. It can be explained by the fact that sufficient profiling traces are available by using cross-subkey training and all 16 S-boxes are expected to leak in a very similar way because of the unmasked sequential software AES S-box implementation. Both factors lead to the better generalization of the deep learning model they are using. We relatively aim to optimize the profiling complexity when this number is sufficient. We also cover more target devices (they only target an unprotected AES software implementation: we additionally cover a masked AES software implementation and a masked hardware one) and distinguishers (they only consider one deep learning distinguisher: we additionally cover other state-of-the-art profiling tools). Overall, their conclusions are complementary to ours and show that S-box pooling can also improve attack performances when the number of profiling traces is limited.

1.5 Organization of the paper

The rest of this paper is organized as follows. Section 2 introduces the necessary background on profiled attacks used in this work. Then, we describe our proposed S-box pooling approach and methodology in Section 3. Finally, Section 4 demonstrates the effectiveness of this pooling approach based on the intensive experimental results on three different AES implementations, for different profiling tools.

2 Background

Since Chari *et al.* have introduced template attacks in their pioneering work [2], profiled attacks have gradually become the commonly recognized most powerful side-channel attacks. Profiled attacks consist of two phases, *i.e.*, the profiling phase and the attack phase. During the profiling phase, an attacker/evaluator uses a profiling device (and has control of the key or at least knows the key) to model the leakage characteristic of the target key-dependent sensitive data (typically the S-box output of symmetric algorithms) with the side-channel traces of the target implementation. The outcome of the profiling phase is the built leakage characteristic models for every possible target sensitive data value, *e.g.*, 256 values of the AES S-box output [5]. During the attack phase, the attacker/evaluator uses the victim device to measure the side-channel traces of the target implementation. And then, he/she matches the traces with the previously built leakage

[5] That is, the identity model is used for labeling as what we do in this work. The Hamming Weight model can also be used for labeling.

characteristic models of the target sensitive data. For each attack trace, the adversary calculates the target sensitive data based on the known input and the guessed key unit (*e.g.*, one key byte). A score is computed for each possible guessed key unit value (256 possible values for one byte of AES key) per each trace. In the end, for each hypothesised key unit value, the scores for all the attack traces are combined using *e.g.*, the maximum likelihood method to get a combined score of that specific guessed key unit value. The combined score of each possible hypothesised key unit value is compared to find the highest one. The hypothesised key unit value with the highest score is considered the recovered key unit. This attack process is repeated for all key units to reveal the complete key.

In this work, we use four different state-of-the-art profiled attacks, namely, template attack (TA) and its pooled variant (TA_p), stochastic attack (SA) and deep learning attack (DL), to experimentally investigate the efficiency of our proposed pooling profiling traces approach.

2.1 Template attack

From an information-theoretic viewpoint, TA is believed to be the most powerful type of SCA [2] when (1) the noise of side-channel traces follows the Gaussian distribution and (2) an unlimited number of traces are available. It makes use of a multivariate normal distribution to model the probability density function of each possible target sensitive data given a leakage observation, so it is parameterized as

$$p(L = l|S = s) = \frac{1}{\sqrt{(2\pi)^d |\sigma_s|}} e^{-\frac{1}{2}(l-\mu_s)^\top \sigma_s^{-1}(l-\mu_s)}. \tag{1}$$

In this equation, d is the number of sample points. $|\sigma_s|$ denotes the determinant of the covariance matrix and $\top$ indicates the transpose. In practice, usually, some points of interest (POIs) are detected first for dimension reduction purpose.

In the attack phase, the probability $p(L = l|K = g)$ for each key candidate is set by $p(L = l|S = s)$ given a known input of each attack trace. The classification of each key guess is then computed based on Bayes' Theorem as follows:

$$p(K = g|L = l) = \frac{p(L = l|K = g) \cdot p(K = g)}{p(L = l)}. \tag{2}$$

Based on an assumption that all attack traces are independent, to make use of all available attack traces for each key guess, a final score of each key guess is calculated as below:

$$p_g = p(g|L) = \frac{\prod_{m=1}^{M} p(L = l_m|K = g) \cdot p(K = g)}{\prod_{m=1}^{M} p(L = l_m)}. \tag{3}$$

In practice, the p_g is usually calculated using the sum of the log-posterior to avoid the potential arithmetic underflow problem. The highest p_g indicates the correct key candidate g^*. The pooled variant of the TA, next denoted TA_p, is

proposed in [3]. It uses only a single pooled covariance matrix σ to reduce the profiling complexity, which is dominated by the estimation of the covariance. The rest is the same as normal TA.

2.2 Stochastic model attack

SA [14] relies on linear regression to build the leakage characteristic models of sensitive data. It assumes that the side-channel leakage observation of the target sensitive data s at time t consists of two parts $l_t(s) = h_t(s) + R_t$, where $h_t(s)$ is the key-dependent part and the latter one is a non-key dependent noise term with zero mean. Similar to TA, the profiling also contains two parts, the linear approximation $\hat{h}_t$ of h_t and the estimation of noise-related covariance matrix σ. The estimation of $\hat{h}_t$ for each time instantiation is done in a chosen suitable u-dimensional vector subspace $\mathcal{F}_{u;t}$. In this work, we choose the subspace $\mathcal{F}_9$ by utilizing the bitwise coefficients of the AES S-box output. Afterwards, d_1 POIs are chosen based on the estimated $\hat{h}_t$ to compute the covariance matrix σ. The profiling results in a Gaussian multivariate density $\hat{f} : \mathbb{R}^{d_1} \to \mathbb{R}$.

