Towards Human Dependency Elimination: AI Approach to SCA Robustness Assessment

Unai Rioja[1,2], Lejla Batina[1], Igor Armendariz[2], and Jose Luis Flores[2]

[1] Digital Security Group, Radboud University, Nijmegen, The Netherlands
{unai.riojasabando,lejla.batina}@ru.nl
[2] Ikerlan Technological Research Centre, Arrasate-Mondragón, Gipuzkoa, Spain
{urioja,iarmendariz,jlflores}@ikerlan.es

Abstract. Evaluating the side-channel resistance of a device in practice is a problematic and arduous process. Current certification schemes require to attack the device under test with an ever-growing number of techniques to validate its security. In addition, the success or failure of these techniques strongly depends on the individual implementing them, due to the fallible and human intrinsic nature of several steps of this path.
To alleviate this problem, we propose a battery of automated attacks as a side-channel analysis robustness assessment of an embedded device. To prove our approach, we conduct realistic experiments on two different devices, creating a new dataset (AES_RA) as a part of our contribution. Furthermore, we propose a novel way of performing these attacks using Principal Component Analysis, which also serves as an alternative way of selecting optimal principal components automatically. In addition, we perform a detailed analysis of automated attacks against masked AES implementations, comparing our method with the state-of-the-art approaches and proposing two novel initialization techniques to overcome its limitations in this scenario. We support our claims with experiments on AES_RA and a public dataset (ASCAD), showing how our, although fully automated, approach can straightforwardly provide state-of-the-art results.

Keywords: SCA · Profiling Attacks · Template attacks · EDAs · Evaluation

1 Introduction

The process of integrating and validating countermeasures against Side-channel attacks (SCA) on embedded devices is known for being a complex and cumbersome task. Current certification schemes like EMVCo [12] or Common Criteria (CC [11]) assess the security of the device under test (DUT) by applying a battery of known SCA (e.g., differential power analysis (DPA) [26], correlation power analysis (CPA) [6], mutual information analysis (MIA) [16,3], template attacks (TA) [8,46,9], and machine learning-based attacks (ML-SCA) [20,19,30,31,35,45,37]). The evaluation approach is to rate each attack by considering the effort required to create and apply the attack for the first time (identification step) and once knowing the techniques developed in the identification (exploitation step) [33]. However, the ever-growing number of possible attack techniques makes it increasingly difficult to master and correctly apply all of them. This makes it challenging to perform this kind of evaluation in a low cost and efficient manner. Furthermore, the success or failure of these attacks is strongly depending on the expertise and capabilities of the attacker: He/She not only needs to stay up to date on the state-of-the-art, but also to master aspects of very different topics (statistics, electronics, signal processing, machine learning, cryptography,

programming, etc.). All these issues make the estimates of the efforts needed to compromise the security of a device quite conditional on the person implementing the tests. And given that humans are error-prone and knowledge is sometimes difficult to transfer from one person to another, technicians and product developers are facing a particularly challenging puzzle.

This problem has already been identified in the past and one of the proposed solutions are leakage assessment tests. These tests (such as TVLA [18]) attempt to eliminate the need to test devices against an accrescent number of attack vectors. They commonly use statistical tests such as *Welch's t*-test [57] or *Pearson's* $\mathcal{X}^2$-test [40], or even Deep Learning [62] or Mutual Information [10], to distinguish whether two sets of data (e.g. random vs fixed) are significantly different. These tests are used in other "conformance style" schemes like ISO/IEC 17825:2016 [21]. The problem is that, as shown in [64], assessing the SCA security of a device based on e.g., TVLA only is usually not enough, as a false positive can occur.

In addition, there also exist works that propose the usage of simulators for leakage assessment [39,29,53]. In a way, those works also share the same objective as ours, since they aim to reduce the cost of the evaluation, but the solutions are very different: evaluating the leakage before tape-out (e.g., using simulated power traces in the early stages of the design process). The main advantage is in the ability to test a chip before actually producing it. Conversely, the major disadvantage is that the simulators designed are typically not generic, as they cover specific architectures only, making them impractical. This motivates the creation of a non-hardware-specific alternative to determine the real security of a device in a simple way, as the one proposed in this paper.

In short, taking the aforementioned ideas into account, a comprehensive SCA evaluation requires attacking the device exhaustively, which is highly complex and resource-intensive. Between all types of SCA, profiling attacks (PA) are considered the most powerful, among which template attacks (TA) and ML-SCA are the most prevalent today [32]. In any case, these attacks can be quite complex and the human intrinsic nature of several parts of the operation (acquisition, preprocessing, point of interest selection, hyper-parameter tuning, etc.) utters the time and energy needed to succeed in the attack quite subjective.

In this paper, we point toward the possibility of automated attacks serving as a robustness test for a device and actually of its cryptographic implementation against TA. Our goal is to mitigate the bias and human dependency in the SCA evaluation process. For this purpose, we propose to perform automatic attacks on different cryptographic implementations to have an objective measure of their robustness against an exhaustive profiling-based attack such as TA. Note that, although DL-SCA are more recent and are becoming a method of choice, in this work we focus on template attacks as the more mature of the two, representing an established and well-understood option for the SCA community. Nevertheless, we claim that a similar strategy can be deployed for other PAs.

The most important contributions of this paper are the following:

1. We propose to use Estimation of Distribution Algorithms (EDA)-based PA automated attacks (as introduced in [48]), in an alternative and innovative way. Namely, orthogonal to [48] that purely focused on EDA-based SCA attacks, we look into another dimension by extending the method to also serve as a robustness assessment test. Our approach advocates to measure the performance of these attacks using newly (for this purpose) proposed metrics that are based on the two best known metrics in the SCA field: Guessing Entropy and Success Rate [59] and compare the robustness of several cryptographic implementations. Without

claiming it being sufficient to determine the security of a device, this test can serve as an automatic check whether it is secure against profiling attacks. We demonstrate the suitability of our method with attacks against two distinct devices (Piñata board [50] and STM32F411-Discovery board [60]). The purpose is to perform automated attacks against the SBox of different AES [13] implementations on the same device to asses its physical security. We also make our traces public, creating the AES_RA dataset [1], as a part of our contribution.
2. We propose an alternative way of performing EDA-based PA by employing it on Principal Component Analysis (PCA)-transformed power traces, rather than on the "raw" traces. We compare this procedure with the "raw" procedure using the random probability initialization for the EDA as proposed in [48] and our two novel initialization methods. To this end, we perform a detailed analysis of automated attacks on masking-protected AES software implementations, comparing the proposed alternatives with the "standard" attacks, and showing the advantages and disadvantages of each method. Our results show that PCA can accelerate the process when the power traces are clean, as the number of relevant time samples in the EDA is decimated. Thus, the number of variables involved in the EDA is also drastically reduced.
3. Moreover, this novel combination of EDA-based PA and PCA serves as an alternative way of selecting the number of principal components (PCs) to keep. Our technique works as a simple and automatic way of selecting not only the number of PCs to keep but also which PCs give the best results. As "there is no definitive answer [to the question of how many components to choose]" [23], we claim that it is an appealing choice when employing PCA in SCA or in some other field. We showcase the performance of our proposal with experiments on the aforementioned AES_RA and a widely used dataset in the field of SCA (ASCAD [4]), providing state-of-the-art results. We compare several EDA-based PAs against "traditional" (not automated) template attacks using PCA for the Point of Interest (POI) selection, showing the advantages of this method.

The paper is organized as follows. Sect. 2 summarizes the important background and related works on this topic. In Sect. 3 we describe our proposed robustness assessment test and the metrics employed for assessing the performance of the EDA-based attacks. We introduce our novel AES_RA dataset in Sect. 4. We specify the procedure of EDA-based Robustness Assessment in Sect. 5, providing experimental results supporting our approach (Contribution 1). In Sect. 6 we elaborate on our alternative EDA-based attack using PCA and the proposed initialization methods when attacking masked-protected AES implementations (Contributions 2 and 3). Sect. 7 contains the experimental results supporting the alternatives proposed in the previous section. Finally, Section 8 concludes the paper.

2 Background and Related Work

In this section, we first present previous works which are relevant for this paper. Afterwards, we give a brief explanation of the background necessary to understand our work.

2.1 Related Work

Here we describe some related previous works on the topics around which the contribution of this paper is framed: automated SCA and the use of PCA in SCA.

Automated SCA: To the best of our knowledge there is hardly any work that aims to automate several of the phases of an SCA and thus mitigate human dependency. The works closest to ours are [48,49]. In those papers, the authors propose stochastic optimization techniques to perform and optimize several steps of a conventional profiling attack (POI selection, template building, and key recovery), relaxing the need for human interaction. However, they do not take into account some important concepts such as the generalisation of the obtained templates [67]. In this paper, we do not only consider these factors but we go a step further and advocate that these automated attacks could serve as an objective way of assessing SCA resistance. Besides, we highlight the importance of selecting a good probability initialisation method for the EDA approach by systematically comparing the performance of different, including the two new, proposals.

PCA: PCA is a statistical technique that computes the PCs and uses them to perform a change of basis on the data. It is commonly used as a dimensionality reduction technique by keeping only the first few PCs and ignoring the rest. Furthermore, in the field of SCA, it has been used for very different purposes. The first appearance of PCA in the field of SCA was its usage as a method to improve power attacks [5]. Later on, PCA was used as a POI selection technique for template attacks in [2]. Afterwards, it has been used for POI selection in profiling attacks in a large number of works (e.g., [58,51,42,17,63,41,4,14]).

Other works try to enhance PCA (among other dimensionality reduction methods) for SCA [7] or compare PCA against other POI selection techniques [44]. PCA has also been used as a preprocessing technique to improve the correlation for the correct key candidate [38]. In [55] the authors followed a different approach and used PCA not as a preprocessing technique, but rather as a common side-channel distinguisher. In any case, our approach is very different from all those papers as we use PCA to improve the performance of the automated TAs (EDA-based PA).

