



On the power of template attacks in highly multivariate context

Maamar Ouladj¹ · Nadia El Mrabet² · Sylvain Guilley^{3,4,5} · Philippe Guillot¹ · Gilles Millérioux⁶

Received: 8 March 2019 / Accepted: 18 August 2020 / Published online: 30 August 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract

When implemented in software (or hardware), a cryptographic protocol can leak sensitive information during its execution. Side-channel attacks can use those leakages in order to reveal some information about the secret used by the algorithm. The leaking side-channel information can take place in many time samples. Measurement appliances can cope with the acquisition of multiple samples. From an adversarial point of view, it is therefore beneficial to attempt to make the most of highly multivariate traces. On the one hand, template attacks have been introduced to deal with multivariate leakages, with as few assumptions as possible on the leakage model. On the other hand, many works have underlined the need for dimensionality reduction. In this paper, we clarify the relationship between template attacks in full space and in linear subspaces, in terms of success rate. In particular, we exhibit a clear mathematical expression for template attacks, which enables an efficient computation even on large dimensions such as several hundred of samples. It is noteworthy that both of PoI-based and PCA-based template attacks can straightforwardly benefit from our approach. Furthermore, we extend the approach to the masking-based protected implementations. Our approach is validated both by simulated and real-world traces.

Keywords Multivariate traces · Template attacks · Dimensionality reduction · Masking · Success rate · Signal-to-noise ratio · Spectral approach for computational speed improvement

1 Introduction

1.1 Context: the side channel threat

Side-channel traces collected from software code are extremely rich, since a same variable can leak at different places. Typically, leakage can spread over several samples within one clock cycle, and in addition, software implementations typically move variables in several registers or memory locations, causing leakage at many clock cycles.

1.2 Problem: making the most of high dimensionality

Modern oscilloscopes sample their input at a very high frequency; hence, it is possible to get more than one leakage sample per leaking sensitive variable. How to exploit such abundance of leakage measurements? Few non-supervised side-channel distinguishers manage such situation. Indeed, the samples usually leak differently; therefore, it is complex without prior knowledge to know how to best combine them constructively.

✉ Maamar Ouladj
maamar.ouladj@etud.univ-paris8.fr

¹ LAGA, UMR 7539, CNRS, Université de Paris VIII,
2 Rue de la liberté, Saint Denis 93200, France

² CEA-Tech, Département SAS, Centre CMP,
Mines de Saint-Etienne, Gardanne 13541, France

³ TELECOM-ParisTech, COMELEC department, SSH group,
Paris Cedex 13, France

⁴ Secure-IC S.A.S., Tour Montparnasse, 75015 Paris, France

⁵ École Normale Supérieure, Département d'Informatique,
CNRS, PSL University, 45 Rue d'Ulm, Paris 75005, France

⁶ CRAN, UMR 7039, ESSTIN, Université de Lorraine,
2 rue Jean Lamour, Vandœuvre-lès-Nancy 54506, France

1.3 State-of-the-art

Clearly, in the past, some strategies have been put forward. For instance, Clavier et al. [11, §3] suggested (albeit in another context, namely that of so-called shuffling countermeasures) to take as monovariate signal the average of the signal over the D samples. As another example, Hajra and Mukhopadhyay [24] investigate non-profiling side-channel analysis. However, for their analysis to work, some a priori structure is injected in the model, which is learned online on the attacking traces. However, in reality, the information con-

