When Similarities Among Devices are Taken for Granted: Another Look at Portability

Unai Rioja[1,2](✉), Lejla Batina[1](✉), and Igor Armendariz[2](✉)

[1] Digital Security Group, Radboud University, Nijmegen, The Netherlands
lejla@cs.ru.nl

[2] Ikerlan Technology Research Centre, Arrasate-Mondragón, Gipuzkoa, Spain
{urioja,iarmendariz}@ikerlan.es

Abstract. The original idea of profiling implies attacking one device with a leakage model generated from an "identical copy", but this concept cannot be always enforced. The leakage model is commonly generated with traces from an "open device", assuming that a model which works for one device should work for another copy as well. In practice, applying a leakage model to a different copy of the same device (commonly called portability) is a hard problem to deal with, as intrinsic differences in the devices or the experimental setups used to obtain the traces cause behavioural variations which lead to an unsuccessful attack. In this paper we propose a novel *similarity assessment* technique that allows evaluators to quantify the differences among various copies of the same device. Moreover, we support this technique with actual experiments to show that this metric is directly related to the portability issue. Finally, we derive a method that improves the performance of template attacks.

Keywords: SCA · Profiling attacks · Template attacks · Portability · SCA evaluation · DTW

1 Introduction

Nowadays profiling attacks are considered the most powerful kind of Side-Channel Attacks (SCAs). The idea behind profiling is different from the traditional concept of Differential Power Analysis (DPA) [3,4,10,19], for which the attack is separated from the device (e.g., every device that implements AES encryption is susceptible to the attacks using these techniques). In profiling attacks, the goal of the attacker is to build a leakage model of a particular device and to recover sensitive information comparing that model with the actual power consumption of the device. The first requirement to carry out this kind of SCA is to have, at least, two devices: the attacked device or the *device under test* (DUT), and another identical hardware device over which we have full control. The reason behind is that the attack requires two different stages: a profiling stage (with a "copy" of the device), in which we model the power consumption (side-channel), and an attack phase (with the "real" device), in which we use the generated model to obtain the secret parameter with only one or a few traces. Conversely, even though the original idea of a profiling attack is to generate the power consumption model for an "identical" copy of the attacked device, this is not always guaranteed in practice. This portability issue is often underestimated in practice, although some previous works suggest that in real-world setups small differences in the production of different devices, aging or even environmental changes during the measurements cause different behaviours of those devices, even leading to an unsuccessful attack [1,3,6,10,11,14,20,29].

In this work, our goal is to show how these dissimilarities can be addressed from the evaluation point of view, measuring how similar two different devices are and giving insights on how successful a portable profiling attack could be. Moreover, although performing this kind of attack is challenging, in this work we show the feasibility of a portable template attack in a realistic setup. We also provide some suggestions to improve the success rate of these attacks with our new point of interest (POI) selection technique.

Our main contribution is a novel *similarity assessment* technique with which, from an evaluation point of view, general similarities/dissimilarities between "identical" copies of the same device can be quantified. In addition, our work has revealed some other contributions that are all detailed as follows:

- This paper proposes the usage of the well-known Dynamic Time Warping (DTW from now on) statistical tool as a *similarity assessment* tool. Our approach shows that the warped distance between two specific graphics can quantify the similarities/dissimilarities between different devices or tracesets.
- We showcase the proposed technique with several experiments (portable template attacks with four different copies of the same device), demonstrating that the performance of the attack is directly related to this metric and hence the more similar two copies of the same device are (or two different sets of traces from the same device) the better results will be obtained.
- Finally, we propose an alternative POI selection technique which helps improving the performance of portable template attacks. This technique is also supported by the aforementioned experiments, showing how an unsuccessful portable attack can be turned into a successful attack by choosing the "best" points of interest while building the templates.

The paper is organized as follows, Sect. 2 summarizes the state of the art and the related work on this topic. Section 3 highlights the common issues with portability as a starting point for our work. Section 4 explains the details of DTW and our *similarity assessment* technique. Section 5 contains the experimental results supporting our *similarity assessment* technique and our *new POI selection* technique (which is also explained in a practical manner in this section). Finally, Sect. 6 draws the conclusions.

2 State of the Art

As mentioned above, profiling attacks are dominant in side-channel analysis nowadays. In the *profiling phase*, the model of the device can be generated by using standard classification techniques like in Template attacks [7,28], Support Vector Machine (SVM) [15,16,21], Random Forest (RF) [22], regression or the Stochastic models approach [32] or recently introduced Deep learning techniques [5,24,27]. In the *attacking phase*, the model is applied and the secret key is guessed. Template attacks and machine learning are the two most popular approaches [23].

In this work, we focus on classical Template attacks because it is a well-known and understood technique in the field of SCA. Moreover, it should be noticed that although template attacks usually require more effort of an expert with signal processing capabilities, they allow the attacker/evaluator to focus on specific parts of the leakage keeping more control in the process (which is not always possible with deep learning techniques).

2.1 Template Attacks

Template attacks are the original form of profiling attacks as proposed for SCA by Chari et al. [7]. These attacks are based on building a multivariate model of the probability distribution of the leakage. The Probability Density Function (PDF) is usually computed assuming that the leakages follow a Gaussian distribution, as in the case of unprotected devices (devices without SCA countermeasures). This is a parametric estimation and we focus on this kind of templates technique because of its fast convergence and the fact that it is widely used and consolidated by many previous works. Nevertheless, it should be mentioned that there are other non-parametric estimations that are able to capture any distribution (which might be helpful with protected devices) like histogram and kernel-based estimators and also some other advanced tools [33].

The main goal in a "traditional" template attack is to deduce the secret (key) used to perform cryptographic operations. Thus, the attacker has to first take measurements of some device's physical property (commonly the power consumption or the electromagnetic radiation emitted by the device) during the manipulation of some intermediate value $iv = f(p, k)$ related to the plaintext p and the secret key k. In the *profiling phase* the attacker uses a set of n_p profiling traces ($T_{p,k}$) to build a Gaussian multivariate model (pdf) for each possible *iv*. In order to do that, the mean vector $\mu_{p,k}$ and the covariance matrix $\sum_{p,k}$ are estimated for each *iv*, creating the so-called *templates*. Then, in the *attack phase*, from a set of n_a real power traces and its input data (plaintext), the attacker tries to guess the correct *iv* value (or its Hamming Weight) by using the maximum likelihood principle. Since $iv = f(p, k)$, knowing *iv* and p the secret key k can be recovered.