Furthermore, we claim that this approach also serves as an automated way of selecting optimal PCs. The choice of a proper number of PCs to keep is crucial for obtaining favourable results. There exist several "traditional" ways to obtain the number of PCs needed, as shown in [24]. Generally speaking, they rely on selecting the largest PCs (e.g., Scree test and Cumulative Percentage of Total Variation). The problem is that, as several related works underline [38,56], when a first few components are selected to reduce the dimension of the data, often the first ones contain more noise than information. This is because the first components contain the most variance, but since PCA does not take leakage information into account, that variance can come from leakage or be mere noise. Therefore, the task of choosing not only the number of components but also which particular components to keep is a complex and application-dependent task. To the best of our knowledge, there are no related works that attempt to do this task in a simple, automated and generalized way. There exist only a few works that propose selection methods for PCA in SCA [38,7], but they have the same drawback as they rely on the variance of the traces and not on its leakage. In [38], the authors propose to compute the Inverse Participation Ratio (IPR) score and collect the PCs in decreasing order accordingly. In [7] authors suggest a new technique (Explained Local Variance, ELV) based on the compromise between the variance provided by each PC and the number of samples necessary to achieve a consistent part of such variance. Both those approaches are complex and human-dependent, unlike ours. Another strength of our method is that it can be employed in masking-protected traces following a "Black-Box" approach (i.e., without knowing the mask), even in high-noise environments.

2.2 Notation

In this section we briefly define the notation used throughout the paper. We adopt the notation introduced in [36], with some adjustments. $\mathbf{T}$ denotes a set of traces $\mathbf{t}$. Each power trace is composed of T time samples $\mathbf{t} = \{t_1, t_2, t_3 \ldots, t_T\}$. The total number of power traces $\mathbf{t}$ in a set of traces $\mathbf{T}$ is denoted by $|\mathbf{T}|$. We use $v = f(p, k)$ for the targeted intermediate value, which is related to a public variable (plaintext p) and a cryptographic primitive (secret key k). $\mathbf{K}$ denotes the set of all possible keys. k^* denotes the (correct) key used by the cryptographic algorithm and the total number of key hypotheses is denoted by $|\mathbf{K}|$. Regarding TAs, we denote each template by $h = (\mathbf{m}, \mathbf{C})$, where $\mathbf{m}$ and $\mathbf{C}$ denote mean vector and covariance matrix, respectively.

2.3 Template attacks

Template Attacks (TAs) were proposed in [8] and represent the first form of profiling attacks, the strongest kind of SCA nowadays. In these attacks, the general idea is to generate a power consumption model to compare it with the actual power consumption of the device and recover sensitive information (i.e., cryptographic keys). There exist different types of profiling attacks depending on how the model is generated. Whereas template attacks use estimation theory to model the probability distribution of the leakage [8,46], other procedures use linear regression (stochastic models approach [52]) or machine learning [20,19,30,31], including the lately introduced tendency of using deep learning techniques [4,25,44] to build the leakage model.

In practice, TAs are commonly used to recover the secret key used by the DUT to perform cryptographic operations. In order to do so, the attacker has to capture a big number of power traces of the DUT while it manipulates some intermediate value $v = f(p, k)$. This intermediate value is related to a known variable (usually the plaintext p) and the secret key k. As the plaintext is known, guessing the intermediate value enables the attacker to recover the secret key.

Then, in the first stage (*profiling phase*) a set of ($\mathbf{T}_p$) profiling traces are used to build a Gaussian multivariate model for each possible intermediate value v, creating the so-called *templates* (denoted by h).

After that, in a second stage (*attack phase*), the attacker uses a set of ($\mathbf{T}_a$) attack traces and its input/output data (plaintext/ciphertext). This information is employed to guess the correct secret key (k^*) by making a hypothesis about its value and computing all possible intermediate values. Then, a *discriminant score* $D\left(k_j \mid \mathbf{t_i}\right)$ is calculated for each key hypothesis k_j and the key hypothesis are ranked in decreasing order of probability. Given a power trace $\mathbf{t}_i$, a commonly used discriminant derived from Bayes rule is $D\left(k_j \mid \mathbf{t_i}\right) = p\left(\mathbf{t_i} \mid k_j\right) p(k_j)$ This discriminant is obtained by omitting the denominator from Bayes' rule, since is the same for each key hypothesis k_j [36,9].

Finally, the attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, ..., g_{|K|}]$, in decreasing order of probability. We are assessing the performance of the attack by using an SCA-specific metric (Guessing Entropy, GE [59]). The guessing entropy is the average position of the correct key k^* in the key guessing vector over multiple experiments. The higher GE value, the more difficult it would be for an attacker to guess the correct key.

To conclude, TAs are optimal from an information-theoretic point of view, but they have several limitations in practice, namely computational complexity problems and the need for dimensional reduction being the most critical ones [9]. The dimensionality reduction is usually selecting a small number of time samples of the power traces (POIs selection [46]), or using a more complex method like Principal Component Analysis (PCA) [2,58] or Fisher's Linear Discriminant Analysis (LDA) [22,15]). Note that, with EDA-based PA, the POI selection is done automatically by the algorithm [48].

2.4 Principal Component Analysis

Principal Component Analysis (PCA) is a widely used statistical technique usually employed to reduce noise or dimensionality in a dataset. This technique is based on computing Principal Components (PCs), derived as linear combinations of the original variables. The most common way to implement this technique is the following [54]:

- **Step 1:** A mean vector $\mathbf{m}$ is calculated, which includes the mean for each of the T dimensions (time samples per traces) of the traces $\mathbf{T}$. Then, the mean is subtracted from each of the T dimensions of each trace t_i.
- **Step 2:** A covariance matrix Σ is constructed. In such a matrix, each (i, j)th element is the covariance between the ith and the jth dimension of the power traces. Thus, the covariance matrix will be a $T * T$ matrix, where T is the number of dimensions (number of samples of the power traces). It should be noted that the computation time increases quadratically relative to the number of samples, as the main shortcoming of this method. The covariance of two dimensions $\mathbf{X}$ and $\mathbf{Y}$ is defined by the following formula:

$$\mathbf{C}(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{n} \left(\mathbf{X}_i - \bar{\mathbf{X}}\right)\left(\mathbf{Y}_i - \bar{\mathbf{Y}}\right)}{n-1}$$

  Where n is the number of elements in both dimensions, $\mathbf{X}_i$ and $\mathbf{Y}_i$ are single elements of $\mathbf{X}$ and $\mathbf{Y}$ respectively, and $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are the sample means of each dimension.
- **Step 3:** The eigenvectors and eigenvalues of the covariance matrix are computed by $\mathbf{\Sigma} = \mathbf{U} * \mathbf{\Lambda} * \mathbf{U^{-1}}$, where $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix and $\mathbf{U}$ is the eigenvector matrix of $\mathbf{\Sigma}$. These matrices provide information about patterns in the power traces. The direction with the most variance coincides with the eigenvector corresponding to the largest eigenvalue ("first principal component"). As T eigenvectors can be derived, there are T PCs that must be ordered from high to low eigenvalue.
- **Step 4:** Then, a number of p PCs can be selected (to reduce the dimensionality of the dataset), building a matrix with these vectors as columns (feature vector). Note that we can also choose to select all the PCs and just transform (i.e., make a change of basis of) the data, as we do in this paper.
- **Step 5:** Once this feature vector $\mathbf{U}^p$ of length p is generated, the original data can be transformed to retain only p dimensions (samples). In order to do so, we can transpose the feature vector $\mathbf{U}^{p'}$ and multiply it with the transposed mean-adjusted data $\mathbf{X}^{'}$, obtaining the transformed dataset $\hat{\mathbf{X}}$:

$$\mathbf{Y} = \mathbf{U}^{p'} * \mathbf{X}^{'} = (\mathbf{X} * \mathbf{U}^p)^{'}$$
$$\hat{\mathbf{X}} = \mathbf{Y}^{'} = \left((\mathbf{X} * \mathbf{U}^p)^{'}\right)^{'} = \mathbf{X} * \mathbf{U}^p$$

2.5 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) are stochastic optimization techniques that search for likely solutions by building explicit probabilistic models of promising candidates. Unlike other evolutionary algorithms whose immense amount of tunable parameters makes it hard to work with (e.g., genetic algorithms), the main advantage of EDAs is their simplicity: they involve a much smaller number of tunable parameters as the new population of individuals is generated from a probability distribution obtained from the best individuals of previous populations [48,34,28].

Estimation of Distribution Algorithms in Side-Channel Analysis EDAs were proposed in [48] in combination with template attacks as a way to perform the POI selection step together with the profiling and key recovery steps. This provides for automated optimisation of the attack, avoiding the need to manually perform various types of analyses with different POI combinations. As an exhaustive enumeration of all combinations is exponential and definitively not feasible, the approach is to use a search strategy based on a quality measure combined with this modern and efficient evolutionary computation algorithm. Fig. 1 shows a graphical representation of the process.