During the attack phase, we only consider the maximum likelihood principle to recover the key following [4,14]. More specifically, the correct key guess g^* is the one that maximizes:

$$p_g = p(g|L) = \prod_{m=1}^{M} \hat{f}(\mathbf{l_t}(s_{m,k}) - \hat{\mathbf{h}}_\mathbf{t}(s_{m,g})). \tag{4}$$

In this equation, k indicates the unknown correct subkey to be revealed.

2.3 Deep learning DPA attack

Different from the above-mentioned classical profiled attacks, DL makes no assumption of the leakage characteristic. It exploits the features (sample points with regard to side-channel traces) to classify the labels (sensitive data in the SCA context) using neural networks (details arrive in Subsection 4.1). The training process of neural networks (corresponds to the profiling) aims to construct a classifier function $F(.) : \mathbb{R}^d \to \mathbb{R}^{|S|}$. This function maps the input trace $\mathbf{l} \in R^d$ to the output vector $\mathbf{p} \in R^{|S|}$ of scores. During the training, for each training batch, the backpropagation method [7] is used to update the trainable parameters of the neural network model aiming at minimizing the loss, which is calculated to quantize the classification error over each training batch. In the attack phase, the built trained model (*i.e.* $F(.)$ with all the final updated trainable parameters) is used to classify each attack trace to obtain its score vector $\mathbf{p}[s_{m,g}]$. Afterwards, the final score vector of each key candidate $\mathbf{p}[g]$ is calculated using all the attack traces (similar to Equation 3). The key candidate $g^* = \text{argmax}\ \mathbf{p}[g]$ is considered the right subkey.

3 Methodology

As discussed in Subsection 1.2, the problem to be solved in this work is: to simultaneously decrease both the required number of profiling traces and the required profiling time while the performance of the online attack is essentially unchanged.

In the following, we first introduce the proposed S-box pooling approach and how it resolves this challenge for security evaluators. We then describe the methodology, which we will use to systematically investigate the soundness of the proposed pooling approach in a profiled attacks context. It includes the metric for comparing profiling performance, the metrics used for training neural networks, the knowledge of POIs assumption, the POI selection approach, and the choice of parameters for different profiled attacks in our experiments. For simplicity, we will only discuss the AES case. But the principle can be easily extended to other symmetric cryptographic algorithms.

3.1 S-box Pooling profiled attack

The core idea of this pooling approach is: to extract the profiling traces of each S-box and to stack the extracted profiling traces for all S-boxes in order to build a new large set of profiling traces. In theory, this approach is based on improving the signal-to-noise ratio of the profiling traces by increasing the amount of the profiling traces. Instead of directly measuring a lot of profiling traces, it is to gather enough profiling traces by extracting and stacking all available S-box calculation segments of each side-channel trace. For instance, we measure 1,600 profiling traces of the first round of AES encryption. So we have 1,600 traces for each S-box. By extracting and stacking then we build a new set of 25,600 (= 16×1,600) profiling traces since there are 16 S-boxes. In the ideal situation where the S-boxes are leaking according to the same model, this strategy will lead to a reduction of the number of profiling traces by a factor of 16 (for the AES). Furthermore, the profiling time will be reduced by the same factor (In this work we only considered that the same deep learning model is used for performance comparison with and without S-box pooling.) since only a single model will have to be built. Of course, in practice, the situation may not be ideal and it is the goal of our following investigations to clarify the extent to which S-box pooling is a good trade-off for concretely relevant case studies. For this purpose, we used the following 4-step method:

1. Step 1: Measure enough profiling traces (the number of profiling traces marked as N_{ori}) being able to retrieve the AES key, perform TA, TA_p, SA and DL profiled attacks on each S-box separately to recover all 16 subkeys. Make them the baseline attack performance for later use and comparison.
2. Step 2: Determine the minimum required number of profiling traces (denoted as N_{min}) to achieve similar profiling performance as the baseline using a binary search algorithm. That is, starting from $N_{ori}/2$ profiling traces to conduct profiled attacks to compare the profiling performance with the baseline until profiling performance similar to the baseline obtained using N_{min} profiling traces.
3. Step 3: Build four new sets of profiling traces by pooling the profiling traces of each S-box based on the predefined S-box pooling ratio (labelled as PR) of 16, 8, 4 and 2 with the determined N_{min}, in which the pooling ratio of 16 means pooling $N_{min}/16$ profiling traces of each S-box to build the new set of profiling traces. Meanwhile, construct another four new sets of profiling

traces without pooling the profiling traces of each S-box, *i.e.*, directly taking N_{min}/PR profiling traces from the original measurement for each S-box.