Template attacks are optimal from an information-theoretic point of view but in practice, they have several limitations: preprocessing dependency (the effort of an expert in the field is mandatory most of the times), computational complexity problems and the need for dimensionality reduction. The latter is usually solved by selecting only a subset of the typically huge amount of samples in each power trace (Points of Interest [POI] selection [28]), applying another data-dimensionality reduction method as Principal Component Analysis (PCA)

[2,34] or Fisher's Linear Discriminant Analysis (LDA) [12,18]. Due to the high computational requirements of the other techniques, in this work, we reduce the dimensionality of the problem by selecting a few samples (Points of Interest). As we have mentioned, it also allows us to focus only on specific parts of the leakage and improve the results of the "classical" template attack (Sect. 5).

2.2 Portability

Although having two identical devices to perform a profiling attack is mandatory, in practice this is not always possible. The traces for the profiling phase and the attack phase are usually captured from the same device in most of the works on this topic [7,9,13]. Attacking a second device with a model generated with a first device is often considered trivial, while in practice this is not the case. Even if two devices are clones ("identical" copies of the same device) there always exist differences in the construction of the devices that can cause different behaviours in timing, voltage, etc. There could be different reasons for this, such as faults in the manufacturing process, aging, slight differences in the resistance and/or capacitance of the circuit, etc. Moreover, when two measurements are taken in different time moments they will often cause deviations in the acquired power traces, which could lead to an unsuccessful attack [1,11,29]. More precisely, we are referring to various small variations in the experimental setup such as I/O interference (serial port, USB, Radio, etc.), influence of the past state, memory management, garbage collection, differences in magnetic field penetration (while taking electromagnetic measurements), changes in environmental parameters (temperature, humidity, electromagnetic noise, etc.), resonance due to LC and RC oscillators, among other phenomenon.

To the best of our knowledge, there are not many papers discussing the portability of profiling attacks. The work of Elaabid et al. is introducing the portability issue and showing how waveform realignment and acquisition campaigns normalization can improve the performance of portable template attacks [11]. The work of Choudary et al. focuses on differences between devices when performing portable template attacks while attacking four different copies of the same device [8,10]. A more recent work successfully implements a portable template attack over a wireless keyboard performing AES encryption [20]. In the CHES 2018 Side Channel Contest CTF, portability was also considered, and the winning attack was able to obtain a 100% of success in all devices [14]. In [6], authors considered the usage of several devices during the various stages of a profiling attack in order to attack an RSA implementation. Bhasin et al. have recently made a comparison between different machine learning techniques using portable profiling attacks, but they only focused on machine learning techniques [3].

Our approach is orthogonal to all those works as we show a general way to find and quantify differences between clone devices, by obtaining a measurement of its dissimilarity. This metric is directly linked to portability: the more similar two devices are, the better performance the portable template attack will achieve and vice versa. Moreover, we show how this information can be used in the POI selection and we propose an improved POI selection technique, which assists in finding the optimal leakage points for different devices.

3 The Issue of Portability

As mentioned in Sect. 2.2, to perform a portable profiling attack is a challenging task, mainly because the behaviour of two theoretically identical devices could be (slightly) different in practice. After several experiments we noticed that the differences between two "identical" devices can be seen clearly in the graphics used for POI selection. Those graphics (POI graphs from now on) are generated by applying certain functions to the power traces in order to find the leaking points and select proper POIs for Template Attacks. Below we describe some of the most commonly used techniques:

Pearson Correlation Coefficient: This is a widely used metric in statistics, which assesses the linear dependence between two variables x and y [17]. It takes on a value between -1 and 1, where 0 means no (linear) correlation and 1 and -1 imply the total positive and negative linear correlation respectively. We compute the Pearson correlation coefficient between the data manipulated by the target device and the power consumption traces of the device while processing the data. For a sample of the entire population, the coefficient is defined by Eq. (1):

$$Correlation(x,y) = \frac{\sum_{i=1}^{N}((x_i-\bar{x})(y_i-\bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i-\bar{y})^2}} \tag{1}$$

SOSD: SOSD or sum of squared differences was proposed for POIs in [13] and is defined by Eq. (2), where $\bar{x}_{y_i}$ is the mean of the power traces and the manipulated data is equal to y_i. Its value is always positive and highlights big differences in means.

$$SOSD(x,y) = \sum_{i,j>i}(\bar{x}_{y_i}-\bar{x}_{y_j})^2 \tag{2}$$

SOST: This is the normalized version of SOSD [13], which is equivalent to the pairwise Student's t-test. It is defined by Eq. (3), where n_{y_i} and n_{y_j} are the number of traces where y is equal to y_i and y_j respectively.

$$SOST(x,y) = \sum_{i,j>i}((\bar{x}_{y_i}-\bar{x}_{y_j})/\sqrt{\frac{\sigma_{y_i}^2}{n_{y_i}}+\frac{\sigma_{y_j}^2}{n_{y_j}}})^2 \tag{3}$$

SNR: Signal-to-noise ratios are commonly used in electrical engineering and signal processing. In the context of a side-channel attack, the SNR of a point of a power trace can be computed by Eq. (4), where P_{exp} is the exploitable power consumption and $P_{sw.noise}$ and $P_{el.noise}$ correspond to the noise component (switching noise and electronic noise). In a nutshell, it quantifies how much information is leaking from a point of the power trace. For a deeper explanation of the SNR calculation we refer to [25].

