Towards Human Dependency Elimination: AI Approach to SCA Robustness Assessment

Unai Rioja, Lejla Batina, *Senior Member, IEEE*, Igor Armendariz, and Jose Luis Flores

***Abstract*— Evaluating the side-channel resistance 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 (Estimation of Distribution Algorihm(EDA)-based) 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, in this context of automation, we propose several novel improvements over current EDA-based attacks, as follows: 1) optimization of the search process by employing two proposed initialization techniques; 2) improvement and analysis of the generalization of the obtained templates; 3) acceleration of the search process by combining EDAs with Principal Component Analysis (PCA). The last contribution also serves as an alternative way of selecting optimal principal components automatically. 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.**

***Index Terms*— SCA, profiling attacks, template attacks, EDAs, evaluation.**

I. 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 [1] or Common Criteria (CC [2]) assess the security of the device under test (DUT) by applying a battery of known SCA (e.g., differential power analysis (DPA) [3], correlation power analysis (CPA) [4], mutual information analysis (MIA) [5], [6], template attacks (TA) [7]–[9], and machine learning-based attacks (ML-SCA) [10]–[14]).

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) [15]. 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 a low-cost and efficient evaluation. Furthermore, the success or failure of these attacks strongly depends 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 a device's security quite conditional on the person implementing the tests. And given that humans are error-prone and knowledge is sometimes challenging to transfer from one person to another, technicians and product developers face 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 [16]) 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 [17] or *Pearson's* $\mathcal{X}^2$-test [18], or even Deep Learning [19] or Mutual Information [20], 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 [22], 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 [23]–[25]. In a way, those works also share the same objective as ours, since they aim to reduce the evaluation's cost, 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 drawback is that current leakage simulators for SCA such as ELMO [23], or its improved version ELMO* [25]

Manuscript received 29 September 2021; revised 28 January 2022 and 23 March 2022; accepted 6 May 2022. Date of publication 18 May 2022; date of current version 23 November 2022. This work was supported in part by the Spanish Government under the Ayudas Cervera para Centros Tecnológicos Grant of the Spanish Centre for the Development of Industrial Technology (CDTI) under Project EGIDA (CER-20191012), and in part by the Basque Country Government under the ELKARTEK collaborative research grants program, project REMEDY - REal tiME control and embeddeD securitY (KK-2021/00091). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Debdeep Mukhopadhyay. *(Corresponding author: Unai Rioja.)*

Unai Rioja is with the Digital Security Group, Institute for Computing and Information Sciences (ICIS), Radboud University, 6525 EC Nijmegen, The Netherlands, and also with the Ikerlan Technological Research Centre, Basque Research and Technology Alliance (BRTA), 20500 Arrasate-Mondragón, Spain (e-mail: urioja@ikerlan.es).

Lejla Batina is with the Digital Security Group, Institute for Computing and Information Sciences (ICIS), Radboud University, 6525 EC Nijmegen, The Netherlands (e-mail: lejla@cs.ru.nl).

Igor Armendariz and Jose Luis Flores are with the Ikerlan Technological Research Centre, Basque Research and Technology Alliance (BRTA), 20500 Arrasate-Mondragón, Spain (e-mail: iarmendariz@ikerlan.es; jlflores@ikerlan.es).

Digital Object Identifier 10.1109/TIFS.2022.3176189

are not generic enough. These emulators use a very simple (instruction-level) model, making a detailed hardware description or information about the used process technology not mandatory. However, they are only suitable for small microcontrollers like Cortex-M0, small RISC-V processors, or AVR processors like ATMEGA328p. Thus, the approach may not be suitable for more advanced processors [25]. This motivates the creation of a non-hardware-specific alternative to determine the actual security of a device in a simple way, as the one proposed in this paper.

In short, considering the ideas mentioned above, 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 [26]. In any case, these attacks can be quite complex, and the intrinsic human nature of several parts of the operation (acquisition, pre-processing, 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 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 is more recent and is 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.

In summary, we list he most important contributions of this paper as follows:

1) We propose to use Estimation of Distribution Algorithms (EDA)-based PA automated attacks (as introduced in [27]) in an alternative and innovative way. Namely, orthogonal to [27] 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 [28] 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 a masking protected implementation is secure against profiling attacks. In other words, the approach can easily detect whether close manipulation of the mask and masked intermediate value exist, reducing the security order and making it weak against profiling attacks. We demonstrate the suitability of our method with attacks against two distinct devices (Piñata board [29] and STM32F411-Discovery board [30]). Thus, we perform automated attacks against the SBox of different AES [31] implementations on the same device to assess its physical security. We also make our traces public, creating the AES_RA dataset [32] as a part of our contribution.

2) We propose several improvements over current EDA-based PAs such as:

- Optimization of the search process, in terms of timing and guessing entropy, by employing two proposed novel initialization techniques for the EDA's probabilistic model.
- Improvement and analysis of the generalization of the obtained templates through cross-validation during the search process.
- Acceleration of the search process by combining EDA-based TAs with Principal Component Analysis (PCA) as an alternative way of performing EDA-based PA by employing it on PCA-transformed power traces, rather than on the "raw" traces.

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 enough, 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]" [33], 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 [34]), 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. Our experiments are limited to cryptographic implementations in software, and therefore the approach is currently restricted to that scenario.

The remainder of this paper is organized as follows. Sect. II summarizes the important background and related works on this topic. In Sect. III 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. IV. We specify the procedure of EDA-based Robustness Assessment in Sect. V, providing experimental results supporting our approach (Contribution 1). In Sect. VI we elaborate our improved EDA-based PA (Contributions 2 and 3). Sect. VII contains the experimental results supporting the modifications proposed in the previous section. Finally, Section VIII concludes the paper.

II. Background and Related Work

In this section, we first present previous related works which are relevant for this paper. Afterward, we briefly explain the background necessary to understand our work.

A. Related Work

1) 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, apart from our previous works [27], [35]. There, we proposed using 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, these works were limited to introducing the method at a very early stage. Thus, some crucial concepts such as the generalization of the obtained templates [36] or the close manipulation of the mask and masked intermediate value in protected implementations [25], [37], were not taken into account. In this paper, we consider these factors and 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 initialization method for the EDA approach by systematically comparing the performance of different, including the two new, proposals.

2) 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 [38]. Later on, PCA was used as a POI selection technique for template attacks in [39]. Afterwards, it has been used for POI selection in profiling attacks in a large number of works (e.g., [34], [40]–[46]).

Other works try to enhance PCA (among other dimensionality reduction methods) for SCA [47] or compare PCA against other POI selection techniques [48]. PCA has also been used as a pre-processing technique to improve the correlation for the correct key candidate [49]. In [50] the authors followed a different approach and used PCA not as a pre-processing 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. Choosing a proper number of PCs to keep is crucial for obtaining favorable results. There exist several "traditional" ways to obtain the number of PCs needed, as shown in [51]. 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 [49], [52], when the 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, 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, no related works 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 [47], [49], but they have the same drawback as they rely on the variance of the traces and not on its leakage. In [49], the authors propose to compute the Inverse Participation Ratio (IPR) score and collect the PCs in decreasing order accordingly. In [47] 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.