4. Step 4: For each S-box pooling ratio PR, perform TA, TA_p, SA and DL profiled attacks on each S-box using the new built set of profiling traces with pooling and using the newly constructed set of profiling traces without pooling. Note that, with pooling, we only need profiling once. While without pooling, we need profiling 16 times because we have to execute profiling for each S-box one by one following the current best practice.

Eventually, compare the profiling performance with and without pooling to the baseline.

3.2 Knowledge of POIs assumption

There is an implicit assumption about the knowledge of the POIs to apply this pooling approach. That is, evaluators can figure out the rough timing interval of each S-box of one AES round calculation in the side-channel traces. Through SPA/CPA and SEMA/CEMA as mentioned in Stage 1 in Subsection 1.1, this is feasible for most of the evaluated security products in the security evaluation grey-box testing context. For instance, the evaluators can vary the input length and/or the key length, perform correlation analyses on the input and output data, make use of the design information such as the location of the AES co-processor in the glue logic area or temporarily switch off some countermeasures like jitters.

Concerning the POI selection for classical profiled attacks TA, TA_p and SA, we use the popular SOST (Sum Of Squared pairwise T-differences) [4] for POI selection, since typically the masking countermeasure is implemented in the evaluated products. It does not require any secret information about the implementation, and it can be easily, efficiently computed. It is also commonly used by security evaluators for profiled attacks.

3.3 Metrics and Selection of parameters

In order to compare the profiling complexity of different tools, information-theoretic metrics like mutual information (MI) are natural candidates [15]. Yet, it requires that all the investigated profiling methods give rise to probabilistic outcomes. In some cases, the tools we consider fairly output scores that do not directly embed such a probabilistic meaning. Therefore, we will preferably evaluate our strategy based on the guessing entropy (GE) metric [16]. Note that both metrics have the same comparative value and are thus equally good for our purposes (but computing guessing entropy curves is generally more expensive than estimating information-theoretic metrics).

Regarding the metrics used to train the neural networks, we use the Negative Log-Likelihood (NLL) loss function [1] for DL profiled attacks, because it is proved that minimizing the NLL loss is equivalent to maximizing the Perceived Information and thus to minimize the online attack complexity, thanks to the recent work [11].

For SA profiled attack, since we do not precisely know which model is followed by the target sensitive intermediate data, we choose a linear 9-element basis (the

eight S-box output bits together with a constant), which is the standard choice for this distinguisher [14].

4 Experimental results

To confirm the soundness of the proposed pooling approach in security evaluations, in this section, we apply the pooling approach to three different representatives of AES implementations, from easy to hard. That is, we consider a sequential S-box software AES without masking, a sequential S-box software AES with first-order masking [12] and a parallel S-box hardware AES with first-order masking. The first DUT (Device Under Test) in Subsection 4.2 represents a sort of ideal case: all the S-boxes are supposed to leak in almost the same way, because all the S-boxes are executed sequentially, and no countermeasures are involved. The aim is to verify how the proposed S-box pooling approach behaves in an almost ideal leakage context. In Subsection 4.3, the second DUT is going further: to investigate how efficient the proposed S-box pooling approach will be in a more realistic scenario. In this case, because all the S-boxes are still executed sequentially, there should be no interference regarding leakage characteristics between them. Finally, the third DUT in Subsection 4.4 is the scenario closest to the security evaluations, because it is a hardware AES co-processor implementation with random masking, which is just the case for most modern evaluated products. In this case, multiple S-boxes are executed in parallel, so the leakage characteristic of each S-box will be affected by the others being executed at the same time. The involved random masking of each S-box will also introduce more discrepancy in the leakage characteristic given a limited amount of profiling traces. It is therefore essential to know how much profiling efficiency the evaluators can gain in this realistic scenario. We label those three DUTs as DUT1, DUT2 and DUT3 in the rest of this paper.

4.1 Common settings

For all the three DUTs, we compare the profiling performance with and without S-box pooling using four different state-of-the-art profiled attacks, namely, TA, TA_p, SA and DL profiled DPA attacks. For each DUT, the same device is used for the profiling and attacking phase. All these four profiled attacks are implemented in Python and PyTorch version 1.7.1 with an NVIDIA GTX 1080Ti GPU. The following common settings are used for all the experiments in this work,

1. No matter whether the masking countermeasure is present or not, the target of all profiled attacks in this work is the first round S-box output of AES encryption. That is, $Sbox(p[i] \oplus k[i])$, in which p denotes a 16-byte AES input and k corresponds to a 16-byte AES key, with i the index of the S-box. Note that we focus on the common scenario where the masked randomness is not given to evaluators. Our conclusions would apply identically in the more worst-case scenario where this randomness is known for profiling. But the concrete gains would be less significant (because we expect the profiling task to be significantly simplified in that case).
2. The used DL neural network model for DL attacks is a published MLP (Multi-Layer Perceptron) [10]. The structure of this model is simple and