$$SNR = \frac{Var(P_{exp})}{Var(P_{sw.noise} + P_{el.noise})} \tag{4}$$

If we compare two POI graphs generated with traces from two "identical" copies of the same device, significant differences between them could be observed. The spikes do not occur at the same time and with the same shape (or strength), which can influence the portability of an attack. In Figs. 1, 2 and 3 these differences can be noticed. Figure 1 shows the power traces of two copies of the same device during some internal computations in which an 8-bit sensitive value is stored in memory (Voltage vs Time (samples)). In Fig. 2 the leaking part of the signal (the exact point in which the 8-bit data is stored) is shown. Also, Fig. 3 shows the output of the aforementioned POI selection functions of both devices. It is important to note that there are significant differences in the magnitude and shape of both graphs. Those differences are very problematic when porting a template attack: the POIs selected to generate the templates in the first device will not match with the optimal POIs in the second device. Thus, the portable template attack will probably fail. When performing a Template attack, the highest points of the spikes that appear in the POI selection graph are usually selected as POIs, in order to reduce the dimensionality of the multivariate leakage model and make the attack feasible. However, as these points are selected taking into account only the profiling device, a big spike for one device could be a "valley" in others. Moreover, if the spikes match, the results could still be bad if the value in the POI graph is too low. We should ideally select spikes with a value that is high enough to represent leakage. In conclusion, two devices

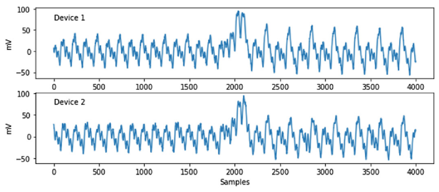


Fig. 1. Differences between devices: Power trace

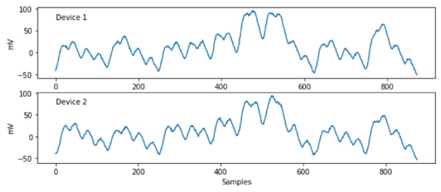


Fig. 2. Differences between devices: Leaking part of the power trace

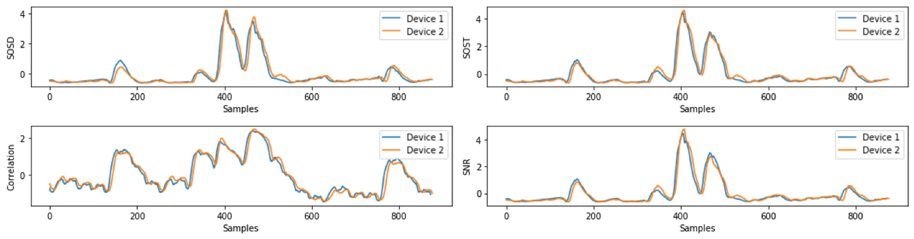


Fig. 3. Differences between POI selection functions

can be "identical" copies, but in practice there are often remarkable differences between their leaking points. When we build templates from one device and then try to attack another one, it is crucial that the selected POI represents a significant leakage in both devices. Otherwise, the behaviour modeled with the Gaussian multivariate distribution will not apply to the second device and the attack will fail.

4 Similarity Assessment

In order to quantify how different two "identical" copies of a device are, a *similarity assessment* technique is proposed next, which is based on the Dynamic Time Warping (DTW) statistical tool.

4.1 Dynamic Time Warping

DTW is a well-known algorithm to measure similarity between two temporal sequences and find the most similar points between them. In other words, this technique is able to quantify the similarity between two signals (even if they are not completely aligned) and obtain the optimal match (alignment). In Figs. 4 and 5 the difference between the "traditional" euclidean distance and the warped distance is demonstrated. Originally, DTW was used in speech recognition [30] but later on, it has been proved its applicability in several fields like gesture recognition, robotics, manufacturing, etc. This technique has been applied also in the SCA field with the elastic alignment [35] as a special kind of alignment using DTW (FastDTW [31], more precisely). This kind of alignment was proposed in order to address cryptographic implementations with random delay countermeasures. Afterwards, Muijrers et al. proposed another alignment algorithm which can deal with this countermeasure [26]. This method can align traces with less computational effort than elastic alignment (DTW algorithm is relatively computationally costly, depending on the length of the path). However, we now propose the usage of DTW algorithm for an entirely different task: assessing differences between tracesets or devices. Our approach is to use DTW to quantify how similar two devices are by measuring the similarity of two temporal sequences representing the leakage of each device.

Warp Path and Warped Distance are the two main outcomes of the DTW calculation. The former indicates the best alignment between two shapes while the latter is a measure of the similarity between signals. In our case, the DTW algorithm is applied to two discrete time signals (traces), $X = x_1, x_2, \ldots, x_i, \ldots, x_{|X|}$ and $Y = y_1, y_2, \ldots, y_j, \ldots, y_{|Y|}$. In order to compute the warp path and warped distance, the DTW algorithm calculates a *cost matrix*: an $|X|$-by-$|Y|$ matrix containing the distances between all samples of X and Y. Figure 6 shows an example of the cost matrix (and its warp path) of the two example curves shown in Figs. 4 and 5. Each element of the matrix (i, j) represents the distance $D(i, j)$ between samples x_i and y_i (the darker the cell is, the largest distance). The warping path (in dark blue), connects the cells with smaller

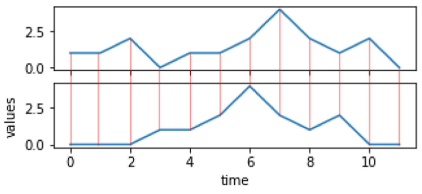


Fig. 4. Euclidian distance

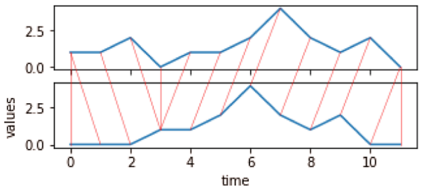


Fig. 5. Warped distance

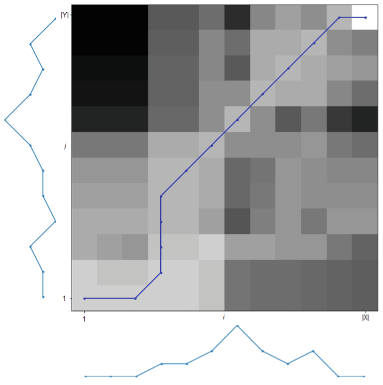


Fig. 6. Cost matrix and warp path (Color figure online)

distance starting from $(1, 1)$ to $(|X|, |Y|)$, and indicates the optimal alignment between those two curves (shown graphically in Fig. 5). The cumulative distances of the warping path is what we call warped distance (our similarity indicator between two time series). Formally speaking, the warping path W between two traces X and Y can be defined as:

$$W(X, Y) = (w_0, w_1, \ldots, w_K) \quad \text{where} \quad \max(|X|, |Y|) \leq K < |X| + |Y|$$