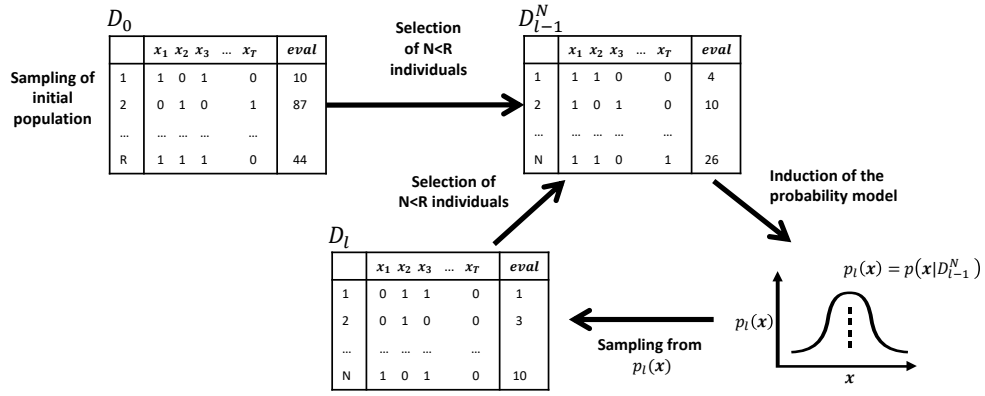


Fig. 1. Illustration of a generic EDA-based PA

First of all, an initial population D_0 of R individuals (POI selection candidates) is generated from a specified probability distribution. To this end, a vector of binary variables of length T (number of samples per trace) is considered:

$$\mathbf{x} = \{x_1, x_2, \ldots, x_n, \ldots, x_T\} = (0, 1, 0, \ldots, 0)$$

Each variable matches with one sample of the power traces and its probability represents the probability of that sample of being selected for the template building. As in [48], we consider that there are no interrelations between the variables, and the probability distribution can be learnt as:

$$p_l(\boldsymbol{x}) = \prod_{i=1}^{T} p_l(x_i)$$

This probability distribution can be initialized at random or based on some criterion, i.e., based on the leakage correlation [48]. Then, these subset D_{l-1}^N of N individuals are evaluated (R attacks are performed with the R candidates). After that, the probability distribution $p(\mathbf{x})$ of promising candidates is estimated from the marginal frequencies of the highest quality solutions (D_{l-1}^N):

$$p_l(x_i) = \frac{\sum_{j=1}^{T} \delta_j(X_i = x_i | D_{l-1}^N)}{T}$$

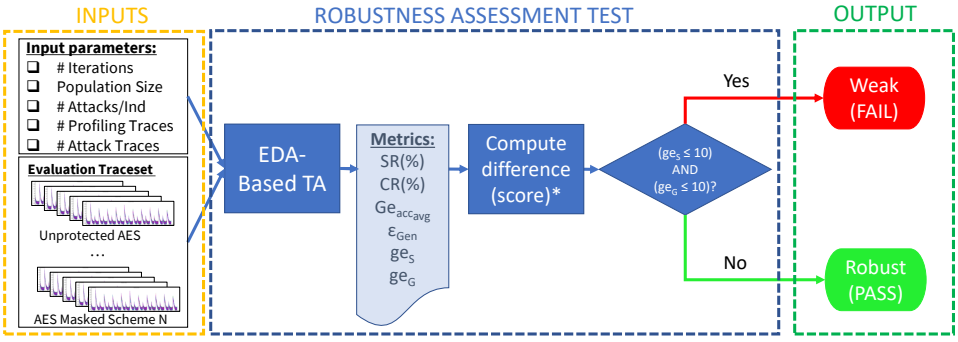
Where:

$$\delta_j(X_i = x_i | D_{l-1}^N) = \begin{cases} 1, & \text{if in the } j\text{-th case of } D_{l-1}^N, X_i = x_i \\ 0, & \text{otherwise} \end{cases}$$

That is to say, the probability of each time sample of being selected as POI for building the leakage model is recomputed based on previous results. Then, a new population D_l^N is sampled and the process is repeated in a new iteration until a stop condition is reached. For a deeper explanation of the attack, we refer to [48].

3 Robustness assessment test

In this section, we describe our approach for the robustness assessment test and the proposed metrics. Fig. 2 shows a schematic of the process. In a nutshell, the idea to perform a battery of automated attacks is in using EDAs and compute the metrics as described below. If the attacks are successful and the model is generalizable, we conclude that the implementation is weak against PAs. We also compute a score to quantify how weak it is compared to an unprotected implementation. The larger value it is, the more difficult gets to recover the secret key for an attacker (see Fig. 2).



*score: Compute difference between metrics (Unprotected vs Masked Scheme N) to quantify/compare

$$Diff\ Score(\%) = \frac{|SR_U - SR_M| + |CR_U - CR_M| + |(ge_{acc_U} - ge_{acc_M})/256| + |\varepsilon_{Gen_U} - \varepsilon_{Gen_M}|}{4}$$

Fig. 2. Robustness assessment test scheme

3.1 Metrics

To compare the performance of different EDA-based PAs (and/or perform the robustness assessment test), we propose to compute four simple metrics. These metrics give us information about the performance of the obtained models and how difficult it would be for an attacker to recover the secret key. They rely on the two more established metrics in the SCA field nowadays [59]: Guessing Entropy and Success Rate.

– **Success rate [SR](%):** When executing an automated attack, an important factor is how accurate the algorithm has been in executing the attacks. To determine this, we propose a modified version of a widely used metric in the SCA field: the success rate [59]. Generally speaking, the success rate of order "o" is the average empirical probability that the correct key candidate is located within the first "o" elements of the key guessing vector.
In our case, we compute a modification of the success rate of order 10, i.e. we divide the number of successful attacks ($GE \leq 10$) by the number of attacks

performed by the EDA. This metric helps us to compare the efficiency of different EDA-based attacks, as we clearly see how certain the EDA-based attack has been. A high SR indicates that the proven implementation is not particularly secure as the algorithm managed to succeed effortlessly. We compute this metric as:

$$SR = \left(\frac{n_{Success}}{n_{Attacks}} \right) \times 100 \tag{1}$$

Where $n_{Success}$ is the number of successful attacks and $n_{Attacks}$ is the total number of attacks.

- **Convergence rate [CR](%):** Another relevant factor is the effort it takes the EDA to achieve successful results. For this we define a metric to assess the number of attacks/iterations of the EDA until the first success.
  Therefore, we propose to divide the number of attacks required until a successful attack is obtained by the total number of attacks (i.e., we measure the number of trials needed to get one success).If we succeed in the first iteration we will see a very high convergence rate. The more attacks it takes the less convergence rate we get. We compute it as follows:

$$CR = \left(1 - \frac{n_{Trials}}{n_{Attacks}} \right) \times 100 \tag{2}$$

  Where n_{Trials} is the number of trials before a successful attack and $n_{Attacks}$ is the total number of attacks.
- **Averaged cumulative final guessing entropy [$ge_{acc_{avg}}$]:** This metric shares goal with the SR, but as SR is quantitative (we take into account whether the attacks are successful or not) we also wanted to compute a qualitative metric that complements the previous one. Since the Guessing Entropy [59] does not quantify whether the attack has been successful or not, but rather how close we are to the optimal solution, it is a perfect candidate for this purpose.
  We therefore propose to use a modified version of this metric that fits the particular needs of this scenario. To calculate it, we simply divide the cumulative final guessing entropy value of the attacks by the total number of attacks:

$$ge_{acc_{avg}} = \frac{\sum_{\mathrm{i}=0}^{n_{Attacks}} \mathrm{GE_i}}{n_{Attacks}} \tag{3}$$

  Where $n_{Attacks}$ is the total number of attacks and GE_i corresponds with the final GE value of the ith attack. This gives us an estimate of how hard it is to obtain a correct GE value.
- **Generalization Error [ε_{Gen}](%):** The goal of this metric is to ensure the applicability of the obtained models, and verify that they are employable in a real attack scenario. In traditional profiling attacks, techniques to avoid overfitting and enhance generalization are usually not contemplated. Today, thanks to the increasingly established trend of ML-SCA, these concepts are becoming more prevalent [67].
  Hence, in this paper, we take into account the generalisation of the models by using a specific measure. The idea is to apply the templates built by our EDA to unseen data, and thus test their performance. To do so, we execute a battery of N attacks over the unseen data (using the optimized model) and compute its averaged final guessing entropy ge_G. We then calculate the difference between this and the averaged final guessing entropy obtained with that model during the searching phase ge_S. Finally, we compute the relative error between these two values as:

$$\varepsilon_{Gen} = \frac{ge_S - ge_G}{ge_{max}} \times 100 \tag{4}$$

Where ge_{max} is the maximum (worst-case) GE. In this case, as we are targeting 8-bit values and the worst-case is 256. If the generalisation error is high it means that, although we succeeded during the search of the model, the templates are not applicable in practice and therefore the attack cannot be considered successful.

4 The AES_RA dataset

In this section, we briefly describe our new AES_RA dataset [1]. Most of the results in relevant previous works mentioned above have been obtained using ASCAD in their experiments. However, although we have used ASCAD too for the sake of comparison (see Sect. 7), we also introduce an additional dataset: AES_RA. The motivation is that we wanted to tackle a more complicated problem, with noisy real-world traces collected from an actual device on the field. In addition, AES_RA fills the gap of an extensive dataset including traces from different AES implementations on the same DUT.

Thus, this dataset contains traces from two different embedded systems which use microcontrollers from the same family. With each device, we acquire traces from three AES implementations: an unprotected software AES and two different masking schemes, resulting in six different setups. Thus, this dataset is divided into two parts: power consumption traces from the Piñata board and capacitor EM power traces from the STM32F411E-Discovery Board. We believe that this dataset, together with ASCAD, allows us to validate our approach comprehensively.

4.1 AES implementations

Three different AES software implementations have been considered. There is a brief explanation of each one of them is given in the sequel.

- **Unprotected AES:** typical AES-128 (in ECB mode) software implementation [27].
- **Masking Scheme 1 (Weak):** A modification of the previous one which matches the same masking method as described in [36] (Masked Lookup Table). In this implementation, the output mask of the SBox operation is removed after each 1-Byte lookup and hence we see a clear correlation of the mask in the SBox time window (See Fig. 3 below). This makes the scheme similar to the one used in ASCAD, as can be observed in its pseoudocode [4]. As we show in the experiments below, the fact that the output mask is leaking during the computation of the SBox makes this implementation very weak against TAs (and PAs in general).
- **Masking Scheme 2 (Robust):** A modification of the previous one, but this time the output mask is removed after the MixColumns operation. This means that, unlike in the previous scheme, the output mask does not leak during the SBox computation, making the scheme much more robust against TAs as we will see below.

For the pseudocode of both masking schemes and more information about the dataset organization, please see the AES_RA GitHub [1].