B. Notation

In this section we briefly define the notation used throughout the paper. We adopt the notation introduced in [53], 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 $\upsilon = f(p, k)$ 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.

C. Template Attacks

Template Attacks (TAs) were proposed in [7] 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). Different types of profiling attacks exist depending on how the model is generated. Whereas template attacks use estimation theory to model the probability distribution of the leakage [7], [8], other procedures use linear regression (stochastic models approach [54]) or machine learning [10], [11], including the lately introduced tendency of using deep learning techniques [34], [48], [55] 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 large number of power traces of the DUT while it manipulates some intermediate value $\upsilon = f(p, k)$. 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 υ, 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 [9], [53].

Finally, the attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|K|}]$, in decreasing order of probability. We are assessing the performance of the attack by using an SCA-specific metric (Guessing Entropy, GE [28]). The guessing entropy is the average position of the correct key k^* in the key guessing vector over multiple experiments. The higher the 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. However, 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 [8]), or using a more complex method like Principal Component Analysis (PCA) [39], [40] or Fisher's Linear Discriminant Analysis (LDA) [56], [57]). Note that, with EDA-based PA, the POI selection is made automatically by the algorithm [27].

D. 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 [58]:

- **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 $\Sigma = \mathbf{U} * \Lambda * \mathbf{U^{-1}}$, where Λ is the diagonal eigenvalue matrix and $\mathbf{U}$ is the eigenvector matrix of Σ. 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 $\mathbf{U}^{p\prime}$ and multiply it with the transposed mean-adjusted data $\mathbf{X}'$, obtaining the transformed dataset $\hat{\mathbf{X}}$:

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

E. Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) are stochastic optimization techniques that search for potential solutions by building explicit probabilistic models of promising candidates. Unlike other evolutionary algorithms, the main advantage of EDAs is their simplicity. On the one hand, EDAs involve a much smaller number of tunable parameters than other evolutionary algorithms (e.g., genetic algorithms, GAs), as the new population is generated from a probability distribution obtained from the best individuals of previous populations [27], [59], [60]. On the other hand, with heuristics such as GA, we not only have to take into account the usual parameters in evolutionary algorithms (probabilities, population percentage, etc.), but we also need an optimal operator design. Namely, as highlighted in a recent study [61], designing and validating mutation and crossover operators is not only critical but an optimization problem in itself. This made us discard other evolutionary techniques, as their inclusion increases the complexity of the attack rather than simplifying the process.

1) Estimation of Distribution Algorithms in SCA: EDAs were proposed in [27] 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 optimization of the attack, avoiding the need to perform various types of analyses with different POI combinations manually. As an exhaustive enumeration of all combinations is exponential and definitively not feasible, our approach uses 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.

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

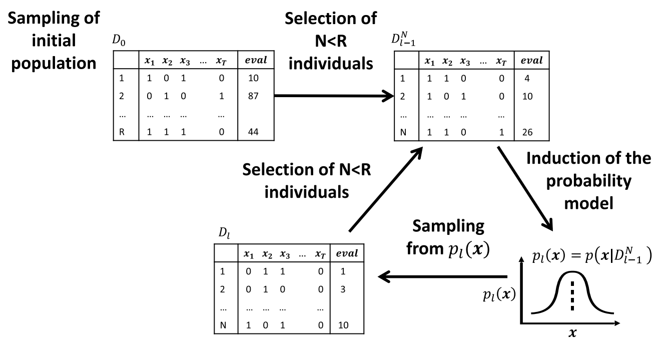
Sampling of initial population
Selection of N<R individuals
Induction of the probability model
Sampling from $p_l(x)$
Selection of N<R individuals
D_0
D_{l-1}^N
D_l
$p_l(x) = p(x|D_{l-1}^N)$
$p_l(x)$
x
eval

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

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 [27], 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 [27]. 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. A "Toy examnple of a generic EDA-based PA can be found on [62]. For a deeper explanation, we refer to [27].

2) Complexity of the Approach: The time required to obtain satisfactory results will depend on the difficulty of the attack, i.e., the number of iterations needed for obtaining GE = 0. It will also vary significantly from one computer (or programming language implementation) to another. However, in terms of time complexity, the cost of evaluating a set of discrete variables with univariate EDAs is linear $O(n)$ [63], which is much less than manual approaches [27] or using DL [64]. In our setup, our tool takes between 10 minutes and an hour to perform a complete iteration, depending on the number of time points and power traces used to build the templates. Thus, the 10 iterations considered in the experiments take between 2 and 10 hours approximately. However, it should be noticed that the time-consuming part of our EDA-Based TA are the template attacks themselves, which represent about 99% of the computation. While challenging targets will require several iterations to succeed, in many cases, success is achieved in the first one, as shown in the experiments. Finally, note that the method is in its early stage, and these results could be still improved, as there is a lot of room for optimization (e.g., attack parallelization, optimization of attack computation, etc. [27]).

III. Robustness Assessment Test

This section describes our approach for the robustness assessment test and the proposed metrics. Fig. 2 shows a schematic of the process. In a nutshell, the idea is to perform a battery of automated attacks, using our improved EDA-based TA (see Sect. VI), 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.

A. Metrics

To assess the performance of our improved EDA-based TA (Sect. VI), and hence execute 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 [28]: Guessing Entropy and Success Rate.

1) 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 [28]. 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:

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

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

2) 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

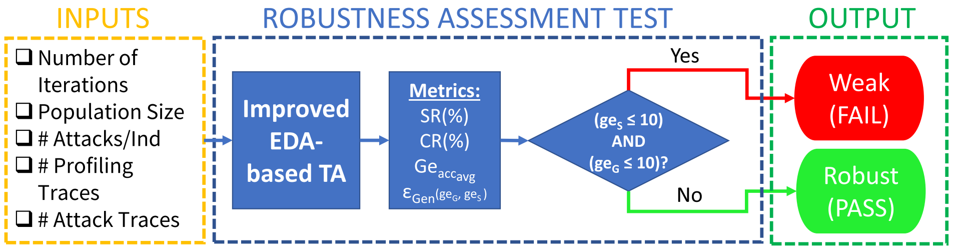

Fig. 2. Robustness assessment test scheme.

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:

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

where n_{Trials} is the number of trials before a successful attack and $n_{Attacks}$ is the total number of attacks.

3) 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 [28] 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:

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

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.

4) 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 [36].

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.

5) Diff Score[DS](%): Additionally, the evaluator can also compute a "Diff score" to quantify how weak the considered implementation is compared to an unprotected implementation. We mainly use this metric for explanatory reasons, but it can be helpful for comparing the results of the attacks over different implementations. To do so, one has to repeat the approach on different implementations (including an unprotected one) and compute the score(s). The larger the value is, the more difficult it gets to recover the secret key for an attacker. We compute it using the following formula, Equation (5), as shown at the bottom of the next page.

Here the sub-index U or M indicate whether the metric corresponds to the attack on the unprotected or masked implementation, respectively.