Here, K is the length of the warp path and $w_k = (i, j)$ the k^{th} element of the path. Also, this path has to follow several constraints. For $w_k = (i_k, j_k)$ and $w_{k-1} = (i_{k-1}, j_{k-1})$ being two consecutive elements of the warp path, the constraints are:

- **Monotonicity:** $i_{k-1} \leq i_k$ and $j_{k-1} \leq j_k$
- **Continuity:** $i_k - i_{k-1} \leq 1$ and $j_k - j_{k-1} \leq 1$
- **Bound:** $w_1 = (1, 1)$ and $w_K = (|X|, |Y|)$

The **boundary condition** ensures that every index of both time series is used in the warp path computation while **monotonicity** and **continuity** constraints assure that we do not skip any sample and we do not go backwards in time. Several paths could satisfy these conditions but the minimum-distance path is considered as the optimal warp path, where the distance is:

$$Dist(W) = \sum_{k=1}^{k=K} w_k$$

This minimum distance is what we call warped distance, a similarity measure between two time series. To find the minimum distance warp path, the distance D of each cell has to be computed. Dynamic programming can solve this problem in a very effective manner, so the value of a cell in the cost matrix is:

$$D(i,j) = d(i,j) + \min[D(i-1,j), D(i,j-1), D(i-1,j-1)]$$

Where d is usually computed as the typical Euclidean distance $d(i,j) = d(x_i,y_j) = (x_i - y_j)^2$ between samples x_i and y_i and D is known as the cumulative distance (the euclidean distance $d(i,j)$ plus the minimum of the cumulative distances of the contiguous cells).

4.2 Similarity Assessment Technique

Once the basis of the DTW technique have been clarified, our *similarity assessment technique* can be explained. In order to assess how similar two devices are, we propose the following steps:

1. **Step 1: Obtain POI graphics of both devices.** A set of n_{POI} traces will be captured from both devices. These traces must be taken when the device is manipulating a certain sensitive variable, which must have a random value each time in order to properly characterize the leakage. Once both sets of traces are taken ($n_{POI(1)}$ and $n_{POI(2)}$), two POI graphs are obtained by applying one of the POI selection functions mentioned in Sect. 3.

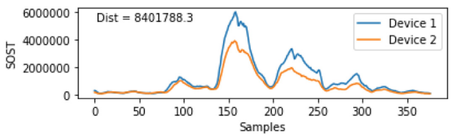


Fig. 7. SOST graph of devices 1 & 2 (Without Standardization)

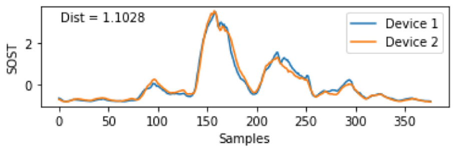


Fig. 8. SOST graph of devices 1 & 2 (With Standardization)

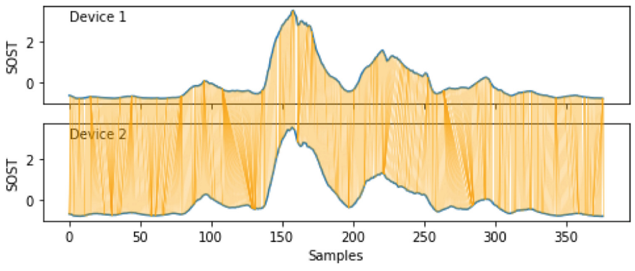


Fig. 9. Warped distance between Device 1 and Device 2

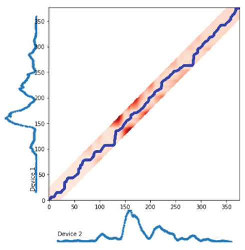


Fig. 10. Warp path between Device 1 and 2

2. **Step 2: Standardize both graphics.** Zero-mean and normalization are standard preprocessing techniques that are mandatory to apply after almost every SCA acquisition. Our case is not an exception: an standardization of both graphs before applying DTW can be helpful since DTW usually interprets those differences as huge dissimilarities. Figures 7 and 8 show how the distance value changes enormously depending on whether we apply this technique or not, even though both graphs are quite similar in shape. Our experimental results show how a portable template attack can be successful if the shapes of its POI graphs are similar enough, even if there are magnitude differences (as shown in Sect. 5). In other words, it is more important how the device leaks its information than the quantity of the leakage (as long as the leakage is big enough). Thus, we propose to standardize each POI graph using Eq. (5), where z is the re-scaled sample, x_i is the sample to scale and μ and σ are the mean and the standard deviation of the trace (Fig. 8).

$$z = \frac{x_i - \mu}{\sigma} \tag{5}$$

3. **Step 3: Compute the DTW algorithm** in order to obtain the minimum distance path (as explained above). In other words, by computing the cumulative of the distances of the minimum distance warping path we obtain the *distance between both graphs*: a quantitative measurement of the similarity/dissimilarity of both time sequences.

Additionally, the graphical representation of the warped distance and warping path (Fig. 9) can be helpful in the POI selection since DTW highlights the parts of the signal which are most similar between devices. To see graphically misalignment problems and behavioural differences between devices is a good starting point for an improved POI selection.

5 Experimental Results

In order to support our *similarity assessment* technique, we have performed realistic experiments involving template attacks with four "identical" copies of the same device (ATmega328P microcontroller) called Device 1 (D1), Device 2 (D2), Device 3 (D3) and Device 4 (D4). Additionally, we propose an improved POI selection technique which helps to enhance the performance of the portable template attack. We consider two main template attack use cases: using one device in the profiling phase and using two devices in the profiling phase.

5.1 Setup

The target is a development board mounting an ATmega328P 8-bit microcontroller working at 16 MHz clock frequency. We are storing random data (8-bit values) in flash memory using a memcpy() operation (in a random address each time). During that operation, we measure the power consumption of the device with a Tektronix CT1 current probe attached to a 20 GS/s digital oscilloscope

Table 1. Portable template attack experiments using Device 1 (D1) for profiling.