4.2 Piñata Board

Piñata is a development board created by Riscure based on an ARM Cortex-M4F core working at a 168 MHz clock speed [50]. It has been physically modified and programmed to be a training target for SCA and Fault Injection. We measure the power consumption of the board during the AES encryption with a Tektronix CT1 current probe attached to a 20 GS/s digital oscilloscope (LeCroy Waverunner 9104)

triggered by the microcontroller, which rises a GPIO signal when the internal computation starts. Each power trace consists of 1 260 samples (1 500 and 1 800 for the masked implementations 1 and 2 respectively) taken at 1 GHz with 8-bit resolution, corresponding to the first SBox operation.

4.3 STM32F411E-DISCO Board

The STM32F411E-DISCO is a development board with an STM32F411VE [61] high-performance Arm® Cortex®-M4 32-bit RISC microcontroller working at 100 MHz. This board (STM32F411E-DISCO) is similar to Piñata (microcontrollers are from the same family), and uses exactly the same code. We measure the power consumption of the board during the AES encryptions with a Langer EM probe over a decoupling capacitor (C38) attached to the oscilloscope (LeCroy Waverunner 9104), wich again is GPIO-triggered by the microcontroller. Each power trace consists of 1 225 samples (1 500 and 1 800 for the masked implementations 1 and 2 respectively).

5 Robustness Assessment test on AES_RA

In this section, we show how the aforementioned attacks could be employed as a robustness assessment test to evaluate the robustness of a device against profiling attacks (template attacks more specifically). Hence, following the scheme from Fig. 2, we perform three EDA-based attacks over three distinct AES implementations: unprotected software AES, AES with masking scheme 1 (Weak), and AES with masking scheme 2 (Robust). As mentioned in Sect. 4, the main difference between masking schemes 1 and 2 is that, due to their implementations, on the former we see a clear correlation with the mask in the targeted time window (SBox) whereas in the latter not. A graphical representation of this fact can be observed in Fig. 3. We repeat this robustness assessment approach two times with two different boards (Piñata and Discovery) and different probes/measures (current and capacitor EM probes respectively).

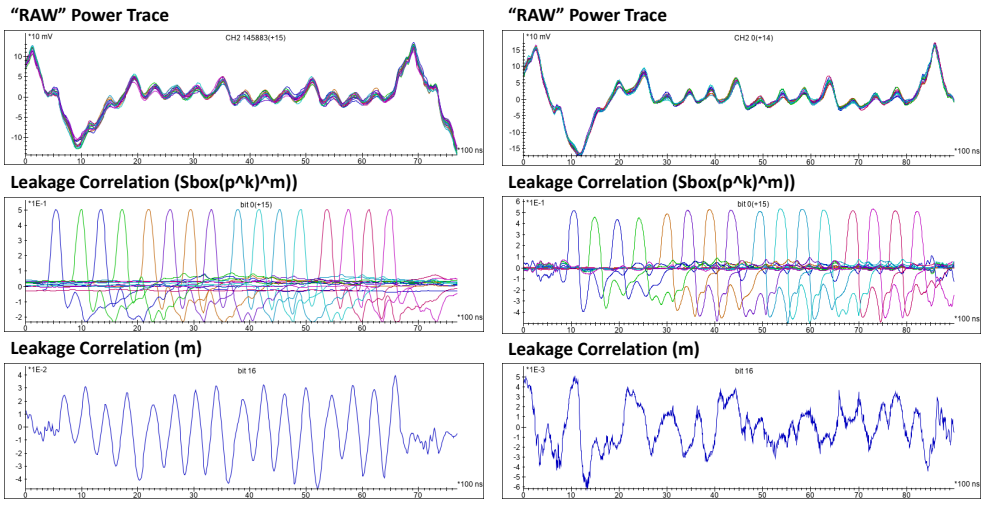


Fig. 3. Piñata board: Leakage of Masking Scheme 1 (left) vs Masking Scheme 2 (right). For all graphics, X-axis: Time ($\times 100ns$), Y-axis: Voltage ($\times 10mV$)/Correlation

5.1 Experimental Results on Riscure Piñata board

We perform three different EDA-based attacks over the three implementations and compute the metrics. The results of the robustness assessment test on the SBox of the three different AES implementations are shown in Table 1. Each row represents a metric (SR, CR, $ge_{acc_{avg}}$ and ε_{Gen}), except for the last two rows, which are the averaged final guessing entropy of the best model over known data ge_S and unseen data ge_G (employed to compute ε_{Gen}). For the EDA parameters, we are using 30 iterations and 50 individuals per population. If all the attacks of one iteration are successful, we stop the EDA process. Since we are evaluating the leakage, we are following a "White-Box" initialization (as explained in Sect. 6.2). Regarding the TA, we are using 20 000 profiling traces (50 000 profiling traces for masking scheme 2 for being a more challenging attack) and 2 000 attack traces.

Metric	**Unprotected AES**	**AES masking scheme 1**	**AES masking scheme 2**
SR (%)	100%	100%	29.63%
CR (%)	100%	100%	52.5%
$\mathbf{ge_{acc_{avg}}}$	1	1.18	52.28
$\varepsilon_{\mathbf{Gen}}$ **(%)**	0%	0%	38.11%
ge_S	1	1	1
ge_G	1	1	84
Test Results		**Weak (Fail)**	**Robust (Pass)**
Diff Score		**0.02%**	**44.00%**

Table 1. Robustness assessment: Piñata

From this test, we can conclude that the masking scheme 1 does not provide any security to the SBox as the results of the attacks are almost the same as the unprotected implementation: we succeed in all the attacks since the first one (SR and CR are in their maximum values and $ge_{acc_{avg}}$ is almost 1) and the generalization of the models is perfect ($\varepsilon_{Gen} = 0$). In the AES masking scheme 2 there is no clear leakage of the output mask (in the time window we are targeting), and hence the model has an especially poor generalization (we do not succeed in this attack, $ge_G = 84$). This makes sense since, as stated in [32], when the mask value is unknown to the attacker during the profiling step, the leakages associated with a key follow a multimodal distribution. This leads to assumption errors whether the adversary exploits Gaussian template attacks. Nevertheless, as highlighted in [69], when the mask leakage is included in the observation time window, the templates are able to relate the dependence between the mask and the masked variable leakage. This explains why we succeed with Masking Scheme 1 but not with Masking Scheme 2 (large generalization error). However, we claim that the fact that we succeed in the searching process shows that there is still some leakage, which may be better exploited in the future. To conclude, note that these two masking implementations are susceptible to a second-order attack, which combines the leakage of two bytes of the key at a time when the mask is removed [36].

5.2 Experimental Results on STM32F4 Discovery board

As we show later in Sect. 7, the traces from STM32F4 have much more noise from the environment than the previous ones with Piñata (due to the acquisition method). Nevertheless, the leakage is still present, as can be observed in Fig. 4, where the difference in the leakage between the two masking schemes in this board is shown.

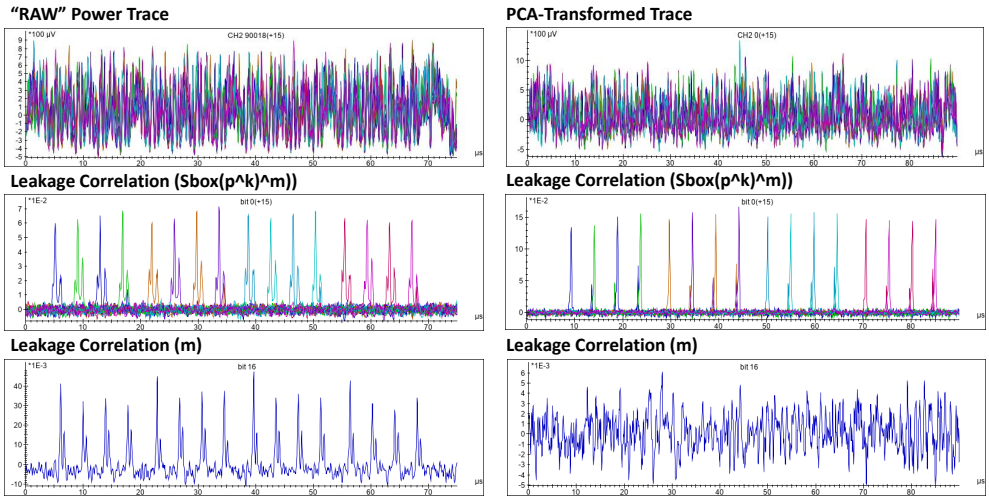


Fig. 4. STM32F411 board: Leakage of Masking Scheme 1 (left) vs Masking Scheme 2 (right). For all graphics, X-axis: Time (μs), Y-axys: Voltage ($\times 100 \mu V$)/Correlation

.

Table 2 shows the results of the robustness assessment test over the three AES implementations. We are using the same EDA parameters as in the previous case. Regarding the TA, we are using 50 000 profiling traces (100 000 profiling traces for Masking Scheme 2 for being a more challenging attack) and 2 000 attack traces.

Metric	**Unprotected AES**	**AES masking scheme 1**	**AES masking scheme 2**
SR (%)	64%	66%	31.14%
CR (%)	98.8%	96.8%	49.43%
$\mathbf{ge_{acc_{avg}}}$	13.48	25.20	43.17
$\varepsilon_{\mathbf{Gen}}$ **(%)**	0%	0%	52.89%
ge_S	1	1	1
ge_G	1	1.33	136.4
Test Results		**Weak (Fail)**	**Robust (Pass)**
Diff Score		**2.14%**	**36.68%**

Table 2. Robustness assessment: STM32F4

From this test, we can obtain similar conclusions to the previous one, which is not unexpected given that the same AES implementations are being used. Again, Masking Scheme 1 does not provide any security against TA since we are achieving nearly the same result as attacking the implementation without countermeasures. In contrast, Masking Scheme 2 does provide a high level of protection: not only obtaining a model that works on the search set is much more difficult, but the generalization of the model, in this case, is even worse than in the previous one (we obtain a ge_G of 136.4).