IV. The AES_RA Dataset

In this section, we briefly describe our new AES_RA dataset [32]. 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. VII), 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.

A. 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.

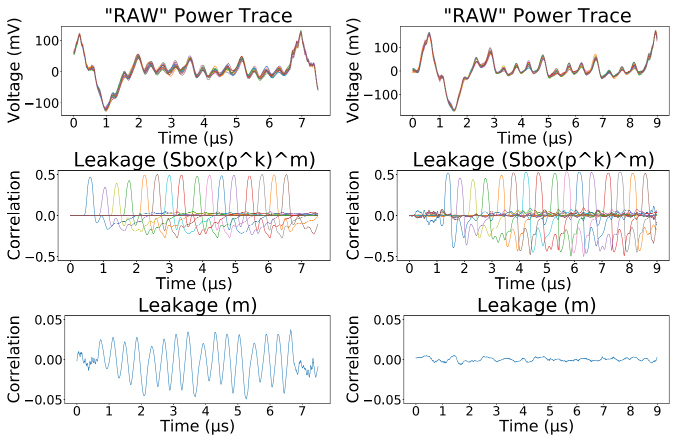

Fig. 3. Piñata board: Leakage of MS1 (left) vs MS2 (right).

- **Unprotected AES:** typical AES-128 (in ECB mode) software implementation [65].
- **Masking Scheme 1 (Weak):** A modification of the previous one which matches the same masking method as described in [53] (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 [34]. As we show in the experiments below, the close manipulation of the shares (i.e., mask and masked intermediate value) make this implementation weak against PAs.
- **Masking Scheme 2 (Robust):** A modification of the previous one in which the output mask is removed after the *ShiftRows* operation. Thus, the output mask does not leak during the SBox computation, unlike in the previous scheme. Thus, there is no close manipulation of the shares, making the implementation secure against PAs.

For the pseudocode of both masking schemes and more information about the dataset organization, please see the AES_RA GitHub [32].

B. Pi nata Board

Piñata is a development board created by Riscure based on an ARM Cortex-M4F core working at a 168 MHz clock speed [29]. 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.

C. STM32F411E-DISCO Board

The STM32F411E-DISCO is a development board with an STM32F411VE [66] 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), which 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).

V. 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 or MS1 (Weak), and AES with masking scheme 2 or MS2 (Robust). As mentioned in Sect. IV, 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).

A. Experimental Results on Riscure Pi nata 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 I. Each row represents either a metric (SR, CR, $ge_{acc_{avg}}$ and ε_{Gen}) or the parameters needed to calculate it ($n_{Success}$, $n_{Attacks}$, n_{Trials}, ge_{acc}, ge_S and ge_G), which are marked in gray. For the EDA parameters, we are using 10 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. VI-B). 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.

$$\mathrm{DS}(\%) = \frac{|SR_U - SR_M| + |CR_U - CR_M| + \left|\left(ge_{acc_U} - ge_{acc_M}\right)/256\right| + \left|\varepsilon_{Gen_U} - \varepsilon_{Gen_M}\right|}{4} \tag{5}$$

TABLE I
ROBUSTNESS ASSESSMENT: PI NATA

Metric	AES	AES MS1	AES MS2
SR (%)	**100%**	**100%**	**29.63%**
$n_{Success}$	500	500	12
$n_{Attacks}$	500	500	500
CR (%)	**100%**	**100%**	**52.5%**
n_{Trials}	1	1	380
$\mathbf{ge_{acc_{avg}}}$	**1**	**1.18**	**52.28**
ge_{acc}	500	523	39311
$\varepsilon_{\mathbf{Gen}}$ **(%)**	**0%**	**0%**	**-32.42%**
ge_S	1	1	1
ge_G	1	1	84
Test Results		**Weak (Fail)**	**Robust (Pass)**
Diff Score		**0.00%**	**59.07%**

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 [26], 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 [67], 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). Other works show how when there is close manipulation of the mask and masked intermediate value, the security order is reduced, making the scheme vulnerable even to first-order attacks [25], [37]. In fact, it is unclear whether the attack works because of unintended interactions or because, due to the presence of mask leakage in the observed time window, templates can relate the dependence between the mask and the masked variable leakage (or both). However, our approach shows the weakness of the implemented masked scheme straightforwardly. 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 [53].

B. Experimental Results on STM32F4 Discovery Board

As we show later in Sect. VII, 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.

Table II 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.

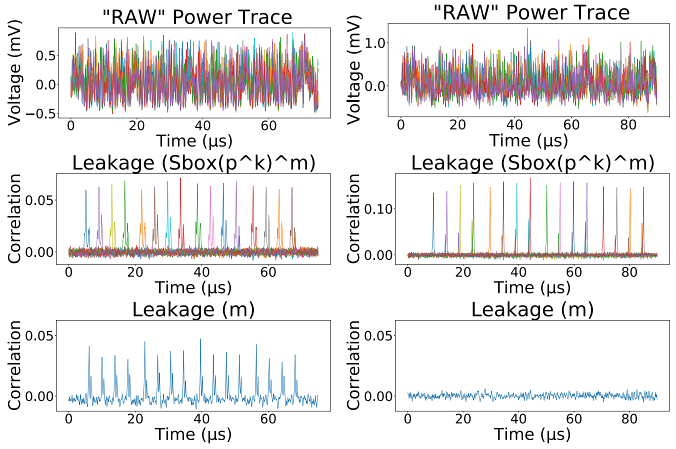

Fig. 4. STM32F411 board: Leakage of Masking Scheme 1 (left) vs Masking Scheme 2 (right).

TABLE II
ROBUSTNESS ASSESSMENT: STM32F4

Metric	AES	AES MS1	AES MS2
SR (%)	**81.80%**	**76.60%**	**9.00%**
$n_{Success}$	409	373	45
$n_{Attacks}$	500	500	500
CR (%)	**99.60%**	**97.80%**	**29.26%**
n_{Trials}	3	12	354
$\mathbf{ge_{acc_{avg}}}$	**7.86**	**13.63**	**58.23**
ge_{acc}	3923.75	6815	29116
$\varepsilon_{\mathbf{Gen}}$ **(%)**	**0.00%**	**-0.13%**	**-52.89%**
ge_S	1	1	1
ge_G	1	1.33	136.4
Test Results		**Weak (Fail)**	**Robust (Pass)**
Diff Score		**2.85%**	**53.93%**

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).

VI. IMPROVEMENTS OVER EDA-BASED TA

In this section, we go into more detail about the improvements we propose over the current EDA-based TAs [27]. In other words, this section describes our second and third contributions as follows: optimization of the search process by employing two proposed EDA's probability distribution initialization methods, improvement and analysis of the generalization of the obtained templates, and an acceleration of the search process by combining EDAs with Principal Component Analysis (PCA).