tents of the whole trace are larger than that soaked from the projection on one single support vector, but it depends on the multivariate trace distribution. Notice that template attacks (which are profiled attacks) achieve this goal. Template attacks on multivariate data have been described accurately as a “process” in state-of-the-art papers [30, §5.3.3]. However, those attacks are still perceived as a recipe (see Sec. 2.1 of [9]), so there is no (except from experimental cases) way to study which parameters impact the success of a template attack. Therefore, some folklore surrounds them. In particular, because the recipe is not formalized, some papers have tried to clarify the different steps and assumptions, in particular Choudary and Kuhn [10] wrote on making many details about template attacks more explicit, such as the notion of “pooled covariance matrix.” The inversion of the traces covariance matrix Σ is feared, and for this reason, a first pass of dimensionality reduction has been suggested right away in the seminal paper by [9]; the authors selected a few tens of so-called points of interest (PoIs), resulting in a smaller Σ which is easy to inverse. Still, we notice a theoretical contradiction, because the *data processing inequality* states that application of any function on the data reduces their informative contents. Subsequently, many researches have been carried out to improve on the heuristic method to identify PoIs in the traces. Archambeau et al. motivated the usefulness of principal component analysis (PCA) in [1] in this respect. Bär et al. improved on this PCA in [2] by actually hand-picking PoIs within PCA vectors. Elaabid et al. [18] account for a method to choose PoIs without the need of manual selection. Their method is as follows: a PCA is computed on all the traces, resulting in one “PCA trace”, and the PoIs are those samples such that the PCA value is larger than a user-defined threshold. The authors observed that this selection does retain most informative samples while discarding noisy samples. Fan et al. suggest to select PoIs as those where the noise is the most Gaussian [20]. Still, even deciding which selection of PoIs is optimal is questioned [45]. Zhang et al. noticed recently that there is still margin for improvement in PoIs selection algorithms [43]. So, to summarize, many papers have focused on reducing the traces dimensionality while retaining in the constructed subset most of the information, with a view to optimize the success rate. A secondary objective is to keep an acceptable computational load for the template attack. Indeed, template attacks, as presented originally, include in the attack phase the evaluation of computationally challenging functions, such as: exponential, matrix inversion, and square roots (see Eqn. (1) in Sec. 2.1 of [9]).

Eventually, we notice that many papers study trace denoising, using typically wavelets [16], independent component analysis (ICA) [29], etc. In our work, we exploit traces “raw,” so as to highlight the sole impact of multivariate analysis on attack efficiency. But, it is noteworthy that both of PoI-based and PCA-based template attacks [20] can also straightfor-

wardly benefit from our approach, both in terms of processing time and in terms of the needed space memory.

1.4 Contributions

In this paper, we analyze template attacks from a theoretical perspective, and derive the mathematical expressions for template building and attack phases. In the context of multivariate normal noise, the two phases are easily simplified (Algorithms 1 and 2) as mere linear operations. In particular, there is no need for exponential and square roots, and only one matrix inversion is required at the end of the template building phase. These formal expressions allow us to drastically improve on the clarity of the actual computations carried out by template attacks and as a result, the formal expression also improves on the computational aspects related to template attacks.

Our noteworthy contributions are detailed below:

- we factor code between training and matching algorithms,
- we optimize further the computation and needed memory space by grouping traces by classes, resulting in a computation on a so-called coalesced data,
- we have the training phase delivers only one matrix, thereby saving repeated computations in the attack phase (see Algorithms 3 and 4).
- As a consequence, we manage to perform template attacks on highly multivariate traces. Contrary to belief, we show that the more samples are taken (i.e., no dimensionality reduction is needed), and the more successful is the template attack. But our approach is also compatible with reduced dimensionality data.
- Said differently, we are able to analyze full-width traces without any preprocessing (which destroys information) and show that this is the best attack strategy, in terms of number of traces, to recover the key.
- in the sequel, we extend our approach to the masking-based protected implementations.
- Finally, we highlight a spectral approach which allows for a near exponential computational improvement in the attack phase (an attack factor in 2^n is reduced to n , where n is the bitwidth of the sensitive variable).

Furthermore, those findings allow us to observe practically the effect of the dimensionality (the number of samples in a trace) on the success rate. Namely, we show that the success rate increases when traces of higher dimensionality are used. We also show that template attacks (without initial dimensionality reduction using PCA, for instance) are more efficient than monivariate attacks (such as the correlation power analysis, also known as CPA [5]) applied on the first direction of the PCA leakage reduction. Such practical

derivations are, to the best of our knowledge, new, and we here provide some numerical values about the actual gain conveyed by attacks of increased dimensionality.