Table 1: MLP model details

MLP
nb_epoch = 50 batch_size_training = 32
Dense(50, activation="relu", input_shape=(nb_samples,)) BatchNormalization() Dense(100, activation="relu") BatchNormalization() Dense(256, activation="softmax") compile(loss='categorical-crossentropy', optimizer='adadelta', metrics=['accuracy'])
learning_rate_policy = ReduceLROnPlateau(optimizer, 'min', factor=0.05, verbose=True)

shallow, while it showed pretty good performance for similar DUTs (similar to our DUT1 and DUT2) in that paper. It also showed great performance for all the three DUTs in our work. In addition, we use the *Adadelta* optimizer and adopt the adaptive learning rate policy *ReduceLROnPlateau* to gradually decrease the learning rate (we used the default Adadelta optimizer initial learning rate of 1.0) with a factor of 0.05 if the training stagnates. We use a batch size of 32 and 50 as the number of epochs for all the DL experiments. All the profiling traces and attack traces used for DL experiments are normalized using the *StandardScalar* function from the Scikit-learn library by removing the mean and scaling to unit variance. For all the DL experiments, we isolate 15% profiling traces as a validation set. This is due to the fact that a validation data set is critical to DL performance as it provides a way to timely detect over-fitting [5]. We used the trained model with the highest validation accuracy for the DL attacks. As aforementioned, all the DL experiments utilize the NLL loss to train the model. The details of the used DL model and the hyperparameters are described in Table 1.

3. As mentioned in Step 2 of Subsection 3.1, we need to determine the minimum number of required profiling traces N_{min} for each DUT before the performance comparison for different profiled attacks. To this end, we use DL attacks to determine the N_{min} for each DUT because only DL attacks can fully recover all the subkeys for all three DUTs as shown in the eprint version of the paper.

4.2 Setting #1: an unmasked sequential AES S-boxes implementation

DUT1 is the ChipWhisperer unmasked software AES with 16 sequential S-boxes. It is expected that all the 16 S-boxes of this AES implementation leak in the same way (or with negligible discrepancy) because of the sequential implementation of the S-box layer without masking. The goal of using this DUT is to verify whether the proposed pooling approach can achieve similar profiling performance using 16 times fewer profiling traces in such an almost ideal context, which is the optimum result we are seeking in terms of decreasing the required number of profiling traces and the required profiling time.

Implementation settings. This software AES implementation is based on an 8-bit ATXmega128D4-AU microcontroller. The 16 S-boxes are executed one by one. A set of 11,000 power consumption traces of 3,000 sample points each has been measured via the on-board ADC of the ChipWhisperer Lite board for the experiments. The CPU is running at 7.37 MHz and the sampling rate is 29.48 MHz. More specifically, 10,000 profiling traces have been measured with random key data and random input data, 1,000 attack traces have been measured with a fixed random key and random input data.

Based on the knowledge of POIs assumption discussed in Subsection 3.2, we first identify the time intervals of each S-box in the power traces utilizing SPA and CPA. In the end, we cut 50 sample points (*e.g.* sample points 110∼160 for S-box 1, 206∼256 for S-box 2, ..., 1550∼1600 for S-box 16) for each S-box from the originally measured power traces to build 16 new subsets of traces, where each new subset corresponds to one S-box. These 16 subsets of traces as a whole are denoted as nPR_1.

Following our designed methodology in Subsection 3.1, we first perform TA, TA_p, SA and DL attacks on these 16 new sets of traces as the baseline of attack performance. This baseline is marked with nPR_1 in the subsequent GE results. Second, the minimum required number of profiling traces N_{min} is determined using a binary search algorithm. In this case, N_{min} is 6,700 based on the DL attack results as explained in Subsection 4.1.

Next, we make use of the proposed pooling approach to build four new sets of profiling traces according to the PR of 16, 8, 4 and 2. They are denoted as PR_16, PR_8, PR_4 and PR_2 respectively. Also, another four sets of profiling traces without pooling are constructed by directly taking N_{min}/PR profiling traces for each S-box from the previously built trace sets nPR_1. They are denoted as nPR_16, nPR_8, nPR_4 and nPR_2 respectively. Similar to nPR_1, each of nPR_16, nPR_8, nPR_4 and nPR_2 consists of 16 subsets, and each subset corresponds to one S-box. We then conduct TA, TA_p, SA and DL attacks on PR_16, PR_8, PR_4, PR_2, nPR_16, nPR_8, nPR_4 and nPR_2.

For the sets with pooling the profiling traces PR_16, PR_8, PR_4 and PR_2, we only perform profiling once followed by attacking all 16 S-boxes to retrieve the subkeys. On the contrary, for the sets without pooling the profiling traces nPR_16, nPR_8, nPR_4 and nPR_2, we need to perform profiling 16 times, each profiling is for one S-box followed by attacking that single S-box.

Finally, we compare all the conducted profiled attacks results (with and without pooling profiling traces) with the baseline results.

Attack results. By comparing the aforementioned profiled attacks results with the baseline GE results as shown in Figure 1, it is demonstrated that using S-box pooling can decrease both the number of profiling traces and profiling time by a factor of 16 considering the DL attack results. For readability, here we only present the results for 3 different sets, *i.e.*, the baseline GE results (nPR_1, 6,700 profiling traces), the GE results with pooling with PR of 16 (PR_16, 6,700 = 16×418 profiling traces) and the GE results without pooling with PR

of 16 (nPR_16, 418 profiling traces). The full comparison is given in the eprint version of the paper.