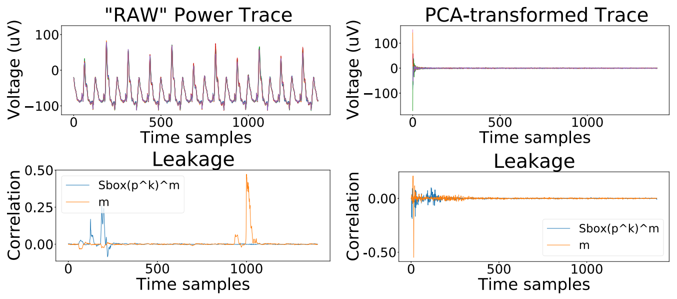

Fig. 5. ASCAD: Raw power traces (left), PCA-transformed traces (right) and leakage correlation.

A. 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 [34], as explained below) each power trace has 1 400 time samples. Thus, if we follow the approach of [27] 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. VII-A2). 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. VII.

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. II 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 [34], [55], [67], we perform the POI selection by using PCA and selecting different numbers of PCs to accomplish the attack.

TABLE III
INITIALIZATION METHODS FOR EDA'S PROBABILITY DISTRIBUTION

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

B. EDA's Probability Distribution Initialization

As mentioned in Sect. II-E, 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 [27] and two novel approaches. Table III summarizes the details of each one of them.

In a nutshell, given the limitations of the initialisation method proposed in [27] (Random Uniform in Table III) when attacking masking implementations, we propose two alternatives for this case. As explained before, this approach is not optimal for 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 [27], 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 $SBox(p \oplus k) \oplus m$. In contrast, we consider the "Rndm" and "Dec" initialization methods "Black-box" methods, as no information about the leakage (and the masks) is used.

C. Generalization of the Templates

As mentioned in Sect. II-A, in [27] the generalization of the obtained templates was not taken into account. Thus, in this paper, we not only evaluate the generalization of the obtained templates but also propose a way to improve it. To do so, we suggest performing cross-validation during the search process. Namely, instead of performing one attack

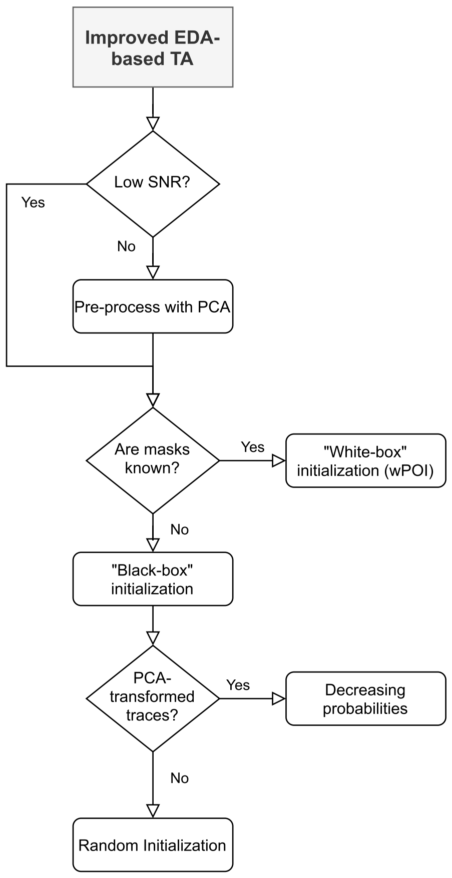

Fig. 6. Improved EDA-based TA flowchart.

per individual, as suggested in [27], performing a battery of N attacks during the searching process. This analysis, combined with the assessment of the generalization by computing the Generalization Error (as mentioned in Sect. III), allows for a better generalization of the obtained templates.

D. Improved EDA-Based TA Workflow

This section describes the workflow that an evaluator should follow while using our improved attack. Fig. 6 includes a flowchart of the strategy. First, we check if the signal is clean enough to apply PCA. For this, we propose calculating the Signal-to-noise-ratio (SNR) of the signal (explained below). Then, one should choose the appropriate initialization method based on whether the mask values are available or not. Note that this procedure indicates which method is most suitable, but this only accelerates the search process. As shown in Sect. VII, all variations manage to obtain successful results with the appropriate number of iterations.

More precisely, we propose to compute the sample Signal-to-noise-ratio (SNR) as the ratio of the mean and the standard deviation [68]:

$$\text{SNR} = \frac{\bar{x}}{s}$$

where s is the sample standard deviation and $\bar{x}$ is the sample mean. This method allows us to compute the SNR even in a “black-box” scenario, i.e., without knowing the leaking intermediate value (nor the masks). However, another method could be used to determine if the signal is clean, such as computing the normalized inter-class variance (NICV) [69] or a simple visual inspection.

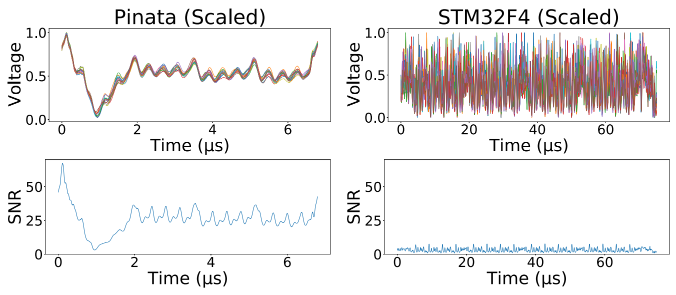

Fig. 7. SNR of Pinata (left) and STM32F4 (right).

Figure 7 shows the SNR of the Piñata and STM32F4 traces. Before computing the SNR, we have normalized the value of the traces between 0 and 1. This ensures that the magnitude differences observed in the SNR plots are due to the presence or absence of noise and not to a difference in scale. In Figure 7 we can observe how the SNR is about ten times higher than in the STM32F4. This confirms what can be seen with the naked eye: the STM32F4 traces contain a lot of measurement noise.

To the best of our knowledge, there is no exact threshold in the literature that indicates the minimum SNR for applying PCA. In any case, in our experiments, we have observed that an SNR lower than 10 (in these conditions) can be an indicator not to use PCA.

VII. Improved EDA-Based TA: Experimental Results

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 [32]. Finally, we draw some conclusions about the experiments. Note that, although in the previous section we have specified which improvements to apply in each case, in this section we apply all the variants to compare the performance of the different approaches.

A. 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 (Random 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.

1) The ASCAD Dataset: ASCAD [34] 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 [34]). The dataset is divided into two parts: fixed key and random

TABLE IV
PARAMETERS OF THE ATTACK (ASCAD)

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

key. Although many related works use the fixed key version for being an easier problem [55], [67], [70], [71], for this work we are using the random 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 (random 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 [34]. As the sensitive intermediate value we use the Hamming Weight of an S-box output: $Y^{(i)}(k^*) = \mathrm{SBox}[P_3^{(i)} \oplus k^*]$.

2) Experimental Results on ASCAD: As mentioned before, the selection of a proper number of PCs is not so straightforward. Some previous works have already performed attacks on ASCAD (fixed key) using TAs combined with PCA for POI selection. For instance, in the ASCAD introductory paper [34], among other relevant papers [55], [70], authors tested different number of first PCs to perform the attacks. This motivated us to combine EDAs and PCA since our approach is able to select the best PCs, not in sequential order, i.e., the best number of (first) components, but the optimal components, i.e., which components, in particular, provide the best results. To the best of our knowledge, there are no papers that implement TAs combined with PCA in the ASCAD random key version. Only in the fixed key version [34], [55], [70], which makes them not very realistic attacks. Therefore, in this paper, we not only test the performance of selecting a number of PCs in sequential order for a "regular" TA, but also we enhance these results by using EDAs for PC selection.