1.5 Outline

In order to present our contributions, we follow the scientific approach described hereafter. In Sect. 2, we present a mathematical modelization of the problem. In Sect. 3, we provide our first contribution, namely the formalization of template attacks. In Sect. 4, efficient algorithms to compute template attacks are given. In Sect. 5, we validate our contributions on real traces taken from an AES running on an ATMega 163 smart card. In particular, we exhibit a spectral approach to speed up the computation of template attacks. Finally, in Sect. 6, we conclude our study.

2 Mathematical modelization of the problem and notations

2.1 Side-channel problem

We model the side-channel problem as follows:

$$T, k \rightarrow Z \rightarrow Y(Z) = Y \rightarrow X = Y + N. \quad (1)$$

In this equation, we have that:

- T is the digital part known by the attacker, typically some text (either plaintext or ciphertext),
- k is the digital part unknown by the attacker, typically some part of key, which is fixed and will be guessed,
- Z is an intermediate value within the targeted cryptographic algorithm, e.g., $Z = \text{SubBytes}(T \oplus k)$ in AES; formally, Z is the sensitive variable which consists in a vectorial Boolean function of T and k ,
- Y is the leakage corresponding to Z — the link between Y and Z is deterministic,
- X is the side-channel leakage measured by the attacker, which consists in Y plus some independent noise N ,
- N is the noise.

We assume all those variables are measured many times (Q times). Therefore, the random variables have a dimensionality Q , with the particularity that k is the same for all Q , and that noise random variables N_q are all i.i.d.

In addition, we assume that the measurements are multi-dimensional of dimensionality D . This can represent the fact that:

- oscilloscopes capture windows of D samples, typically at high sampling rates, resulting in traces of many samples per clock period,
- simultaneous captures by various probes (e.g., power and electromagnetic, or also multiple electromagnetic probes placed at various locations).

2.2 Additional notations

We adopt the following notations:

- S : the cardinality of the space where Z belongs to, that is if $Z \in \{0, 1\}^n$, then $S = 2^n$; to ease notations, we also assume (without loss of generality though) that the number of T values is S . For instance, Z typically arises from a bijective substitution box applied on $T \oplus k$.
- Σ : the $D \times D$ covariance matrix of the noise, such that $N_q \sim \mathcal{N}(0, \Sigma)$ ($0 \leq q < Q$).

Moreover, we will use the reordering trick put forward in [28], which consists in trading \sum_q for $\sum_t \sum_{q:t_q=t}$, where $\sum_{q:t_q=t}$ is a shortcut notation for $\sum_{\substack{cq \in \{1, \dots, Q\}, \\ \text{such that: } t_q=t}}$.

Therefore, we have the following “types” for variables defined in Sect. 2.1:

- T and k are bitvectors (of second dimension Q), not necessarily of the same length,
- Z is a matrix of Booleans $\{0, 1\}^n \times Q$,
- Y and X are matrices of $D \times Q$ real numbers.

The peculiarity of Y is that we have $Y(Z_q) = Y(Z_{q'})$ if $T_q = T_{q'}$ (under the same key k).

We introduce the notion of *coalesced matrices*. This notion has been introduced in [28, §3.3], but was not named there. Therefore, we qualify this notion (averaging traces corresponding to the same sensitive variable) as “coalescence.” It arises from the fact that Y and X , random variables in (1), depend only on values from Z (except from X which also depends on the noise, which is independent from other random variables), and therefore, their matrices can be coalesced from size $D \times Q$ to $D \times S$.

Definition 1 (*Coalesced matrices*) Coalesced model $\tilde{Y}(k)$ is the $D \times S$ matrix

$$\left(\tilde{Y}_{t=0,k}, \dots, \tilde{Y}_{t=S-1,k} \right),$$

i.e., where t is enumerated in lexicographical order. This matrix is a property of the leakage from the device under test, i.e., it is not a random variable, but a constant value. For any measurement q , we denote $Y_q = \tilde{Y}_{t_q,k}$.