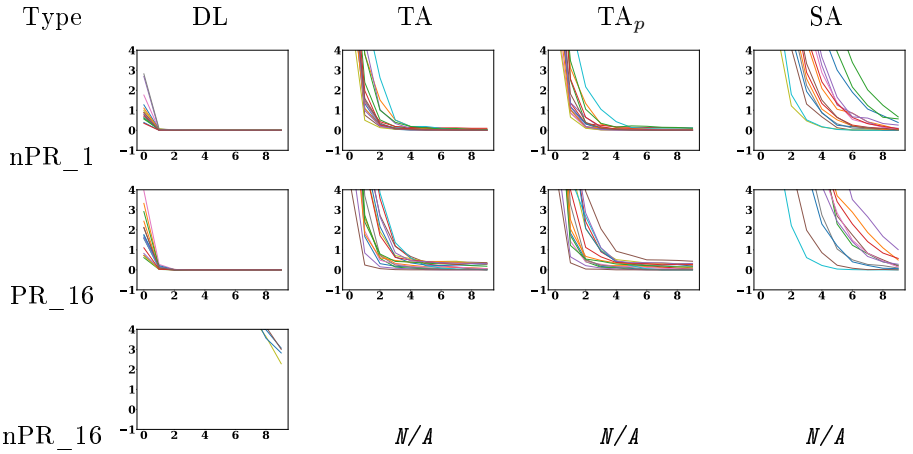

Fig. 1: DUT1 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy.

Considering the GE results of different profiled attacks on this DUT in the same setting (the same PR, with pooling or not), it is worth noting that DL shows the best attack performance. It suggests that the DL attack has a better generalization for this DUT. TA and its variant TA$_p$ show similar but slightly worse attack performance while SA shows the worst attack performance for this DUT, which is in line with the observations in [4][6]. For the nPR_16 setting, the GE results of TA, TA$_p$ and SA attacks are not available due to very few profiling traces for each class of S-box output, which leads to singular results during the profiling.

More interestingly, this trend is the same when we consider the impact of pooling profiling traces. It can be observed that the DL attack performance of the PR_16 case using pooling is already comparable to the baseline DL attack performance, while the DL attack performance of the nPR_16 case without using pooling is way worse than the baseline DL attack performance. The DL attack performance of the PR_8 case using pooling already outperforms the baseline DL attack performance. Hence, these GE results show that pooling can decrease the required number of profiling traces by a factor of 16 without reducing the attack performance. This is the optimal result one can achieve and it most likely holds because all 16 S-boxes are leaking in almost the same way. From a time complexity point of view, using a pooling ratio of 16 means using the same total amount of profiling traces as the baseline case, while only profiling once in the

[6] It is out of scope because the goal of this work is to verify the efficacy of the proposed pooling approach in terms of decreasing both the required number of profiling traces and the required profiling time.

pooling case and profiling 16 times in the baseline case. It results in decreasing the required profiling time by a factor of 16 as well. As mentioned before, this is an almost ideal case: no masking and sequential S-boxes. Next, we will further investigate the impact of the masking countermeasure on the performance of the proposed pooling approach.

4.3 Setting #2: a masked sequential AES S-boxes implementation

DUT2 is the public data set ASCADv1, which contains the traces of a first-order masked software AES with 14 sequential S-boxes. Because the first two S-boxes are not masked, only the last 14 masked S-boxes are evaluated in our experiments. This is still a sequential S-boxes implementation, so it is expected that those S-boxes will leak in a very similar way.

Implementation settings. The ASCADv1 data set is based on an implementation in an 8-bit ATMega8515 microcontroller with a first-order masking countermeasure. 60,000 EM (Electromagnetic Radiation) traces with a fixed AES key were acquired using an EM coil at a sampling rate of 2 GHz, in which 50,000 are the profiling traces and the remaining 10,000 traces are the attack traces. Each trace consists of 100,000 sample points.

Based on the knowledge of POIs assumption, we narrowed down the time intervals of each S-box in the EM traces. In the end, 700 sample points (*e.g.*, sample points 45400∼46100 for S-box 3, 32910∼33610 for S-box 4, ..., 18330∼19030 for S-box 16) for each S-box were cut out from the originally acquired EM traces to build 14 new subsets of traces. These 14 subsets of traces as a whole are denoted as nPR_1. TA, TA_p, SA and DL attacks are then conducted on nPR_1 as the baseline of attack performance. The minimum required number of profiling traces N_{min} of 37,000 was found based on the DL attack results.

We further constructed four new sets of profiling traces according to the PR of 14, 7, 4 and 2. They are denoted as PR_14, PR_7, PR_4 and PR_2 respectively. In addition, their counterparts without S-box pooling, *i.e.*, nPR_14, nPR_7, nPR_4 and nPR_2 were constructed as well. TA, TA_p, SA and DL attacks were then performed on these 8 sets of traces and the results are compared with the baseline.