6 EDA-based attacks on masked AES implementations

In this section, we go into more detail about performing automated attacks on masked implementations using EDAs. In other words, this section describes our second and

third contributions: our alternative way of using EDA-based PA combined with PCA and our proposed EDA's probability distribution initialization methods.

6.1 Combining EDA-based PA with PCA

In order to accelerate the EDA-based PA process, in this paper we propose to preprocess the traces using PCA before launching the EDA-based PA. Although this implies a higher degree of complexity (as PCA is computationally expensive), this has several advantages that can justify its usage in some applications. The reason is that, if PCA behaves correctly, all the relevant information will be gathered on the first PCs. This can be used to reduce the number of varaiables (time samples) in the EDA-based PA.

An example of how PCA behaves in practice can be observed in Fig. 5. On the left side of the figure ("Raw" power traces), we can observe that the leakage correlation of the mask and the masked intermediate value appears on two distinct zones. Note that in this dataset (ASCAD [4], as explained below) each power trace has 1 400 time samples. Thus, if we follow the approach of [48] and use a uniform initialization of probabilities, it will take time for the EDA to find the right time samples as each one of the 1 400 samples has the same probability of being selected (see Sect. 7.1). Conversely, if we observe the right side of Fig. 5 (PCA-Transformed traces), we can see how the relevant leakage information is congregated on the first PCs. This allow us to accelerate the search, as we can consider only a number of first PCs for the EDA-based PA, and hence reduce the complexity of the probabilistic model of the EDA (i.e., number of variables involved). Therefore, the EDA will find proper POIs (PCs in this case) more efficiently, as we show in Sect. 7.

In addition, this approach serves as an automated alternative for selecting not only the number of PCs to keep but also which ones in particular. As shown below, in the experiments, there usually exist some PCs that not only do not provide any relevant information to the model, but their inclusion negatively affects its performance. In addition, there is usually a tipping point beyond which the results worsen if we add more PCs. This appropriate number of relevant PCs is complex to find manually in practice (especially for the less experienced). It should be noticed that, as mentioned in Sect. 2 there exist other methods for selecting the number of PCs to keep. The problem is that the success or failure of these techniques depends significantly on the application and the technician implementing them. In contrast, we claim that our approach can find optimal PCs effortlessly. Nevertheless, for the sake of comparison, a "traditional" TA has been conducted (i.e., without the usage of the EDA-based PA approach). As in a number of related works [4,25,69], we perform the POI selection by using PCA and selecting different numbers of PCs to accomplish the attack.

6.2 EDA's probability distribution initialization

As mentioned in Sect. 2.5, when performing a EDA-based PA, different strategies can be followed for setting the initial probability distribution (i.e., probabilities of each time sample of being selected). As we show in experimental results, how we initialize the probabilities of the EDA has a strong impact on the attack results. Thus, apart from comparing the "raw" and "PCA" approaches, we also consider different initializations for each one. In this work, we consider the random initialization method proposed in [48] and two novel approaches. Table 3 summarizes the details of each one of them.

In a nutshell, given the limitations of the initialisation method proposed in [48] (Random Uniform in Table 3) when attacking masking implementations, we propose two alternatives for this case. As explained before, this approach is not optimal for

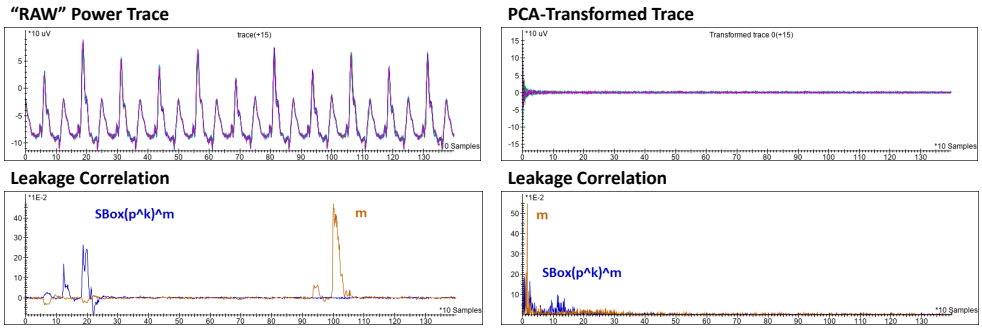


Fig. 5. ASCAD: Raw power traces (left), PCA-transformed traces (right) and leakage correlation. X-axis: Time Samples ($\times 10$) Y-axis: Voltage ($\times 10\mu V$)/Correlation

Acronym	Initialization method	Type	Description
Rndm	Random uniform	Black-Box	Each sample has the same probability of being selected initially.
Dec	Decreasing probabilities	Black-Box	Used with the EDA+PCA approach. First samples have a higher probability of being selected. The further away they are from the first PC, the less probability they have.
wPOI	Using leakage information	White-Box	"Leakage correlation" graphics (POI) are used to initialize the probabilities and guide the EDA to the areas where the leakage is located.

Table 3. Considered initialization methods for EDA's initial probability distribution

this use case as we do not give any information to the EDA about where the leakage is located and it will take time for the EDA to find the leaking time samples. Thus, we propose two alternatives: Decreasing Probabilities (for PCA-Transformed traces only) and a "White-Box" approach in which we initialize the probabilities using the correlation of the unmasked intermediate value $SBox(p \oplus k) \oplus m$. Note that this approach was used in [48], but only with unprotected implementations, as masking randomizes the intermediate values making the correlation with the intermediate values null. In this work we propose to employ this approach also with masked implementations. We consider this a "White-Box" approach as, contrary to the other two cases, we need to know the mask m to compute the unmasked intermediate value $SBox(p \oplus k) \oplus m$.

7 Experimental results on masked AES implementations

In this section, we compare the performance of different EDA-based attacks on different datasets, including the modifications proposed in the previous section. We first perform various attacks over a public dataset to demonstrate that our approach can provide state-of-the-art results without human intervention. Then, we conduct the same analysis over our novel AES_RA dataset [1]. Finally, we draw some conclusions about the experiments.

7.1 Results on a public dataset

For demonstrating our approach, apart from our own dataset, we have employed a widely used dataset in the SCA field: ASCAD (Variable Key). We first perform a

"regular" TA using PCA for the POI selection. Then, we perform different automated attacks, with the settings explained in the previous section.

The ASCAD dataset ASCAD [4] was the first open database for DL-SCA and includes electromagnetic emanation traces of an 8-bit AVR microcontroller (ATmega8515), implementing a masked AES-128 implementation (see [4]). The dataset is divided into two parts: fixed key and variable key. Although many related works use the fixed key version for being an easier problem [25,69,68,65], for this work we are using the variable key version. This allows us to perform a more realistic use case, as we can use random keys for the profiling step and a fixed key for the key recovery step, as an attacker would do in practice. The data set provides 300 000 traces where 200 000 are used for profiling (variable key) and 100 000 are used for the attack (fixed key). These traces contain a window of 1 400 relevant raw samples per trace, representing the third byte of the first round masked S-Box operation (See Fig. 5). For a deeper explanation of the ASCAD dataset, we refer to [4]. As the sensitive intermediate value we use an S-box output: $Y^{(i)}(k^*) = \text{SBox}[P_3^{(i)} \oplus k^*]$.

Experimental Results on ASCAD Table 4 summarizes the parameters of the automated attacks. As mentioned before, the selection of a proper number is not so straightforward. Figure 6 (left) shows the results of several attacks using a different number of PCs. Generally speaking, adding more PCs has a good effect on the results until we reach a point (around 25 PCs) in which the addition of more makes the attack not feasible. It should be noticed that, if we follow the Scree Test approach (classical PC selection method [24]), and we plot the eigenvalues to manually inspect where the curve changes from a steep line to a straight line (elbow), the relevant information is supposed to be on the 15 first PCs. Nevertheless, we obtain better results with 25 PCs. If we observe Fig. 5 (right) we can understand why. Although the biggest part of the leakage of m is concentrated around the 15th PC, the leakage of the unmasked intermediate value $SBox(p \oplus k) \oplus m$ spreads around the first 200 PCs.

Parameter	# Profiling Traces	# Attack Traces	# Attacks/Ind	# Iterations	Pop. Size
Setting	20 000	1 000	4	10	50

Table 4. Parameters of the attack (ASCAD)

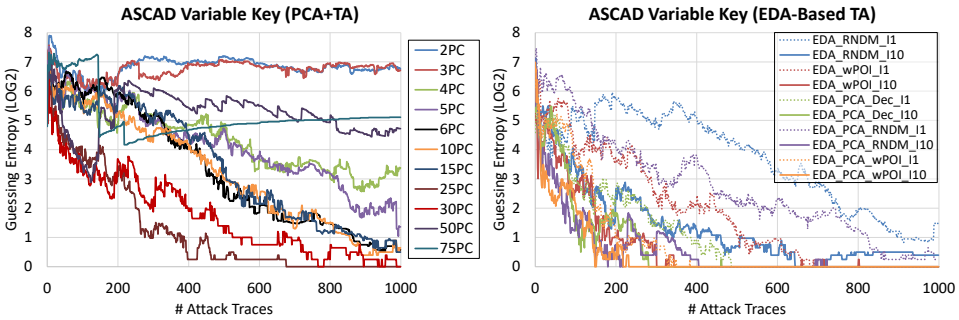


Fig. 6. Attacks on ASCAD: regular PCA + TA (left) and EDA-based TAs (right)

At this point we compare two types of attacks using EDAs, one over "raw" traces (EDA in figures) and one over "PCA-Transformed" traces (EDA_PCA in figures), using different initialization approaches (see Table 3). Fig. 6 (right) show the results of the best candidate of the first and last iteration of each approach.