Table IV summarizes the parameters of the automated attacks. Figure 8 (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 [51]), 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.

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 III). Fig. 8 (right) show

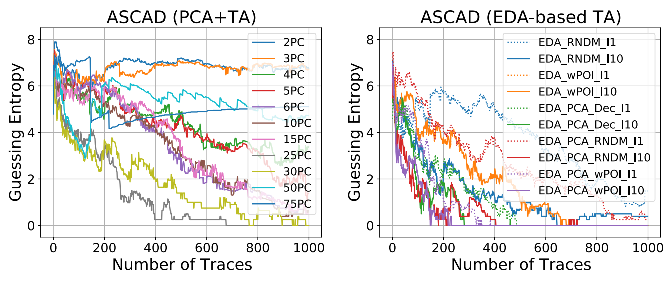

Fig. 8. ASCAD random keys: regular PCA+TA (left) and EDA-based TAs (right).

TABLE V
"RAW EDA" APPROACH VS "PCA+EDA" APPROACH (ASCAD RANDOM KEY)

Metric	**EDA (Rndm)**	**EDA (wPOI)**	**EDA+PCA (Rndm)**	**EDA+PCA (Dec)**	**EDA+PCA (wPOI)**
SR (%)	**12.00%**	**71.00%**	**72.00%**	**94.60%**	**98.20%**
$n_{Success}$	60	355	360	473	491
$n_{Attacks}$	500	500	500	500	500
CR (%)	**91.58%**	**99.80%**	**94.39%**	**100.00%**	**99.80%**
n_{Trials}	43	2	29	1	2
$\mathbf{ge_{acc_{avg}}}$	**65.57**	**17.61**	**21.53**	**3.01**	**1.76**
ge_{acc}	32786.75	8806	10765.75	1503.75	877.75
$\varepsilon_{\mathbf{Gen}}$ **(%)**	**-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

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 V), 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. 8 (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 [72]–[74] Table VI shows a comparison of the best performing attacks on ASCAD Random Keys (using the Hamming Weight model) in terms of number of attack traces required to reach GE = 0 ($\bar{N}t_{GE}$).

B. 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

TABLE VI
TOP RESULTS ON ASCAD (RANDOM KEY) WITH HAMMING WEIGHT MODEL

	DL-based [72]	DL-based [73]	DL-based [74]	EDA-Based TA
$\bar{N}t_{GE}$	470	496	911	≈ 200

TABLE VII
PARAMETERS OF THE ATTACK (STM32F4)

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

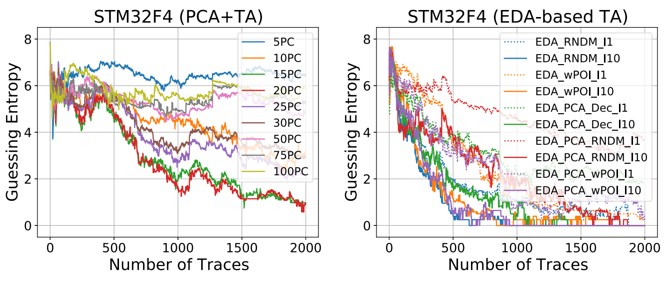

Fig. 9. Attacks on STM32F4: regular PCA + TA (left) and EDA-based TAs (right).

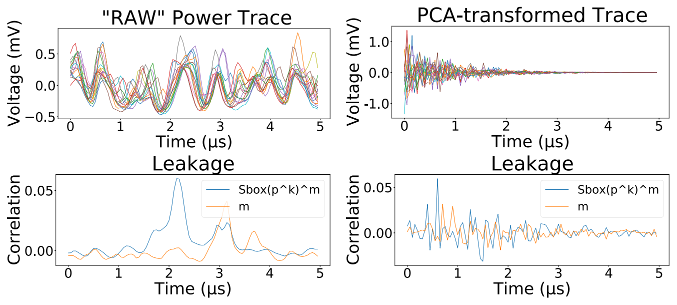

Fig. 10. STM32F4: Raw power traces (left), PCA-transformed traces (right) and leakage correlation.

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. 10) instead of using the full window of 16 lookups. This makes this experiment similar to the previous one with ASCAD. Table VII summarizes the parameters of the attacks.

As in the previous experiment, Fig. 9 (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 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.

Again, we compare the "raw" EDA-based attack and the attack on the "PCA-Transformed" traces (Fig. 9 (right)). Please note that the results are shown in the same manner as in the

TABLE VIII
"RAW EDA" APPROACH VS "PCA+EDA" APPROACH (STM32F4)

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%**
$n_{Success}$	430	377	341	136	373
$n_{Attacks}$	500	500	500	500	500
CR (%)	**99.80%**	**100.00%**	**97.39%**	**77.56%**	**97.80%**
n_{Trials}	2	1	14	113	12
$\mathbf{ge_{acc_{avg}}}$	**8.46**	**9.88**	**19.08**	**37.74**	**13.63**
ge_{acc}	4232.25	4941.5	9539	18870.5	6815
$\boldsymbol{\varepsilon_{Gen}}$ **(%)**	**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

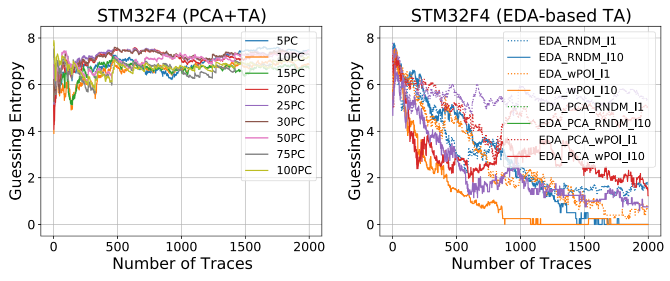

Fig. 11. Attacks on STM32F4 (Full Window): regular PCA + TA (left) and EDA-based TAs (right).

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 VIII 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. 9 (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. 10, 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.

1) 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

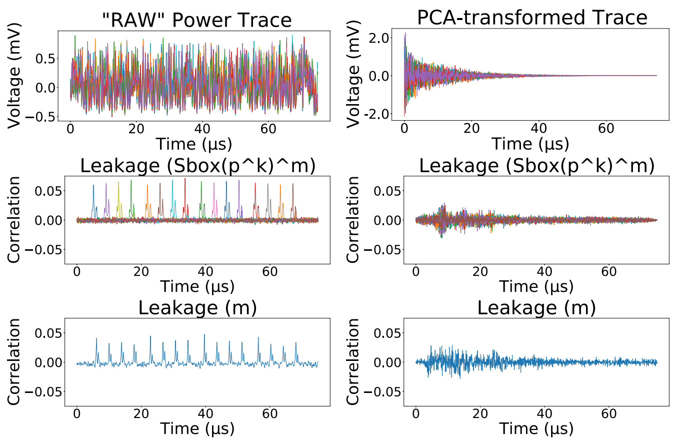

Fig. 12. STM32F4 (Full Window): Raw power traces (left), PCA-transformed traces (right) and leakage correlation.