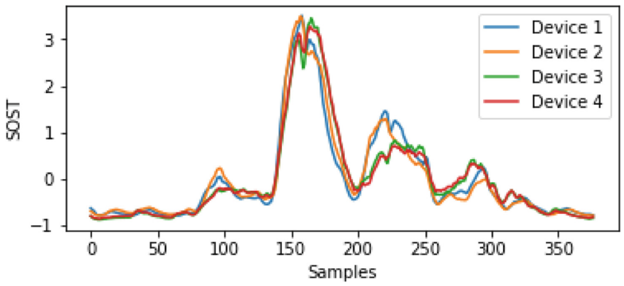
POI	Rank (D1 vs D1)	Rank (D1 vs D2)	Rank (D1 vs D3)	Rank (D1 vs D4)
[97, 158, 220, 294]	**1**	46	**17**	**18**

(LeCroy Waverunner 9104) triggered by the microcontroller, which rises a GPIO signal when the internal computation starts. Each power trace is formed by 400 samples taken at 1 GHz with 8-bit resolution. As an attacker, our goal is to obtain the exact 8-bit value loaded in flash memory using template attacks. A set of n_p profiling traces are taken from the profiling device(s) (storing random 8-bit values) and labeled with the stored value. The traces are preprocessed by aligning them and applying the aforementioned standardization technique. Then, a SOST function is ran in order to find possible POIs. 256 templates are built by computing the mean and co-variance matrix for each labeled group (in the selected POIs), using the pooled matrix optimization method. In the *attacking phase* a set of n_a power traces of the attacking device storing a fixed 8-bit value are taken. Then the multivariate model is applied and the 8-bit value is guessed using the maximum likelihood principle. Each label will obtain a confidence value and the 256 labels will be ranked. We consider the attack successful when the correct candidate obtains a rank of 25 or less (the correct candidate is in the top 10% of candidates). We assume that then, the correct value could be guessed using (optimized) brute force.

5.2 Use Case 1: Template Attack Using One Device in Profiling Phase

"Raw" Template Attack

In the *profiling phase* 20 000 power traces of D1 are taken and labeled with the stored value. The traces are preprocessed and the SOST function is run in order to find possible POI (Fig. 8, Device 1). Four significant spikes can be seen, corresponding to the different moments in which the copied variable leaks (production, travel across a bus, load into register, etc.). Thus, four POI are selected, one for each significant spike [97, 158, 220, 294] and the model using 256 templates is built. In the *attacking phase* 1 000 power traces of the same device (D1) storing a fixed value are taken. Then the multivariate model is applied and the 8-bit value is guessed. In Table 1 we can see that the rank of the correct candidate in this attack (D1 vs D1) is 1 after 1 000 traces (successful attack). In order to perform a more realistic template attack, 1 000 power traces from a second device (D2) storing the same fixed 8-bit value are taken. Then the model computed before with traces from D1 is applied and the fixed value is guessed. As it can be seen in Table 1, the attack is unsuccessful (the rank of the correct candidate in this attack (D1 vs D2) is 46) because the multivariate model of D1 does not apply to D2. The process is repeated with devices D3 and D4 (results appear in Table 1). In this case the attack is successful (the rank of the correct candidate is less than 25), but the results are not optimal. To enhance the model we apply our *similarity assessment* technique and the *improved POI selection* technique.



Devices	Distance
D1 vs D2	1,1028
D1 vs D3	2,4159
D1 vs D4	2,7628
D2 vs D3	2,4792
D2 vs D4	2,7153
D3 vs D4	0,9128

Fig. 11. SOST graphics of each one of the four analyzed devices and distances between them

Improving the Results

First and foremost, we apply our *similarity assessment* method. Thus, 20 000 power traces of the second device storing random 8-bit values are taken and labeled with the stored value. The traces are preprocessed and the SOST function is run (Figs. 7 and 11, Device 2). The same process is repeated with D3 and D4. Then, once we have the four POI selection graphs (Fig. 11, one for each device), we run the DTW algorithm to compute the warped distance between D1 and the rest (Fig. 11, D1 vs D2, D1 vs D3, D1 vs D4). Note that we also calculate other distances (D2 vs D3, D2 vs D4, D3 vs D4) to measure the similarities among the four devices. These results lead to the following conclusions: Devices D3 and D4 are the most similar ones since their distance is the smallest one. Thus, the distances between D1 & D3 and D1 & D4 are similar, and this fact validates the results of the attacks D1 vs D3 and D1 vs D4, which are also similar (Fig. 1). D2 is the most similar device to D1, but the attack is not successful. The reason is that the selected POIs are not optimal for this device, as shown below.

Improved POI Selection: To improve the effectiveness of the attack we suggest the following steps:

1. **Analyze the performance of each POI individually** to find which points correspond to good leaking points (when we port the model to another device, or even in the same device) and which ones are perturbing the model. In order to do that, we build templates using only one of the aforementioned POI each time [97, 158, 220, 294]. Then, we apply the templates to each device (D1, D2, D3 and D4). The results are presented in Table 2, where it can be seen that the POIs with better performance for D2 (the one with the worst results) are the second and the third POIs.
2. **Try different combinations of the POI with better performance** to guess which combination has best results. Theoretically, the optimal solution would be a combination of all of them, but in practice it is not always the case. To avoid trying all possible combinations, points with better performance in all cases (identified in previous step) must be selected. Also,

Table 2. *Improved POI selection* (when Device 1 (D1) is used for profiling.)

Step	POI	Rank (D1 vs D1)	Rank (D1 vs D2)	Rank (D1 vs D3)	Rank (D1 vs D4)	GS
1	1st POI [97]	**9**	121	121	121	38.32
	2nd POI [158]	**1**	70	170	161	35.19
	3rd POI [220]	**3**	98	53	53	24.87
	4th POI [294]	50	136	121	121	41.18
2	POI 1+3+4	**1**	140	13	14	27.01
	POI 2+3	**1**	45	72	65	17.44
	POI 1+2+3	**1**	36	49	42	**12.68**
	POI 1+2	**1**	56	116	110	25.50
	POI 3+4	**3**	148	37	38	31.73
3	[92, 153, 218]	**1**	**2**	**16**	**15**	2.48

the warped distance graphic (Fig. 9) can be used to identify the most similar parts of these signals and choose the POI combinations accordingly. In this case, we have tried the following combinations: 1+3+4, 2+3, 1+2+3, 1+2 and 3+4. Results can be seen in Table 2 (Step 2). To help identifying which combination works best in all devices we recommend to compute what we call *Goodness Score* (Table 2, column GS). This metric is computed by adding the ranks of each row weighted by a coefficient ($GS = Rank_{D_1} * C_{D_1} + Rank_{D_2} * C_{D_2} + Rank_{D_3} * C_{D_3} + Rank_{D_4} * C_{D_4}$). The value of the coefficients is obtained dividing the rank of the first attack in the Table 1, by the number of possible combinations (256), so we obtain coefficients from 0 to 1 with the emphasis put on the devices with worse result. The lower GS value, the better performance (generally speaking). Note that the combination with the best performance (in all devices) includes those three points of interest (1+2+3).