If we observe the results of the metrics defined above (Table 5), the improvement of using the EDA+PCA approach in this dataset can be observed. In general, the attacks using EDA+PCA are more efficient and achieve better results than using EDA only. The improvement in the SR and CR shows that with EDA+PCA the procedure is more efficient (we succeed earlier and in more attacks). The same happens in terms of guessing entropy, $ge_{acc_{avg}}$ is lower in the EDA+PCA case, as we succeed in more attacks. If we observe Fig. 6 (right), we can see that the attacks using EDA+PCA have a very good performance, with its guessing entropy converging around 200 traces. The EDA attack using the "White-Box" initialization has a very good performance too, but the attack with random initialization is less effective. Nevertheless, all attacks are successful and show a good generalization ability. In addition, our approach is able to reduce the number of traces needed for the secret disclosure from 400 to 200, when compared to the PCA+TA approach. In the best case, we manage to recover the key with around 100 traces, a state-of-the-art result in this dataset (ASCAD with random keys), when comparing with other related works [43,66,47].

Metric	EDA(Rndm)	EDA(wPOI)	EDA+PCA(Rndm)	EDA+PCA(Dec)	EDA+PCA(wPOI)
SR (%)	12,0%	71,0%	72,0%	94,6%	98,2%
CR (%)	91,4%	99,6%	94,2%	99,8%	99,6%
$\mathbf{ge_{acc_{avg}}}$	65,57	17,61	21,53	3,01	1,76
$\varepsilon_{\mathbf{Gen}}$ **(%)**	1,33%	0,74%	0,27%	0,00%	0,39%
ge_S	1,32	1,00	1,00	1,00	1,00
ge_G	4,72	2,89	1,69	1,00	1,99

Table 5. "RAW EDA" approach vs "PCA+EDA" approach (ASCAD Variable)

7.2 Results on AES_RA

To test the performance of our approach in a noisier environment, we use AES_RA. To this end, we repeat the same analysis as with ASCAD. Note that, for this experiment, we are using traces from STM32F4 with Masking Scheme 1 (Weak). Besides, we also employ a time window of 100 samples corresponding to the first byte of the masked SBox lookup (See Fig. 8) instead of using the full window of 16 lookups. This makes this experiment similar to the previous one with ASCAD. Table 6 summarizes the parameters of the attacks.

Parameter	# Profiling Traces	# Attack Traces	# Attacks/Ind	# Iterations	Pop. Size
Setting	50 000	2 000	4	10	50

Table 6. Parameters of the attack (STM32F4)

As in the previous experiment, Fig. 7 (left) shows the results of performing a "traditional" TA using a different number of PCs. As before, there is an inflexion point (20PC) after which the results worsen if we use more PCs to generate the model. Again, the Scree Test does not provide a good number of PCs to keep (50). In this case, although the masked AES implementation is similar to the one used in

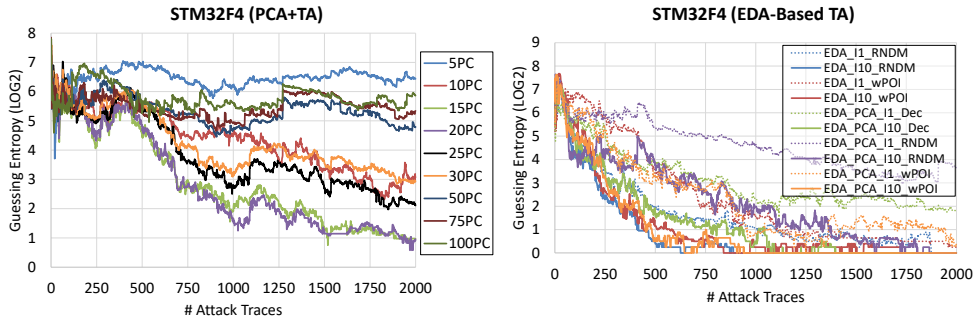


Fig. 7. Attacks on STM32F4: regular PCA + TA (left) and EDA-based TAs (right)

ASCAD, the attack is more difficult due to the amount of measurement noise included in the traces. This makes PCA less effective as apart from the variation produced by the leakage there is a lot of variation in the power traces due to environmental noise captured by the capacitor EM probe.

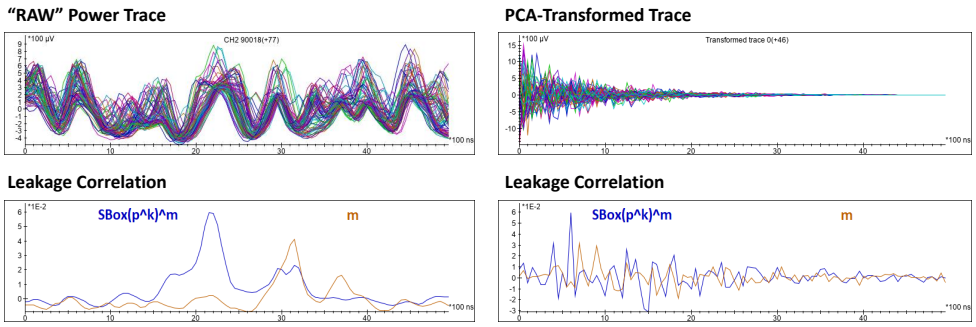


Fig. 8. STM32F4: Raw power traces (left), PCA-transformed traces (right) and leakage correlation. X-axis: Time ($\times 100ns$), Y-axis: Voltage ($\times 100\mu V$)/Correlation

.

Again, we compare the "raw" EDA-based attack and the attack on the "PCA-Transformed" traces (Fig. 7 (right)). Please note that the results are shown in the same manner as in the previous use case. This time the results are slightly different. Although we succeed with all approaches, EDA+PCA does not improve the results so much in this case. In terms of Guessing Entropy, the best performing methods are the "White-Box" approaches (EDA_wPOI and EDA_PCA_wPOI). Note that EDA_Rndm and EDA_PCA_Dec also provide relatively good results.

Table 7 shows the results of our metrics. Generally speaking, the results are worse than in the previous use case (due to noise), but they are in line with the results shown in Fig. 7 (right). In this case, not only all metrics are not better while using the PCA+EDA approach, but they are worse in general. About generalization, all methods show a small generalization error except EDA+PCA(Dec), which do not succeed in the attack on unseen data. The main reason for this, as can be observed in Fig. 8, is that the leakage is not concentrated on the first PCs. Thus, we are including PCs that do not contain leakage information and hence worsen the model. This makes

both the approach of performing "traditional" attacks and using the EDA+PCA(Dec) not the most optimal for this case.

Metric	EDA(Rndm)	EDA(wPOI)	EDA+PCA(Rndm)	EDA+PCA(Dec)	EDA+PCA(wPOI)
SR (%)	86,00%	75,40%	68,20%	27,20%	74,60%
CR (%)	99,60%	99,80%	97,20%	77,40%	97,60%
$\mathbf{ge}_{\mathbf{acc}_{\mathbf{avg}}}$	8,46	9,88	19,08	37,74	13,63
$\varepsilon_{\mathbf{Gen}}$ **(%)**	0,00%	0,00%	0,00%	8,92%	0,00%
ge_S	1	1	1	1	1
ge_G	1	1	1	23,83	1

Table 7. "RAW EDA" approach vs "PCA+EDA" approach (STM32F4)

The challenge of EM Capacitor Probe traces As shown in the previous experiments, although the masking implementation 1 (Weak) is similar to the one employed in ASCAD, obtaining good results is more complicated. The main reason for that is the acquisition method: STM32F4 power traces were obtained using a EM capacitor probe (Langer probe). This allows a less invasive acquisition (as there are not removed capacitors and the board is not modified at all) but the leakage of the traces is weaker as it is merged with the variation caused by the environmental noise.

This is especially problematic when we use a wider window. If we repeat the previous experiment with a window of 1 800 time samples corresponding to the 16 lookups, the results are extremely defective (See Fig. 9 and Table 8). With the full window, we cannot succeed with a "traditional" TA using PCA for POI selection (Fig. 9 (left)). Regarding EDA-based attacks, only attacks without PCA provide particularly good results (Fig. 9 (right)). To understand this, one should take a look at Figures 10 and 11.

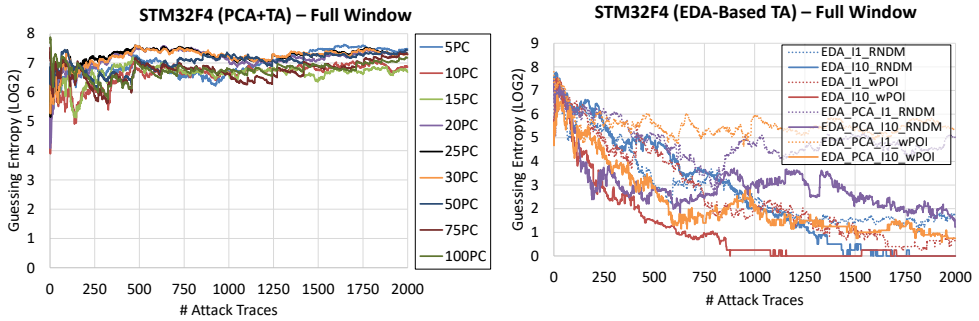


Fig. 9. Attacks on STM32F4 (Full Window): regular PCA + TA (left) and EDA-based TAs (right)

Fig. 10 shows "raw" power traces, PCA-transformed traces and their respective leakage graphics for the STM32F4 board (Full Window). Fig. 11 shows the same graphics for the same implementation in Piñata board (clean traces taken with a current probe). With Piñata, the traces are clean from enviromental noise. This allows PCA to perform successfully as all leakage is gathered on the largests PCs. In this case, the attack is extremely easy, as shown previously in Sec. 5.1.