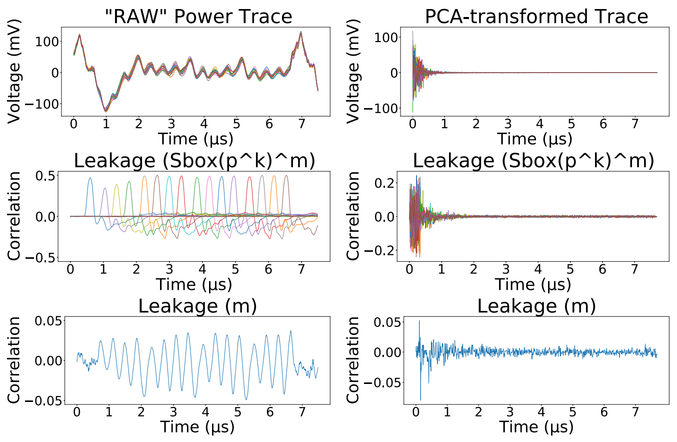

Fig. 13. Pinata (Full Window): Raw power traces (left), PCA-transformed traces (right) and leakage correlation.

are extremely defective (See Fig. 11 and Table IX). With the full window, we cannot succeed with a "traditional" TA using PCA for POI selection (Fig. 11 (left)). Regarding EDA-based attacks, only attacks without PCA provide particularly good results (Fig. 11 (right)). To understand this, one should take a look at Figures 12 and 13.

Fig. 12 shows "raw" power traces, PCA-transformed traces and their respective leakage graphics for the STM32F4 board (Full Window). Fig. 13 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. V-A.

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. 12). Moreover, the magnitude of the leakage has decreased substantially. This explains why the "traditional" TA+PCA does not work in this setup (Fig. 11, left). Regarding the Scree Test, it suggests

TABLE IX
"RAW EDA" Vs "PCA+EDA" (STM32F4 - Full Window)

Metric	EDA (Rndm)	EDA (wPOI)	EDA+PCA (Rndm)	EDA+PCA (Dec)	EDA+PCA (wPOI)
SR (%)	**58.60%**	**66.00%**	**2.40%**	**Null**	**3.40%**
$n_{Success}$	293	330	12	Null	17
$n_{Attacks}$	500	500	500	Null	500
CR (%)	**98.20%**	**96.99%**	**65.33%**	**Null**	**56.91%**
n_{Trials}	10	16	174	Null	219
$\mathbf{ge_{acc_{avg}}}$	**38.32**	**25.20**	**77.62**	**Null**	**71.90**
ge_{acc}	19161.75	12598.75	38811.25	Null	35949.75
$\varepsilon_{\mathbf{Gen}}$ **(%)**	**-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

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 IX, 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.

C. 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.

VIII. Conclusion 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 without user intervention.

We have shown that masking schemes like Masking Scheme 1 or the one used in ASCAD are especially weak. Our approach shows whether there exist close manipulation of the mask and masked intermediate value and hence possible unintended interactions making the scheme weak against profiling attacks. 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. Besides, although we have chosen software AES implementations for demonstrating our approach, we claim that the approach could be extendable to other use cases and is a good starting point for future work that will consider AES hardware implementations, implementations of other ciphers, other dimensionality reduction techniques like LDA, or other kinds of profiling attacks.

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.

Appendix A
Results on ASCAD (Random Keys) With Desynchronization = 50

The ASCAD traces we have used for the experiments do not feature desynchronization. In fact, applying EDA-based TAs over misaligned traces goes against the very nature of template attacks. On the one hand, ASCAD was created as a benchmarking reference for DL-based SCAs [34]. Note that neural networks can deal with slight shifts in the signal [75].

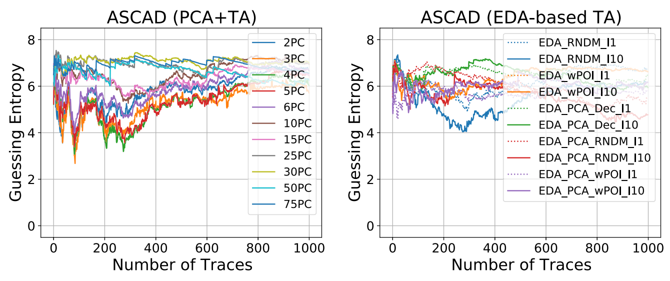

Fig. 14. ASCAD Random Keys (desync = 50): regular PCA+TA (left) and EDA-based TAs (right).

However, some works have shown that, rather than eliminating the need to realign traces, DL approach merely mitigates it, as pre-processing can significantly improve the performance of neural networks [76]. Besides, DL-based attacks present some other disadvantages against EDA-based PAs, as shown in [64]. On the other hand, when traces are misaligned (due to a poor trigger signal or random delays introduced by some countermeasure), one has to apply a resynchronization method before running the template attack [9]. There exist different tools for this propose: static alignment [53], wavelet transform [77] or elastic alignment [78], among other solutions [79]–[81]. When the misalignment is small, a larger number of traces can sometimes compensate for it. In ASCAD, the proposed misalignment is 50 or 100 samples, which is too much for the attack to work (without realigning the traces), at least using the same setup.

In any case, to illustrate this issue we have repeated the same experiment but with desynchronization of 50 samples (See Fig. 14). Note that, although the attack does not work under these conditions, a resynchronization of the traces is trivial in this case, which would provide results comparable to those of Sect. VII-A2.

Acknowledgment

The authors are thankful to an anonymous referees for their constructive comments that help them to improve the paper.

References

[1] EMVCo. (2001). *EMV Specifications*. [Online]. Available: https://www.emvco.com/

[2] Common Criteria. (Apr. 2017). *Common Criteria v3.1. Release 5*. [Online]. Available: https://www.commoncriteriaportal.org/cc/index.cfm

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO*, M. Wiener, Ed. Berlin, Germany: Springer, 1999, pp. 388–397.

[4] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES*. Germany, Berlin: Springer, 2004, pp. 16–29.

[5] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems CHES*. Berlin, Germany: Springer, 2008, pp. 426–442.

[6] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual information analysis: A comprehensive study," *J. Cryptol.*, vol. 24, no. 2, pp. 269–291, 2011.

[7] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems CHES*, Berlin, Germany: Springer, 2002, pp. 13–28.

[8] C. Rechberger and E. Oswald, "Practical template attacks," in *Information Security Applications*. Berlin, Germany: Springer, 2005, pp. 440–456.

[9] M. O. Choudary and M. G. Kuhn, "Efficient, portable template attacks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 2, pp. 490–501, Feb. 2018.

[10] G. Hospodar, B. Gierlichs, E. D. Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: A first study," *J. Cryptograph. Eng.*, vol. 1, no. 4, pp. 293–302, Oct. 2011.

[11] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," *J. Cryptograph. Eng.*, vol. 5, no. 2, pp. 123–139, Jun. 2015.