Attack results. Figure 2 displays our results. For the same reason as in the previous case, only the results of nPR_1 (37,000 profiling traces), PR_14 (37,000 = 14×2,642 profiling traces) and nPR_14 (2,642 profiling traces) are shown here, our eprint version of the paper provides the full results. Again, the DL attack results suggest that using pooling can decrease both the number of profiling traces and profiling time by a factor of 14.

Our observations regarding the different performances of different profiles distinguishers are similar to the ones made for DUT1. Namely, the DL attack shows slightly better results than TA and its variant TA_p, while SA shows the worst performance.

The same trend is also observed with regard to the impact of pooling profiling traces: PR_14 already gives slightly better results than the baseline nPR_1 considering the DL attacks. The results without using pooling are significantly

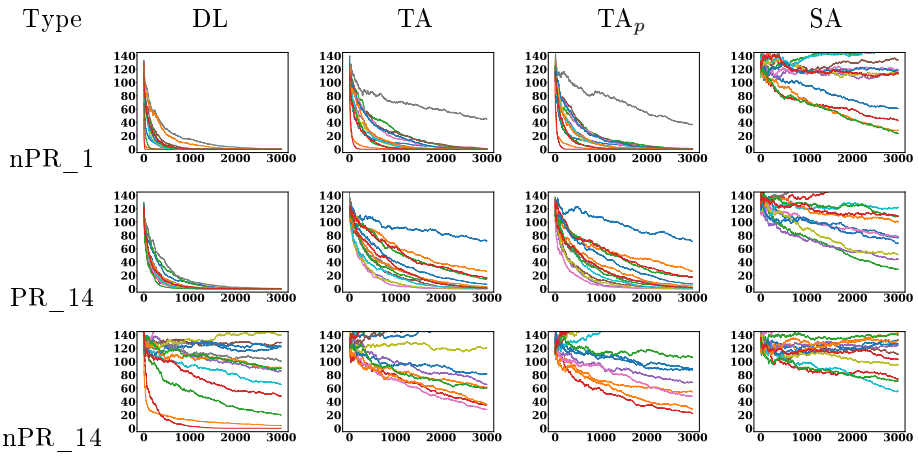

Fig. 2: DUT2 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy.

worse than the baseline ones. In short, from both the data complexity and time complexity viewpoints, using pooling can decrease the required number of profiling traces and required profiling time by a factor of 14, which is the optimal result. Most likely all those 14 masked S-boxes leak in a very similar way.

In this sequential masked S-boxes case, using pooling also leads to optimal attack performance. It further confirms that S-box pooling can be used for protected implementations in security evaluations. To further assess the impact of other implementation factors on the performance of the pooling approach, a very realistic scenario is finally brought into our scope. Namely, many modern secure microcontrollers have a dedicated AES co-processor, in which random masking is adopted and parallel S-boxes are sometimes implemented for performance purpose. Such an implementation will be assessed in our next target implementation.

4.4 Setting #3: a masked parallel AES S-boxes implementation

DUT3 is a 90 nm secure microcontroller with an AES co-processor, which is equipped with first-order masked parallel S-boxes, *i.e.*, 4 S-boxes are executed at the same time. It is expected that the leakage characteristic of each S-box within a group will be affected by the other 3 S-boxes in the same group being executed in parallel. Furthermore, this DUT also has other built-in countermeasures such as hardware and software time jitters, power balancing and power smoothing.

Implementation settings. A set of 520,000 EM traces of 100,000 sample points each has been measured via an SGS Brightsight EM coil using a LeCroy Waverunner 620Zi oscilloscope at a sampling rate of 10 GHz. The coil is located on top of the AES co-processor from the back side of the chip. The operating frequency of the AES co-processor is 32 MHz with variable internal clock enables. 480,000 profiling traces have been measured with random key data and random

input data. 40,000 attack traces have been measured with a fixed random key and random input data.

In accordance with the knowledge of POIs assumption, using SPA/CPA and SEMA techniques with the knowledge of the AES co-processor location, the time intervals of each group of 4 parallel S-boxes were figured out in the EM traces. For instance, different key lengths and different input lengths were used to execute the AES encryption to observe the difference they caused in the power and EM traces. Different numbers of AES rounds could be distinguished in the EM traces and each round contains 4 groups of EM peaks, with each group corresponding to the execution of 4 parallel S-boxes. It has to be mentioned that multiple alignments steps had to be done because the measured EM traces were heavily misaligned. We used the same re-synchronization method as in [17], which exploits correlation to align each group of EM peaks in the traces. It is a 3-step procedure.

1. Manually select a searching interval S that contains the operation to be aligned among all the traces.
2. Manually choose a smaller reference interval R_{T_i} specific to each trace T_i.
3. Within the whole interval S of each trace, search for the segment to be aligned by computing the Pearson's correlation between each segment (the same length as R_{T_i}) and the reference feature R_{T_i}. The right segment is the one showing the highest correlation within the whole interval S. The trace is abandoned if the highest correlation is lower than a pre-defined threshold chosen by the evaluator.