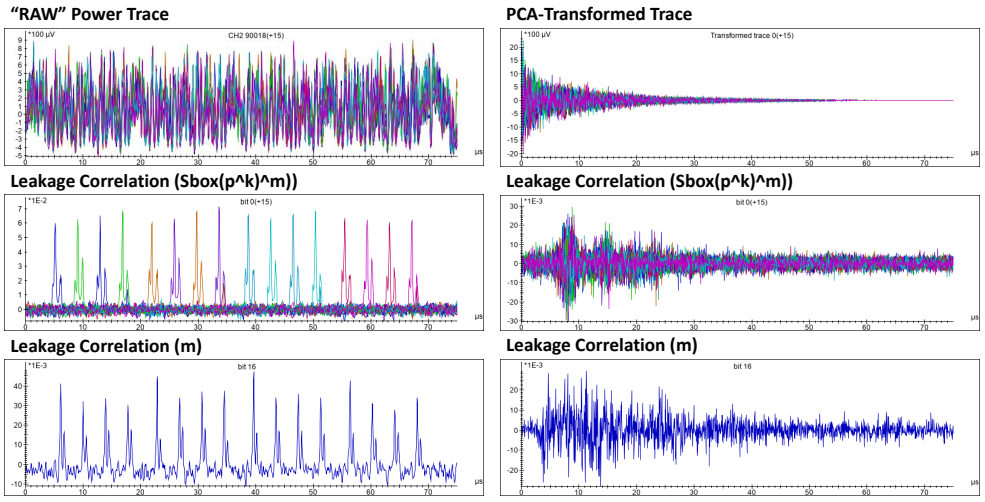


Fig. 10. STM32F4 (Full Window): Raw power traces (left), PCA-transformed traces (right) and leakage correlation. X-axis: Time (μs), Y-axis: Voltage ($\times 100 \mu V$)/Correlation.

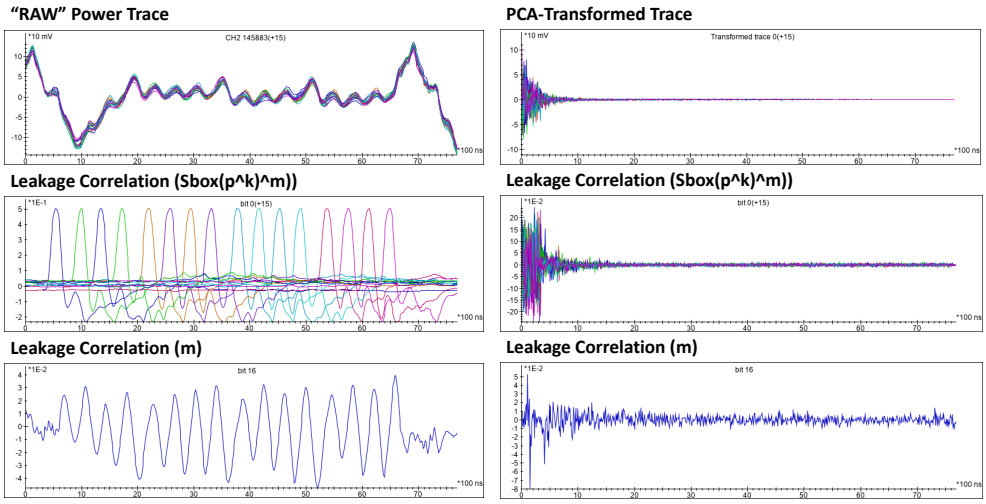


Fig. 11. Pinata (Full Window): Raw power traces (left), PCA-transformed traces (right) and leakage correlation. X-Axys: Time ($\times 100 ns$), Y-Axys: Voltage ($\times 10 mV$)/Correlation

Conversely, in our traces from SM32F4, instead of collecting all the leakage in a few PCs, PCA mixes this leakage with the variation produced by the environmental noise, causing the leakage to be attenuated and distributed over the entire PCA-transformed trace. Indeed, there is almost no leakage in the first 100 PCs (See Fig. 10). Moreover, the magnitude of the leakage has decreased substantially. This explains why the "traditional" TA+PCA does not work in this setup (Fig. 9, left). Regarding the Scree Test, it suggests using 550 PCs, which is completely impractical. On the one hand, this number is too large to build templates, especially taking into account that the purpose of using PCA in this case is to reduce the dimensionality of the traces. On the other hand, as shown in the previous use case, building templates with PCs that do not contain leakage information but rather noise variation, worsens the model.

For all these reasons, as can be seen in Table 8, PCA not only performs worse in this case but does not work at all if we do not select the appropriate PCs, which is extremely tedious to do manually. Among the attacks using PCA, only EDA_PCA_Rndm and EDA_PCA_wPOI manage to find a model which works in the set of traces used for the search, but they have very bad generalization. On the other hand, the attacks EDA_Rndm and EDA_wPOI perform quite good and have a small and tolerable generalization error.

Metric	**EDA(Rndm)**	**EDA(wPOI)**	**EDA+PCA(Rndm)**	**EDA+PCA(Dec)**	**EDA+PCA(wPOI)**
SR (%)	58,60%	66,00%	2,40%	Null	3,40%
CR (%)	98,00%	96,80%	65,20%	Null	56,80%
$\mathbf{ge_{acc_{avg}}}$	38,32	25,20	77,62	Null	71,90
$\varepsilon_{\mathbf{Gen}}$ **(%)**	0,13%	0,13%	18,46%	Null	35,87%
ge_S	1	1	2,75	Null	2
ge_G	1,333	1,333	50	Null	93,83

Table 8. "RAW EDA" approach vs "PCA+EDA" approach (STM32F4 - Full Window)

7.3 Summary

We can draw the following conclusions from the experiments above:

- Using PCA-Transformed traces accelerates the EDA-based PA process when the traces are clean (clean EM measurements/current probe and no capacitors). This is the best option when seeking to optimize a model, provided that the nature of the traces allows it: they must be free of, or with little, ambient noise.
- Another limitation could be the number of traces and time points per trace. Since the computation time grows exponentially with these two factors, it may be prevented for very large datasets or very wide attack windows.
- However, our approach has shown an excellent performance as a PC selection method, being able to automatically identify the best components even in the more challenging use cases. This makes it an engaging option when working with PCA in SCA, or in some other field.
- Nevertheless, although they require knowing the mask m, "White-Box" approaches work properly in both situations (with and without PCA), being the best approach from an evaluation perspective. In our experiments, these approaches have substantially improved the performance of "traditional" attacks, with the additional benefit of being done automatically and with no user intervention.

8 Conclusions and Future Work

Our results show the suitability of automated TAs working as a robustness assessment test of an embedded device's physical security. It allows the evaluator to determine whether protected AES implementations are secure (while comparing them to an unprotected AES implementation) without user intervention.

We have shown that masking schemes like Masking Scheme 1 or the one used in ASCAD are especially weak. As a consequence, the SBox's output mask should be removed from the state matrix out of the time window of the SBox to make them more robust against PAs. Nevertheless, we were able to find models that work in some sets of traces with Masking Scheme 2, and we claim that AES_RA can serve as a relevant candidate to further study PAs on masking-protected AES implementations.

We claim that this approach is not limited only to software AES implementations, although we have chosen this realistic use case for demonstrating our approach. Hardware AES implementations, different ciphers, or even other kinds of profiling attacks are potential future work candidates.

We have also shown that using PCA-transformed traces can hasten the EDA-based PA in some scenarios, achieving superior results. Furthermore, we have demonstrated how this approach can be used as an automated way of selecting optimal principal components, obtaining state-of-the-art results without manual intervention even some more troublesome cases (very noisy traces).

Finally, as an open research question, we would like to mention the common gap between security evaluation in academia and in commercial labs. As mentioned above, concepts like generalization were "traditionally" not considered when working with TAs. Indeed, current evaluation schemes specify which types of tests (attacks) to undertake, but without much detail on how to conduct them. This makes it relatively easy for false negatives/positives to occur. As we have seen in the experimental results with the Masking Scheme 2, finding a model that works on a finite set of traces is relatively simple. Conversely, getting one that is generalizable and can perform an actual attack is much more complex. Unless we take this into account, we might think that the implementation is weak as we have succeeded in the attack, however, this model is not applicable in the real world by an attacker. Therefore, we believe that a more comprehensive common framework on SCA resistance assurance could be interesting. This, together with the application of artificial intelligence to mitigate the human dependency, could make life much easier for cybersecurity evaluators and product developers.

References

1. AES-RA: The AES_RA Dataset. `https://github.com/AES-RA/AES_RA` (2021)
2. Archambeau, C., Peeters, E., Standaert, F., Quisquater, J.: Template attacks in principal subspaces. In: Cryptographic Hardware and Embedded Systems - CHES 2006. pp. 1–14 (2006)
3. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.X., Veyrat-Charvillon, N.: Mutual Information Analysis: a Comprehensive Study. J. Cryptology **24**(2), 269–291 (2011)
4. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. Journal of Cryptographic Engineering **10** (06 2020). https://doi.org/10.1007/s13389-019-00220-8
5. Bohy, L., Neve, M., Samyde, D., Quisquater, J.J.: Principal and independent component analysis for crypto-systems with hardware unmasked units. Proceedings of e-Smart 2003 (01 2003)
6. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 16–29 (2004)
7. Cagli, E., Dumas, C., Prouff, E.: Enhancing dimensionality reduction methods for side-channel attacks. In: Smart Card Research and Advanced Applications. pp. 15–33 (2016)
8. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Cryptographic Hardware and Embedded Systems - CHES 2002. pp. 13–28 (2002). https://doi.org/10.1007/3-540-36400-5_3
9. Choudary, M.O., Kuhn, M.G.: Efficient, portable template attacks. IEEE Transactions on Information Forensics and Security **13**(2), 490–501 (Feb 2018). https://doi.org/10.1109/TIFS.2017.2757440
10. Cristiani, V., Lecomte, M., Maurine, P.: Leakage assessment through neural estimation of the mutual information. In: Applied Cryptography and Network Security Workshops. pp. 144–162 (2020)