3. **Adjust the selected POI:** once we have found the best combination, we can try to tune each POI moving the selected point in a small number of samples (and adding another one near if necessary) and check if the results improve. Based on experience, points that are not exactly at the peak sometimes provide better results because the behaviour of the devices is more similar in those points. Summing up, we identified the most suitable zones to select POIs in previous step, and we are now tuning the exact value of these POIs. In our case, after a few trials we obtained very good results with [92, 153, 218] (Table 2, Step 3).

After applying the proposed techniques, we can conclude the following:

- **Not all POIs have the same performance with all devices:** For instance, in Table 2 we can see how the templates built with the 2nd POI have better performance attacking D2 than devices D3 or D4. However, with the 3rd and 4th POIs we obtain a better performance attacking devices D3

Table 3. Portable template attacks experiments using Device 3 (D3) for profiling.

Step	POI	Rank (D3 vs D3)	Rank (D3 vs D4)	Rank (D3 vs D1)	Rank (D3 vs D2)	GS
0	[97, 158, 220, 294]	**1**	**7**	114	130	
1	1st POI [97]	**15**	131	153	153	149.47
	2nd POI [158]	**5**	30	75	75	72.32
	3rd POI [220]	**16**	22	206	206	197.01
	4th POI [294]	**2**	117	135	135	131.88
2	POI 1+3+4	**1**	43	194	191	184.56
	POI 2+3	**1**	**8**	105	105	100.30
	POI 1+2+3	**1**	**9**	111	117	109.09
	POI 1+2	**2**	**22**	70	66	**65.30**
	POI 3+4	**3**	42	196	196	187.97
3	[90, 153, 156]	**8**	**1**	**16**	**8**	11.25

and D4. The same happens with combinations of those POIs: POI 1+2 has better performance for D2 than others, while profiling with POI 1+3+4 has good results for devices D3 and D4 but not for D2, etc.

– **DTW's wrapping distance can assess the similarity between devices.** After selecting proper POI, we can see how the results of the Template attacks and the *similarity assessment* technique are directly linked. In Fig. 11 we notice how devices D3 and D4 have the smallest distance between them, which means that they are very similar (Table 2 shows how the results of the attacks are almost the same for both). Another example is that if we compare the distance between D1 and the rest, the device with shortest distance is D2, and the performance of the portable attack is generally better with that device. Again, distances from devices D3 and D4 are similar, as well as the results of the portable attacks.

Extending the Problem to Other Device

With the aim of confirming the results shown above we repeat the same process, but using D3 for generating the templates (Results shown in Table 3). The conclusions obtained when profiling with D1 are now validated. Devices D3 and D4 are very similar and hence the attack using the 4 POIs located in the 4th spike of Device 3's SOST is effective for D4 but unsuccessful for D1 and D2. After analysing the performance of each POI and its combinations we obtain the same conclusion: the POIs that work for D4 (the most similar device) do not work properly for the rest of devices (D1 and D2). Only after adjusting the POI with the best performance in devices D1 and D2 (POI 1+2) and adding another point near the second POI, we can obtain good results in those devices, but at the cost of sacrificing some performance for devices D3 and D4. In conclusion, it is important to get a balance and find POIs that represent leakage in all devices.

In this case, we had to select POIs that are not optimal for attacking D4 but allowed us to successfully attack devices D1 and D2. Thus, the templates generated with the improved POI selection are suitable for attacking each one of those four devices.

Extending the Problem to Different Measurements of the Same Device With this experiment, we want to show that measurements taken at different times may have substantial differences caused by slight changes in the device or its environment. With D3, the previous process is repeated two times, obtaining two sets of 20 000 random traces and two sets of 1 000 fixed traces. After comparing both SOST graphs (Fig. 12) it can be noticed that, although both graphs are very similar there are slight differences between both traces. In order to check whether those differences affect the performance, we carry out two different attacks. In the first one, we build templates and implement the attack with traces from the first measurement set (D3) while in the second attack we use traces from the second measurement set (D3*) with the model from the first measurement. The attack is successful in both cases, but the correct candidate ranks as 1 in the first attack and as 4 in the second one, showing the worse performance in the later.

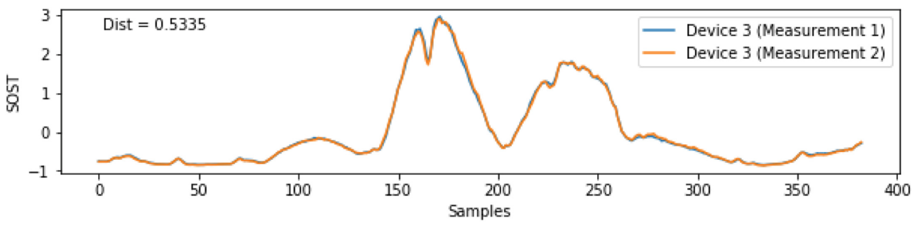


Fig. 12. SOST graph of the 1st and 2nd measurements (Device 3).