Coalesced measurement \tilde{X} is the matrix

$$(\tilde{X}_0, \dots, \tilde{X}_{S-1}) = \left(\frac{\sum_{q:t_q=0} X_q}{\sum_{q:t_q=0} 1}, \dots, \frac{\sum_{q:t_q=S-1} X_q}{\sum_{q:t_q=S-1} 1} \right).$$

In this equation, we denote by n_t the number of plaintexts such that t_q equals to t . If for some t the number n_t it is equal to zero, then by convention $\tilde{X}_t = 0$. Alternatively, the empty classes can be pruned. Both choices are equivalent.

The advantage of Definition 1 is twofold. First, as long as $Q \geq \#T = 2^n$, the matrix X (of size $D \times Q$) is reduced to a matrix \tilde{X} (of size $D \times 2^n$). Second, when n_t is not equal to zero, \tilde{X}_t as defined here is the empirical average of $\mathbb{E}(X|T = t)$ over n_t values. Thus, for any t and q , X_q (for $t_q = t$) and \tilde{X}_t have same expectation, but \tilde{X}_t has an empirical standard deviation divided by $\sqrt{n_t}$ relatively to that of X_q .

It shall also be noted that in our model, the $D \times Q$ matrix Y (and the $D \times S$ coalesced matrix \tilde{Y}) are general. In particular, we do not assume any structure, such as \tilde{Y} being the product of a $D \times 1$ column by a $1 \times S$ row, as done in [6,24].

Examples of Z functions are given hereafter:

- in the software case, $T, k \in \{0, 1\}^n$, and $Z = T \oplus k$;
- in the hardware case, for the case of AES ($n = 8$) attacked on the last round:
 - $T, k \in \{0, 1\}^n$, and $Z = (\text{InvSubBytes}(T \oplus k), T)$ for the bytes of the first row, and
 - $T = (T_1, T_2) \in \{0, 1\}^n \times \{0, 1\}^n$, and $Z = (\text{InvSubBytes}(T_1 \oplus k), T_2)$ for the other three rows.

Those two cases refer to the two models expressed in [14]:

- Only manipulated Data Leak (ODL): only the manipulated value influences the leakage.
- Memory Transitions Leak (MTL): two values (the previous one and the new one) of a memory unit (e.g., a register) influence the power consumption and also the device leaks some combination of the two consecutively manipulated values.

We also introduce other useful notations for matrices:

- Let A be a square matrix, then its trace is the sum of elements along its diagonal $\text{tr}(A) = \sum_i A_{i,i}$; let A and B two rectangular matrices such that AB and BA are square. Then, the trace has the property that $\text{tr}(AB) = \text{tr}(BA)$.
- Covariance matrices are symmetrical, and all their eigenvalues are positive. When some eigenvalues are zero, it means that rows are redundant, and those are implicitly removed until all eigenvalues are strictly positive. Alternatively, more measurements shall be captured. Therefore, we will consider covariance matrices are invertible. For example, the noise covariance matrix Σ

has an inverse, denoted Σ^{-1} . We will also make use of the notation $\Sigma^{-1/2}$ for the (unique¹) matrix such that $\Sigma^{-1/2} \Sigma^{-1/2} = \Sigma^{-1}$.

Eventually, let us define the following operator:

Definition 2 (The k th trace operator tr_k) Let $k \in \mathbb{F}_2^n$. The k th trace operator tr_k of a $2^n \times 2^n$ square matrix M is

$$\text{tr}_k(M) = \sum_t M_{t, t \oplus k}.$$

The regular trace of a square matrix M is simply $\text{tr} = \text{tr}_0$.