[12] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Proc. SPACE*. Cham, Switzerland: Springer, Dec. 2016, pp. 3–26.

[13] S. Picek, I. P. Samiotis, J. Kim, A. Heuser, S. Bhasin, and A. Legay, "On the performance of convolutional neural networks for side-channel analysis," in *Security, Privacy, and Applied Cryptography Engineering*, Cham, Switzerland: Springer, 2018, pp. 157–176.

[14] L. Masure, C. Dumas, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 348–375, Nov. 2019.

[15] V. Lomne, *Common Criteria Certification of a Smartcard: A Technical Overview*. Lausanne, Switzerland: CHES 2016 Tutorial #1, 2016.

[16] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," in *Proc. NIST Non-Invasive Attack Testing Workshop*, 2011, pp. 1–15.

[17] F. Standaert, "How (not) to use welch's t-test in side-channel security evaluations," in *Proc. CARDIS*, 2018, pp. 65–79.

[18] A. Moradi, B. Richter, T. Schneider, and F.-X. Standaert, "Leakage detection with the X^2-test," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 1, no. 2018, pp. 209–237, Feb. 2018.

[19] T. Moos, F. Wegener, and A. Moradi, "DL-LA: Deep learning leakage assessment: A modern roadmap for SCA evaluations," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 3, pp. 552–598, Jul. 2021.

[20] V. Cristiani, M. Lecomte, and P. Maurine, "Leakage assessment through neural estimation of the mutual information," in *Proc. Appl. Cryptogr. Netw. Secur. Workshops*, 2020, pp. 144–162.

[21] *Information Technology Security Techniques Testing Methods for the Mitigation of Non-Invasive Attack Classes Against Cryptographic Modules*, Standard ISO/IEC 17825:2016, International Organization for Standardization, 2016. [Online]. Available: https://www.iso.org/standard/60612.html

[22] C. Whitnall and E. Oswald, "A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules')," in *Proc. 25th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Cham, Switzerland: Springer, Nov. 2019, pp. 256–284.

[23] D. McCann, E. Oswald, and C. Whitnall, "Towards practical tools for side channel aware software engineering: 'Grey box' modelling for instruction leakages," in *Proc. 26th USENIX Secur. Symp.*, Vancouver, BC, Canada: USENIX Association, Aug. 2017, pp. 199–216.

[24] Y. L. Corre, J. Großschädl, and D. Dinu, "Micro-architectural power simulator for leakage assessment of cryptographic software on ARM cortex-M3 processors," in *Constructive Side-Channel Analysis and Secure Design*. Springer: Cham, Switzerland, 2018, pp. 82–98.

[25] M. A. Shelton, N. Samwel, L. Batina, F. Regazzoni, M. Wagner, and Y. Yarom, "Rosita: Towards automatic elimination of power-analysis leakage in ciphers," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–16.

[26] L. Lerman, R. Poussier, O. Markowitch, and F.-X. Standaert, "Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: Extended version," *J. Cryptograph. Eng.*, vol. 8, no. 4, pp. 301–313, Nov. 2018.

[27] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz, "Auto-tune POIs: Estimation of distribution algorithms for efficient side-channel analysis," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108405.

[28] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Advances in Cryptology—EUROCRYPT 2009*, A. Joux, Ed. Berlin, Germany: Springer, 2009, pp. 443–461.

[29] Riscure. *Piñata Board Brochure*. Accessed: May 19, 2022. [Online]. Available: https://www.riscure.com/uploads/2017/07/pi%C3%B1ata_board_brochure.pd%f

[30] STMicroelectronics. *Discovery Kit With STM32F411VE MCU*. Accessed: May 19, 2022. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f411ve.html%

[31] Federal Information Processing Standard. (Nov. 2001). *FIPS 197: Announcing the Advanced Encryption Standard (AES)*. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[32] AES-RA. (2021). *The AES_RA Dataset*. [Online]. Available: https://github.com/AES-RA/AES_RA

[33] R. Johnson and D. Wichern, *Applied Multivariate Statistical Analysis*, 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[34] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 163–188, Jun. 2020.

[35] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz, "Towards automatic and portable data loading template attacks on microcontrollers," in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 437–443.

[36] L. Wu *et al.*, "On the attack evaluation and the generalization ability in profiling side-channel analysis," Cryptol. ePrint Arch., Tech. Rep. 2020/899, 2020. [Online]. Available: https://ia.cr/2020/899

[37] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F.-X. Standaert, *On the Cost of Lazy Engineering for Masked Software Implementations*, vol. 8968, M. Joye, Ed. Cham, Switzerland: Springer, 2014, pp. 64–81.

[38] L. Bohy, M. Neve, D. Samyde, and J.-J. Quisquater, "Principal and independent component analysis for crypto-systems with hardware unmasked units," *Proc. e-Smart*, Jan. 2003, pp. 1–9.

[39] C. Archambeau, E. Peeters, F. Standaert, and J. Quisquater, "Template attacks in principal subspaces," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2006, pp. 1–14.

[40] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2008, pp. 411–425.

[41] E. Saeedi and Y. Kong, "Side channel information analysis based on machine learning," in *Proc. 8th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Dec. 2014, pp. 1–7.

[42] N. Mukhtar, M. Mehrabi, Y. Kong, and A. Anjum, "Machine-learning-based side-channel evaluation of elliptic-curve cryptographic FPGA processor," *Appl. Sci.*, vol. 9, no. 1, p. 64, Dec. 2018.

[43] A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury, "Practical approaches toward deep-learning-based cross-device power side-channel attack," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 2, pp. 2720–2733, Dec. 2019.

[44] L. Weissbart, Ł. Chmielewski, S. Picek, and L. Batina, "Systematic side-channel analysis of Curve25519 with machine learning," *J. Hardw. Syst. Secur.*, vol. 4, no. 4, pp. 314–328, Dec. 2020.

[45] N. Mukhtar, A. P. Fournaris, T. M. Khan, C. Dimopoulos, and Y. Kong, "Improved hybrid approach for side-channel analysis using efficient convolutional neural network and dimensionality reduction," *IEEE Access*, vol. 8, pp. 184298–184311, 2020.

[46] H. Feng, W. Lin, W. Shang, J. Cao, and W. Huang, "MLP and CNN-based classification of points of interest in side-channel attacks," *Int. J. Netw. Distrib. Comput.*, vol. 8, no. 2, pp. 108–117, 2020.

[47] E. Cagli, C. Dumas, and E. Prouff, "Enhancing dimensionality reduction methods for side-channel attacks," in *Smart Card Research and Advanced Applications*. Cham, Switzerland: Springer, 2016, pp. 15–33.

[48] S. Picek, A. Heuser, A. Jovic, and L. Batina, "A systematic evaluation of profiling through focused feature selection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 2, pp. 2802–2815, Sep. 2019.

[49] D. Mavroeidis, L. Batina, T. van Laarhoven, and E. Marchiori, "PCA, eigenvector localization and clustering for side-channel attacks on cryptographic hardware devices," in *Machine Learning and Knowledge Discovery in Databases*. Cham, Switzerland: Springer, 2012, pp. 253–268.