During the measurement campaign, it was not possible to trigger the oscilloscope close to the first AES round. Therefore, the EM traces were aligned several times to get close to the first AES round step by step, followed by more local alignments within the first AES round because there are 4 groups of EM peaks to be aligned one by one. Targeting different time intervals (for both the searching interval and reference interval), we applied this alignment method multiple times to the EM traces. In general, we chose new intervals when the misalignment was getting larger, and we repeated this process until the target interval was well aligned. In this way, we can gradually align each group of EM peaks corresponding to 4 parallel S-boxes being executed.

After all the alignments, 500 sample points for each group of 4 parallel S-boxes were kept from the original EM traces to build 16 new subsets of traces and as a whole, they are marked as nPR_1. We used each segment 4 times, targeting each S-box once. TA, TA_p, SA and DL attacks were then conducted on nPR_1 as the baseline of attack performance. The minimum required number of profiling traces N_{min} of 410,000 was further determined based on the DL attack results.

Similar to previous DUTs, we prepared eight new sets of profiling traces with or without pooling the profiling traces, *i.e.*, PR_16, PR_8, PR_4 and PR_2 and nPR_16, nPR_8, nPR_4 and nPR_2. We performed TA, TA_p, SA and DL attacks on these 8 sets of traces and compared the results with the baseline.

Attack results. The comparison of results is shown in Figure 3. Once again, we only present the results of nPR_1 (410,000 profiling traces), PR_16 (410,000

= 16×25,625 profiling traces), PR_8 (820,000 = 16×51,250 profiling traces) and nPR_8 (51,250 profiling traces). The full results are provided in the eprint version of the paper. This time the DL attack results demonstrate that using pooling can decrease both the number of profiling traces and profiling time by a factor of at least 8.

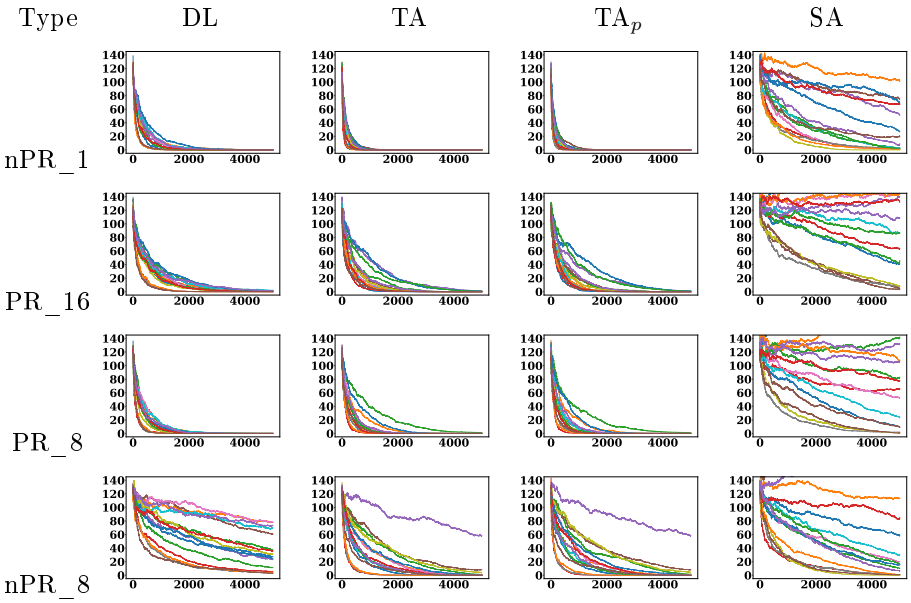

Fig. 3: DUT3 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy.

Different from the previous two DUTs, the TA and its variant TA$_p$ show better results than DL for all the 5 sets of profiling traces without pooling, *i.e.* nPR_16, nPR_8, nPR_4, nPR_2 and the baseline nPR_1. The other way around is observed if the proposed S-box pooling approach is adopted: DL then slightly outperforms TA and TA$_p$. Focusing on the pooling cases and the DL results, using PR of 16 achieves slightly worse results compared to the baseline, while using PR of 8 leads to better results than the baseline. These observations put forward that even when evaluating a more challenging (and therefore more practically relevant) hardware implementation with masking, the S-box pooling approach remains effective. It does not lead to an optimal factor gain of 16 for the data and time complexity but still reduces these complexities to a significant factor 8. This sub-optimal gain is most likely caused by the parallel execution of 4 S-boxes. The leakage model of a single S-box is interfered by other 3 S-boxes among the same group.

To summarize, taking all the results for all the 3 DUTs as shown in Table 2 into account, it is concluded that, for unmasked and masked sequential S-boxes implementations, the gain of using pooling can reach the optimal, *i.e.*,

Table 2: Summary of the experiments of 3 DUTs

	PR	N_{ori}	N_{min}	Samples	Implementation	Data Complexity Gain	Time Complexity Gain
DUT1	16/8/4/2	10,000	6,700	50	SW Sequential	16 (*optimal*)	16 (*optimal*)
DUT2	14/7/4/2	50,000	37,000	700	Masked SW Sequential	14 (*optimal*)	14 (*optimal*)
DUT3	16/8/4/2	480,000	410,000	500	Masked HW Parallel	$\geqslant$8 (*sub-optimal*)	$\geqslant$8 (*sub-optimal*)

decreasing both the required number of profiling traces and required profiling time by a factor of 16 (or 14). For the masked parallel S-boxes implementation that we analyzed, the gain is slightly sub-optimal but still very noticeable for security evaluators, *i.e.*, decreasing both the required number of profiling traces and required profiling time by a factor of 8. It confirms S-box pooling can be a useful new tool for security evaluators.