5.3 Use Case 2: Template Attack Using Two Devices in Profiling Phase

In order to improve the performance of the portable template attack, we try building templates with two different devices and attacking a third one. Thus, a model is built with 20 000 profiling traces from D1 and 20 000 profiling traces from D2. The traces are preprocessed as in previous experiments and the SOST function is calculated (Fig. 13, Device 1+2). This figure, when compared with the SOST of each device separately (Fig. 11), suggests that the new SOST is a combination of the other two. We select 7 POIs in each one of the significant spikes ([95, 142, 153, 173, 207, 218, 237]) and generate the 256 templates. In the *attacking phase* 1 000 power traces of devices D1 and D2 storing a fixed 8-bit value are used (500 from each one). Then the model is applied and the fixed value is guessed. The results of this non-portable attack using two devices for profiling are excellent and they can be seen in Table 4 (Step 0, [D1+D2] vs [D1+D2]). For the portable attack, the model is applied to 1 000 power traces from D3 storing the same fixed value. As it can be seen in Table 4 (Step 0), the results are quite bad because the model does not suit this device. The same happens with D4. As in the first use case, the *similarity assessment* method is applied. Thus, 20 000 power traces from the third and fourth device storing random 8-bit values are taken and labeled with the stored value. The traces are preprocessed and the SOST function is calculated (Fig. 13, D3 and D4). We can see that both shapes are completely different and hence the attack is not successful because we are not selecting representative POIs for all the devices. To improve the POI selection, we apply our technique, as shown in Sec. 5.2. First, we analyze the performance of each POI individually (Table 4, Step 0). Note that the results are almost the same in D3 and D4 because these devices are very similar. Then, different combinations of the 7 POIs are tested (Table 4, Step 2). The combination which throws best results is 2+3+6. After that, we try to adjust the POI of interest, achieving the attack using the points [143, 157, 217] (Table 4, Step 3).

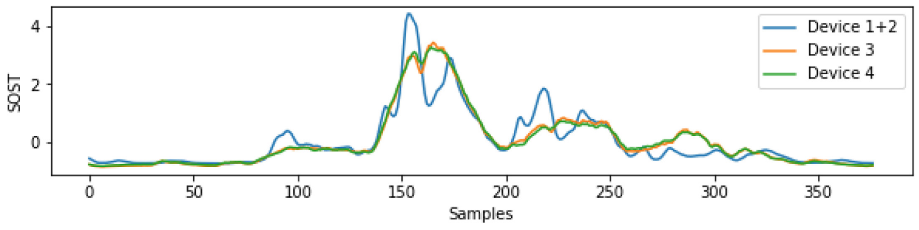


Fig. 13. SOST graph of devices 1+2, 3 and 4

Extending the Problem to Other Devices

In order to consolidate the results shown above we repeat the same process, using devices D1 and D3 for templates generation. First of all, the SOST graph of the profiling traceset (20 000 random traces form D1 and 20 000 random traces from D3) is computed (Fig. 14). In this case, the model is built with traces from two devices which have significant differences, as we have seen in previous experiments. Then, a multivariate model is built using 10 POI located in each one of the significant spikes of the SOST ([90, 104, 115, 155, 166, 178, 216, 234, 267, 296]). The results appear in Table 5 (Step 0). Note that the results are quite good without applying our improved POI selection because we have constructed a model with devices which are similar to the attacked devices (Devices D1 & D2 and D3 & D4 are very similar). Nevertheless, the results with D2 are not as good as expected, so we perform our improved POI selection. After analyzing the performance of each POI individually, we can notice that, in general, all points work better for one device than for the other. Thus, we try different

Table 4. Portable template attacks experiments using Device 1 (D1) and Device 2 (D2) for profiling.

Step	POI	Rank ([D1+D2] vs [D1+D2])	Rank ([D1+D2] vs D3)	Rank ([D1+D2] vs D4)	GS
0	[95, 142, 153, 173, 207, 218, 23]	**1**	42	41	
1	1° POI	**1**	114	114	36.96
	2° POI	**4**	61	61	19.79
	3° POI	**6**	56	43	16.10
	4° POI	**8**	74	74	24.02
	5° POI	92	201	201	65.53
	6° POI	**2**	58	58	18.81
	7° POI	**9**	56	56	18.19
2	POI36	**2**	**16**	**18**	5.52
	POI23467	**1**	**20**	**19**	6.33
	POI3467	**1**	26	**24**	8.11
	POI367	**1**	**22**	**21**	6.98
	POI67	**2**	54	55	17.68
	POI236	**1**	**15**	**14**	**4.71**
	POI346	**1**	**24**	**24**	7.79
3	[143, 157, 217]	**1**	**4**	**5**	1.46

POI combinations to find the optimal one. In this case, we try to select points that provide good results in both devices, avoiding points that work especially bad in a certain device. The combination which provides the best results is POI 1+3+4+5+6+7+9. Finally, after adjusting the selected POI, we obtain excellent results with [91, 115, 153, 168, 181, 216, 266] (Table 5, Step 3).

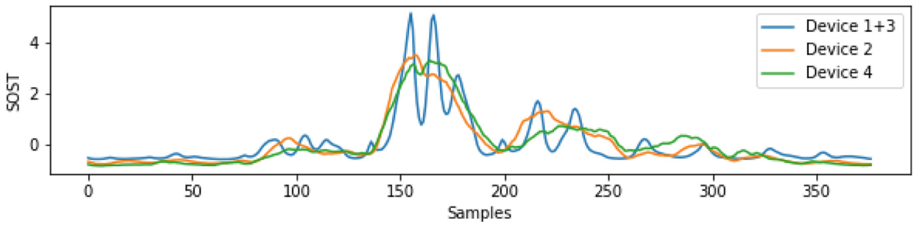


Fig. 14. SOST graph of devices 1+3, 2 and 4

Table 5. Portable template attacks experiments using Device 1 (D1) and Device 3 (D3) for profiling.

Step	POI	Rank ([D1+D3] vs [D1+D3])	Rank ([D1+D3] vs D2)	Rank ([D1+D3] vs D4)	GS
0	[90, 104, 115, 155, 166, 178, 216, 234, 267, 296]	**1**	**25**	**1**	
1	1° POI [90]	34	**18**	97	2.27
	2° POI [104]	**15**	119	61	11.92
	3° POI [115]	**16**	122	27	12.08
	4° POI [155]	**13**	49	**5**	4.86
	5° POI [166]	**6**	72	52	7.26
	6° POI [178]	**3**	88	37	8.75
	7° POI [216]	11	51	41	5.18
	8° POI [234]	**1**	199	**22**	19.52
	9° POI [267]	**24**	86	44	8.66
	10° POI [296]	**4**	126	77	12.62
2	POI 5+7+9	**2**	34	**12**	3.38
	POI 1+5+7+9	**1**	29	**12**	2.88
	POI 1+4+5+6+9	**1**	34	**20**	3.40
	POI 1+3+4+5+6+7	**1**	**25**	**1**	2.45
	POI 1+3+4+5+6+9	**2**	35	**4**	3.44
	POI 1+3+4+5+6+7+9	**1**	**19**	**1**	**1.86**
	POI 3+4+5+6+7+9	**1**	**21**	**1**	2.06
	POI 3+5+6+7+9	**1**	30	**9**	2.97
3	[91, 115, 153, 168, 181, 216, 266]	**1**	**2**	**1**	0.20