[50] Y. Souissi, M. Nassar, S. Guilley, J.-L. Danger, and F. Flament, "First principal components analysis: A new side channel distinguisher," in *Information Security and Cryptology ICISC 2010*. Berlin, Germany: Springer, 2011, pp. 407–419.

[51] I. T. Jolliffe, *Principal Component Analysis* (Springer Series in Statistics). Cham, Switzerland: Springer, 2002.

[52] R. Specht, J. Heyszl, M. Kleinsteuber, and G. Sigl, "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*. Cham, Switzerland: Springer, 2015, pp. 3–19.

[53] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards* Cham, Switzerland: Springer, 2007.

[54] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Cryptographic Hardware and Embedded Systems CHES*. Berlin, Germany: Springer, 2005, pp. 30–46.

[55] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, May 2019.

[56] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[57] R. A. Fisher, "The statistical utilization of multiple measurements," *Ann. Eugenics*, vol. 8, no. 4, pp. 376–386, Aug. 1938.

[58] L. Smith. (2002). *A Tutorial on Principal Components Analysis*. [Online]. Available: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_component%s.pdf

[59] J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms* (Studies in Fuzziness and Soft Computing), vol. 192, 1st ed. Berlin, Germany: Springer, Jan. 2006.

[60] P. Larranaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation* (Genetic Algorithms and Evolutionary Computation), vol. 2, 1st ed. New York, NY, USA: Springer, 2002.

[61] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021.

[62] Urioja. (2021). *EDA-based SCA Toy Example*. [Online]. Available: https://github.com/urioja/EDA-Based_SCA_ToyExample

[63] T. Chen, K. Tang, G. Chen, and X. Yao, "On the analysis of average time complexity of estimation of distribution algorithms," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 453–460.

[64] U. Rioja, L. Batina, I. Armendariz, and J. L. Flores, "Keep it unbiased: A comparison between estimation of distribution algorithms and deep learning for human interaction-free side-channel analysis," *CoRR*, vol. abs/2111.13425, 2021. [Online]. Available: https://arxiv.org/abs/2111.13425

[65] Kokke. (2014). *Tiny AES in C*. [Online]. Available: htps://github.com/kokke/tiny-AES-c

[66] STMicroelectronics. (Dec. 2016). *STM32F411VET6 Datasheet*. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/929991/STMICROELECTRONIC%S/STM32F411VET6.html

[67] Y. Zotkin, F. Olivier, and E. Bourbao, "Deep learning vs template attacks in front of fundamental targets: Experimental study," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 1213, Jan. 2018.

[68] NIST. (Mar. 2017). *Signal to Noise Ratio*. [Online]. Available: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/snr.%htm

[69] S. Bhasin, J.-L. Danger, S. Guilley, and Z. Najm, "NICV: Normalized inter-class variance for detection of side-channel leakage," in *Proc. Int. Symp. Electromagn. Compat.*, May 2014, pp. 310–313.

[70] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 1–36, Nov. 2019.

[71] Y.-S. Won, X. Hou, D. Jap, J. Breier, and S. Bhasin, "Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3215–3227, 2021.

[72] G. Perin, Ł. Chmielewski, and S. Picek, "Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 337–364, Aug. 2020.

[73] L. Wu, G. Perin, and S. Picek, "I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1293, Jan. 2020.

[74] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 677–707, Jul. 2021.

[75] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *CHES 2017*, W. Fischer and N. Homma, Eds. Cham, Switzerland: Springer, 2017, pp. 45–68.

[76] Y. Zhou and F.-X. Standaert, "Deep learning mitigates but does not annihilate the need of aligned traces and a generalized ResNet model for side-channel attacks," *J. Cryptograph. Eng.*, vol. 10, no. 1, pp. 85–95, Apr. 2020.

[77] X. Charvet and H. Pelletier, "Improving the DPA attack using wavelet transform," in *Proc. Phys. Secur. Test. Workshop*, vol. 46, 2005, pp. 1–15. [Online]. Available: https://csrc.nist.rip/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper14.pdf

[78] J. G. J. van Woudenberg, M. F. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *Topics in Cryptology—CT-RSA 2011*, A. Kiayias, Ed. Berlin, Germany: Springer, 2011, pp. 104–119.

[79] R. Muijrers, J. V. Woudenberg, and L. Batina, "RAM: Rapid alignment method," in *Proc. CARDIS*, 2011, pp. 266–282.

[80] S. Nagashima, N. Homma, Y. Imai, T. Aoki, and A. Satoh, "DPA using phase-based waveform matching against random-delay countermeasure," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1807–1810.

[81] K. M. Abdellatif, D. Couroussé, O. Potin, and P. Jaillon, "Filtering-based CPA: A successful side-channel attack against desynchronization countermeasures," in *Proc. 4th Workshop Cryptogr. Secur. Comput. Syst.*, Jan. 2017, pp. 29–32.

Unai Rioja received the Diploma degree in industrial electronics and automaton engineering and the M.Sc. degree in industrial engineering from the University of La Rioja, Spain, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Digital Security Group, Radboud University, Nijmegen, The Netherlands. He is also a part of the Cybersecurity in Embedded Systems Team, Ikerlan Technological Research Center. Currently, he is working as a Researcher with Ikerlan. His main research interests are in the area of analysis, design, and evaluation of embedded systems and mainly focus on cybersecurity for embedded systems and side-channel attacks.

Lejla Batina (Senior Member, IEEE) received the Professional Doctorate degree in engineering from the Eindhoven University of Technology in 2001 and the Ph.D. degree from KU Leuven, Belgium, in 2005. She worked as a Cryptographer at SafeNet B.V. (2001–2003). She is currently a Professor in embedded systems security with Radboud University, Nijmegen, The Netherlands. Her research group consists of 12 researchers and eight Ph.D. students have graduated under her supervision. She is an Editorial Board Member of top journals in security IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and *ACM Transactions on Embedded Computing Systems*.

Igor Armendariz received the Ph.D. degree from the University of the Basque Country in 1996. He was a Researcher with CEIT from 1995 to 1996. He was a Professor in computer science with the University of the Basque Country from 1996 to 2000. He began working with the Communication Department, Ikerlan, from 2000 to 2015. He is currently a Researcher with the Cybersecurity in Embedded Systems Team, Ikerlan Technological Research Center. He is also a part of the Industrial Cybersecurity Department. Cybersecurity specialist acc. to IEC-62443-4-1, IEC-62443-4-2 (TÜV Rheinland, Components) #230/19. He is co-supervising a couple of Ph.D. students, working with side-channel attacks and countermeasures. He received the Scholarship Award at CEIT from 1991 to 1994.

Jose Luis Flores received the M.Sc. degree in robotics and advanced control from the University of the Basque Country. He is currently a Researcher with the Cybersecurity in Embedded Systems Team, Ikerlan Technological Research Center. As such, the main lines he works on in each organization are embedded system security at Ikerlan and machine learning and optimization at the university. His main interests are related to artificial intelligence and cybersecurity.