5 Conclusion

During the security evaluations of symmetric algorithm implementations, usually, evaluators have to repeat the profiling of each S-box to attack them separately. The burden is getting heavier when both classical profiled attacks (such as template attacks) and deep learning attacks are mandated by the certification bodies. Additionally, the requested number of profiling traces is also increasing for some certification bodies due to many countermeasures being implemented in the evaluated products. Whether there exists a way to perform such profiling efficiently is, therefore, an important and practically motivated research question. To this end, the community so far mostly focused its efforts on improving the attack performance targeting one S-box. In this work, we analyze the complementary approach of trying to decrease the profiling (data and time) complexity thanks to S-box pooling.

Intensive experiments on three different AES implementations were performed with 108 different configurations. The results demonstrate that, for representative unmasked software sequential S-boxes implementation and the masked software sequential S-boxes implementation, using this S-box pooling approach can decrease both the required number of profiling traces and required profiling time by a factor of 16 (or 14), which corresponds to the optimal gain. For the masked hardware parallel S-boxes implementation that we analyzed, the gain is still sub-optimal in terms of data complexity and time complexity. Nevertheless, we can decrease both the required number of profiling traces and required profiling time at least by a factor of 8 in this practically relevant setting. We used only the first round S-box computations for simplicity, indeed this S-box pooling can be extended to all S-box computations of all AES rounds. The goal of the paper is to show that in some practically-relevant contexts, S-box pooling can lead to significant gains but there are admittedly implementations for which it may not be applicable, *e.g.*, if the S-boxes leak differently.

We believe these results show that the efficient exploitation of profiling measurements, for example, thanks to S-box pooling, can be a useful addition to the evaluators' toolbox. On the one hand, it is easy to adopt and integrate into existing toolchains. On the other hand, there are contexts in which this simple

optimization can lead to significant gains (remembering that concretely, a factor 16 is highly relevant for evaluation tasks that can take days).

Acknowledgements We would like to express our gratitude to anonymous reviewers for their insightful comments. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the ERC project SWORD (Grant Number 724725).

References

1. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 45–68. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_3, https://doi.org/10.1007/978-3-319-66787-4_3
2. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer (2002)
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. Lecture Notes in Computer Science, vol. 8419, pp. 253–270. Springer (2013). https://doi.org/10.1007/978-3-319-08302-5_17, https://doi.org/10.1007/978-3-319-08302-5_17
4. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4249, pp. 15–29. Springer (2006). https://doi.org/10.1007/11894063_2, https://doi.org/10.1007/11894063_2
5. Guyon, I.: A scaling law for the validation-set training-set size ratio. In: AT & T Bell Laboratories (1997)
6. Hu, F., Wang, H., Wang, J.: Cross-subkey deep-learning side-channel analysis. Cryptology ePrint Archive, Report 2021/1328 (2021), https://ia.cr/2021/1328
7. Kelley, H.J.: Gradient theory of optimal flight paths. Ars Journal **30**(10), 947–954 (1960)
8. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5_9, https://doi.org/10.1007/3-540-68697-5_9
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25, https://doi.org/10.1007/3-540-48405-1_25

10. Maghrebi, H.: Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. IACR Cryptol. ePrint Arch. **2020**, 436 (2020), https://eprint.iacr.org/2020/436
11. Masure, L., Dumas, C., Prouff, E.: A comprehensive study of deep learning for side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 348–375 (2020). https://doi.org/10.13154/tches.v2020.i1.348-375, https://doi.org/10.13154/tches.v2020.i1.348-375
12. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Dumas, C.: Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. IACR Cryptol. ePrint Arch. p. 53 (2018), http://eprint.iacr.org/2018/053
13. Quisquater, J., Samyde, D.: Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In: Attali, I., Jensen, T.P. (eds.) Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2140, pp. 200–210. Springer (2001). https://doi.org/10.1007/3-540-45418-7_17, https://doi.org/10.1007/3-540-45418-7_17
14. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: CHES. Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer (2005)
15. Standaert, F., Koeune, F., Schindler, W.: How to compare profiled side-channel attacks? In: Abdalla, M., Pointcheval, D., Fouque, P., Vergnaud, D. (eds.) Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5536, pp. 485–498 (2009). https://doi.org/10.1007/978-3-642-01957-9_30, https://doi.org/10.1007/978-3-642-01957-9_30
16. Standaert, F., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_26, https://doi.org/10.1007/978-3-642-01001-9_26
17. Zhou, Y., Standaert, F.: Deep learning mitigates but does not annihilate the need of aligned traces and a generalized resnet model for side-channel attacks. J. Cryptogr. Eng. **10**(1), 85–95 (2020). https://doi.org/10.1007/s13389-019-00209-3, https://doi.org/10.1007/s13389-019-00209-3