6 Conclusions

As mentioned above, the portability issue is usually underrated. Lots of related works obtain the profiling and attacking data sets from the same device instead of performing a profiling attack in the way it was conceived: generating a power model on an "identical" copy of the device to attack. In this work we have shown how performing this kind of attacks in a realistic setup is a complex task, since slight differences in the construction of the devices or in the acquisition of the traces cause different behaviours that usually ruin the attack. While some devices maximize leakage in a particular point of the power traces, the leakage of another "identical" copy of the device can be (slightly) shifted. Therefore, it is crucial to take into account these variations and to generate the models using the points where the leakage exist in all devices. In this paper we present a way to understand better how variations between devices occur, and we describe how to build models that allows finding and exploiting the common leakage. The experimental results show how our *similarity assessment* measurement (warped distance) is directly related to portability, since the portable template attacks have better performance in the most similar devices (devices with a smaller distance between them). The experimental results also confirm that our *improved POI selection* technique helps in finding the POIs which represent leakage in several devices and in avoiding the ones that perturb the model, making it not applicable to a particular device. Moreover, we have shown how building multivariate leakage models with several devices also enhances the performance of the attack, even more if the model is generated with devices with behavioural differences.

In conclusion, the proposed *similarity assessment* technique allows evaluation laboratories to identify behavioural differences between devices and quantify them, and improves the POI selection in template attacks, as shown in the different use cases presented.

References

1. Akkar, M.-L., Bevan, R., Dischamp, P., Moyart, D.: Power analysis, what is now possible. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 489–502. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_38
2. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_1
3. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., Shrivastwa, R.R.: Mind the portability: a warriors guide through realistic profiled side-channel analysis. In: Network and Distributed System Security Symposium, January 2020. https://doi.org/10.14722/ndss.2020.24390
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
5. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 45–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_3
6. Carbone, M., et al.: Deep learning to evaluate secure RSA implementations. Cryptology ePrint Archive, Report 2019/054 (2019)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
8. Choudary, M.O., Kuhn, M.G.: Efficient, portable template attacks. IEEE Trans. Inf. Forensics Secur. **13**(2), 490–501 (2018). https://doi.org/10.1109/TIFS.2017.2757440
9. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 253–270. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_17
10. Choudary, O., Kuhn, M.G.: Template attacks on different devices. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 179–198. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10175-0_13
11. Elaabid, M., Abdelazizand Guilley, S.: Portability of templates. J. Cryptogr. Eng. **2**(1), 63–74 (2012). https://doi.org/10.1007/s13389-012-0030-6

12. Fisher, R.: The statistical utilization of multiple measurements. Ann. Eugen. (Cambridge) **8**, 376–386 (1935). https://doi.org/10.1111/j.1469-1809.1938.tb02189.x
13. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_2
14. Gohr, A., Jacob, S., Schindler, W.: CHES 2018 side channel contest CTF - solution of the AES challenges. IACR Cryptology ePrint Archive (2019)
15. Heuser, A., Zohner, M.: Intelligent machine homicide. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 249–264. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29912-4_18
16. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. J. Cryptogr. Eng. **1**, 293–302 (2011). https://doi.org/10.1007/s13389-011-0023-x
17. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning: With Applications in R. JABES **19**, 556–557 (2014). https://doi.org/10.1007/s13253-014-0179-9
18. Johnson, R.A., Wichern, D.W. (eds.): Applied Multivariate Statistical Analysis. Prentice-Hall Inc., Upper Saddle River (1988)
19. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
20. Kim, K., Kim, T.H., Kim, T., Ryu, S.: AES wireless keyboard: Template attack for eavesdropping. In: Black Hat Asia, Singapore (2018)
21. Lerman, L., Bontempi, G., Markowitch, O.: Side channel attack: an approach based on machine learning. In: Constructive Side-Channel Analysis and Secure Design, COSADE (2011)
22. Lerman, L., Bontempi, G., Markowitch, O.: A machine learning approach against a masked AES. J. Cryptogr. Eng. **5**(2), 123–139 (2015). https://doi.org/10.1007/s13389-014-0089-3
23. Lerman, L., Poussier, R., Markowitch, O., Standaert, F.X.: Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. J. Cryptogr. Eng. **8**(4), 301–313 (2018). https://doi.org/10.1007/s13389-017-0162-9
24. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. SPACE **2016**, 3–26 (2016). https://doi.org/10.1007/978-3-319-49445-6_1
25. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Boston (2007). https://doi.org/10.1007/978-0-387-38162-6
26. Muijrers, R.A., van Woudenberg, J.G.J., Batina, L.: RAM: rapid alignment method. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 266–282. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-27257-8_17
27. Picek, S., Samiotis, I.P., Kim, J., Heuser, A., Bhasin, S., Legay, A.: On the performance of convolutional neural networks for side-channel analysis. In: Chattopadhyay, A., Rebeiro, C., Yarom, Y. (eds.) SPACE 2018. LNCS, vol. 11348, pp. 157–176. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05072-6_10
28. Rechberger, C., Oswald, E.: Practical template attacks. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31815-6_35

29. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 109–128. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_8
30. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. **26**(1), 43–49 (1978). https://doi.org/10.1109/TASSP.1978.1163055
31. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**, 561–580 (2007). https://doi.org/10.3233/IDA-2007-11508
32. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
33. Schneider, T., Moradi, A., Standaert, F.-X., Güneysu, T.: Bridging the gap: advanced tools for side-channel leakage estimation beyond Gaussian templates and histograms. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 58–78. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_4
34. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_26
35. Woudenberg, J., Witteman, M., Bakker, B.: Improving differential power analysis by elastic alignment. CT-RSA **6558**, 104–119 (2011). https://doi.org/10.1007/978-3-642-19074-2_8