11. Common Criteria: Common Criteria v3.1. Release 5. https://www.commoncriteriaportal.org/cc/index.cfm? (apr 2017)
12. EMVCo: EMV specifications. https://www.emvco.com/ (2001)
13. Federal Information Processing Standard: FIPS 197: Announcing the Advanced Encryption Standard (AES) (November 2001). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
14. Feng, H., Lin, W., Shang, W., Cao, J., Huang, W.: MLP and CNN-based Classification of Points of Interest in Side-Channel Attacks. International Journal of Networked and Distributed Computing **8**, 108–117 (2020)
15. Fisher, R.: The statistical utilization of multiple measurements. Annals of Eugenics (Cambridge) **8**, 376–386 (November 1935). https://doi.org/10.1111/j.1469-1809.1938.tb02189.x
16. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Cryptographic Hardware and Embedded Systems – CHES 2008. pp. 426–442 (2008)
17. Golder, A., Das, D., Danial, J., Ghosh, S., Sen, S., Raychowdhury, A.: Practical approaches toward deep-learning-based cross-device power side-channel attack. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **PP**, 1–14 (07 2019). https://doi.org/10.1109/TVLSI.2019.2926324
18. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side-channel resistance validation. NIST Non-Invasive Attack Testing Workshop (2011)
19. Heuser, A., Zohner, M.: Intelligent machine homicide. In: Constructive Side-Channel Analysis and Secure Design. pp. 249–264. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29912-4_18
20. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. J. Cryptographic Engineering **1**, 293–302 (Oct 2011). https://doi.org/10.1007/s13389-011-0023-x
21. International Organization for Standardization: ISO/IEC 17825:2016 information technology — security techniques — testing methods for the mitigation of non-invasive attack classes against cryptographic modules. https://www.iso.org/standard/60612.html (2016)
22. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)
23. Johnson, R., Wichern, D.: Applied multivariate statistical analysis. Prentice Hall, Upper Saddle River, NJ, 5. ed edn. (2002)
24. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics, Springer-Verlag (2002). https://doi.org/10.1007/b98835
25. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(3), 148–179 (May 2019)
26. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 388–397. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
27. kokke: Tiny AES in C. https://github.com/kokke/tiny-AES-c (2014)
28. Larranaga, P., Lozano, J.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Genetic algorithms and evolutionary computation, vol. 2. Springer US, 1st edn. (Jan 2002). https://doi.org/10.1007/978-1-4615-1539-5
29. Le Corre, Y., Großschädl, J., Dinu, D.: Micro-architectural power simulator for leakage assessment of cryptographic software on arm cortex-m3 processors. In: Constructive Side-Channel Analysis and Secure Design. pp. 82–98 (2018)
30. Lerman, L., Bontempi, G., Markowitch, O.: Side channel attack : an approach based on machine learning. In: Constructive Side-Channel Analysis and Secure Design (2011)
31. Lerman, L., Bontempi, G., Markowitch, O.: A machine learning approach against a masked aes. Journal of Cryptographic Engineering **5**(2), 123–139 (Jun 2015). https://doi.org/10.1007/s13389-014-0089-3
32. Lerman, L., Poussier, R., Markowitch, O., Standaert, F.X.: Template attacks versus machine learning revisited and the curse of dimensionality in side-channel

analysis: extended version. Journal of Cryptographic Engineering **8** (11 2018). https://doi.org/10.1007/s13389-017-0162-9
33. LOMNE, V.: Common criteria certification of a smartcard: a technical overview. CHES 2016 tutorial #1 (2016)
34. Lozano, J., Larranaga, P., Inza, I., Bengoetxea, E.: Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, Studies in Fuzziness and Soft Computing, vol. 192. Springer-Verlag Berlin Heidelberg, 1st edn. (Jan 2006). https://doi.org/10.1007/3-540-32494-1
35. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking Cryptographic Implementations Using Deep Learning Techniques. In: SPACE 2016. pp. 3–26. Springer (December 2016)
36. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer (2007)
37. Masure, L., Dumas, C., Prouff, E.: A Comprehensive Study of Deep Learning for Side-Channel Analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020** (11 2019). https://doi.org/10.13154/tches.v2020.i1.348-375
38. Mavroeidis, D., Batina, L., van Laarhoven, T., Marchiori, E.: PCA, eigenvector localization and clustering for side-channel attacks on cryptographic hardware devices. In: Machine Learning and Knowledge Discovery in Databases. pp. 253–268 (2012)
39. McCann, D., Oswald, E., Whitnall, C.: Towards practical tools for side channel aware software engineering: 'Grey Box' modelling for instruction leakages. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 199–216. USENIX Association, Vancouver, BC (Aug 2017)
40. Moradi, A., Richter, B., Schneider, T., Standaert, F.X.: Leakage Detection with the $\mathcal{X}^2$-Test. IACR Transactions on Cryptographic Hardware and Embedded Systems (2018). https://doi.org/10.13154/tches.v2018.i1.209-237
41. Mukhtar, N., Fournaris, A.P., Khan, T.M., Dimopoulos, C., Kong, Y.: Improved hybrid approach for side-channel analysis using efficient convolutional neural network and dimensionality reduction. IEEE Access **8**, 184298–184311 (2020). https://doi.org/10.1109/ACCESS.2020.3029206
42. Mukhtar, N., Mehrabi, A., Kong, Y., Anjum, A.: Machine-learning-based side-channel evaluation of elliptic-curve cryptographic FPGA processor. Applied Sciences **9**, 64 (12 2018). https://doi.org/10.3390/app9010064
43. 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)
44. 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 **PP**, 1–14 (09 2019). https://doi.org/10.1109/TVLSI.2019.2937365
45. Picek, S., Samiotis, I.P., Kim, J., Heuser, A., Bhasin, S., Legay, A.: On the performance of convolutional neural networks for side-channel analysis. In: Security, Privacy, and Applied Cryptography Engineering. pp. 157–176 (2018)
46. Rechberger, C., Oswald, E.: Practical template attacks. In: Information Security Applications. pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31815-6_35
47. 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)
48. Rioja, U., Batina, L., Flores, J.L., Armendariz, I.: Auto-tune pois: Estimation of distribution algorithms for efficient side-channel analysis. Computer Networks **198**, 108405 (2021). https://doi.org/https://doi.org/10.1016/j.comnet.2021.108405
49. Rioja, U., Batina, L., Flores, J.L., Armendariz, I.: Towards automatic and portable data loading template attacks on microcontrollers. In: 2021 22nd International Symposium on Quality Electronic Design (ISQED). pp. 437–443 (2021). https://doi.org/10.1109/ISQED51717.2021.9424276
50. Riscure: Piñata board brochure. `https://www.riscure.com/uploads/2017/07/pi%C3%B1ata_board_brochure.pdf`

51. Saeedi, E., Kong, Y.: Side channel information analysis based on machine learning. In: 2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS). pp. 1–7 (2014). https://doi.org/10.1109/ICSPCS.2014.7021075
52. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Cryptographic Hardware and Embedded Systems - CHES 2005. pp. 30–46 (2005). https://doi.org/10.1007/11545262_3
53. Shelton, M., Samwel, N., Batina, L., Regazzoni, F., Wagner, M., Yarom, Y.: Rosita: Towards automatic elimination of power-analysis leakage in ciphers. In: Network and Distributed System Security Symposium (01 2021). https://doi.org/10.14722/ndss.2021.23137
54. Smith, L.: A tutorial on principal components analysis. `http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf` (2002)
55. Souissi, Y., Nassar, M., Guilley, S., Danger, J.L., Flament, F.: First principal components analysis: A new side channel distinguisher. In: Information Security and Cryptology - ICISC 2010. pp. 407–419. Springer Berlin Heidelberg (2011)
56. Specht, R., Heyszl, J., Kleinsteuber, M., Sigl, G.: Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution em measurements. In: Constructive Side-Channel Analysis and Secure Design. p. 3–19 (2015)
57. Standaert, F.: How (not) to use welch's t-test in side-channel security evaluations. CARDIS 2018 **2018** (2018)
58. Standaert, F.X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Cryptographic Hardware and Embedded Systems - CHES 2008. pp. 411–425 (2008). https://doi.org/10.1007/978-3-540-85053-3_26
59. 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)
60. STMicroelectronics: Discovery kit with STM32F411VE MCU. `https://www.st.com/en/microcontrollers-microprocessors/stm32f411ve.html`
61. STMicroelectronics: STM32F411VET6 datasheet. `https://www.alldatasheet.com/datasheet-pdf/pdf/929991/STMICROELECTRONICS/STM32F411VET6.html` (Dec 2016)
62. Wegener, F., Moos, T., Moradi, A.: DL-LA: Deep Learning Leakage Assessment: A modern roadmap for SCA evaluations. IACR Cryptology ePrint Archive **2019**, 505 (2019), `https://eprint.iacr.org/2019/505`
63. Weissbart, L., Chmielewskiz, L., Picek, S., Batina, L.: Systematic side-channel analysis of curve25519 with machine learning. Journal of Hardware and Systems Security **4**, 314–328 (12 2020). https://doi.org/10.1007/s41635-020-00106-w
64. Whitnall, C., Oswald, E.: A critical analysis of iso 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In: Advances in Cryptology – ASIACRYPT 2019, 25th International Conference on the Theory and Application of Cryptology and Information Security. pp. 256–284. Springer (11 2019). https://doi.org/10.1007/978-3-030-34618-8_9
65. Won, Y.S., Hou, X., Jap, D., Breier, J., Bhasin, S.: Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks. IEEE Transactions on Information Forensics and Security **16**, 3215–3227 (2021). https://doi.org/10.1109/TIFS.2021.3076928
66. 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)
67. Wu, L., Weissbart, L., Krcek, M., Li, H., Perin, G., Batina, L., Picek, S.: On the attack evaluation and the generalization ability in profiling side-channel analysis. In: Cryptol. ePrint Arch., Tech. Rep 899 (2020)
68. 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** (07 2019). https://doi.org/10.13154/tches.v2020.i1.1-36
69. Zotkin, Y., Olivier, F., Bourbao, E.: Deep learning vs template attacks in front of fundamental targets: experimental study. IACR Cryptol. ePrint Arch. **2018**, 1213 (2018)