2.3 Characterization of traces

The traces can be characterized according to their *signal-to-noise ratio* (SNR) as defined in [30, § 4.3.2, page 73]. Referring to Eqn. (1), the *monovariate* SNR is defined as the SNR at each sample of the trace. Specifically, the monovariate SNR trace is defined as:

Definition 3 (Monovariate SNR trace)

$$\text{SNR} = \frac{\text{VAR}(\mathbb{E}(X|T))}{\mathbb{E}(\text{VAR}(X|T))}, \tag{2}$$

where \mathbb{E} denotes the expectation and VAR the variance operators.

We notice that this notion of SNR is useful to predict the approximated number of traces $Q_{80\%}$ to recover the key, with 80% confidence. Indeed, $Q_{80\%}$ is proportional to the inverse of SNR, as demonstrated in [4].

In some cases, such as in software implementations, the monovariate leakage can feature a strong SNR. However, this gives no intuition regarding the proportion of signal which is informative within the trace. Typically, could the SNR be improved more by reducing further the noise? The normalized inter-class variance (NICV [3]) allows to answer this question. The NICV is the proportion of the traces variance which can be explained by:

Definition 4 (Monovariate NICV trace)

$$\text{NICV} = \frac{\text{VAR}(\mathbb{E}(X|T))}{\text{VAR}(X)}. \tag{3}$$

¹ Recall that Σ is a symmetric matrix. Therefore, there exists one invertible matrix P such that $\Sigma = PDP^{-1}$, where D is a diagonal matrix whose diagonal coefficients are all positive. It is customary to call the diagonal coefficients of D the eigenvalues of Σ and P the matrix of eigen-vectors of Σ . Let us denote $D^{1/2}$ the diagonal matrix where diagonal coefficients are the square root of those of D . Then, $\Sigma^{1/2} := PD^{1/2}P^{-1}$ matches the definition, since $\Sigma^{1/2}\Sigma^{1/2} = PD^{1/2}P^{-1}PD^{1/2}P^{-1} = PD^{1/2}D^{1/2}P^{-1} = PDP^{-1} = \Sigma$.

Owing to the law of total variance, that is

$$\text{VAR}(X) = \text{VAR}(\mathbb{E}(X|T)) + \mathbb{E}(\text{VAR}(X|T)),$$

it can be seen that NICV is bounded between 0 and 1, and that $\text{NICV} = 1/(1 + 1/\text{SNR})$. This implies that

$$\frac{\partial \text{NICV}}{\partial \text{SNR}} = \frac{1}{(1 + \text{SNR})^2} > 0,$$

i.e., NICV is an increasing function of SNR. The attack success rate is thus improved by increasing either the SNR or the NICV.

3 Formalization of template attacks

In this section, we present our main result. We formalize the two phases of the template attack, namely learning and attacking. Their algorithms reveal a very simple mathematical expression for building the template model and for matching never seen traces (expression which happens to be very similar). In particular, this leads us to simplify the computation for the two phases: the mathematical expressions involve only linear algebra (neither logarithms nor exponential functions need to be evaluated); only one matrix inversion is required (namely once in the end of the profiling stage). Moreover, we show that the Q traces used either for profiling or matching can be regrouped in $S = 2^n$ classes. In the sequel, we use 2^n (e.g., $2^n = 256$ for the AES) in lieu of S for the paper to be concrete. The attack two phases can in turn be computed using 2^n -dimensional vectors and matrices of dimension $2^n \times 2^n$. We will say that the number of traces Q is coalesced into 2^n classes, which further simplifies the computations.

We start by analyzing the classical template attacks (without coalescence) in Sect. 3.1. We then see the gain of considering template attacks with coalesced traces and models in Sect. 3.2. Eventually, we compare results with attacks making use of dimensionality reduction in Sect. 3.3.

3.1 Template attack (without coalescence)

3.1.1 Template building = profiling stage

In the profiling stage, the attacker computes \tilde{Y} , a $D \times 2^n$ matrix, and Σ . An experimental method to profile is the following:

- In the estimation of \tilde{Y} , the attacker fixes Z to some all possible values (the column index of \tilde{Y}), and averages many measurements, so as cancel out the noise; in order to explore all the possible values of Z , the attacker can

enumerate all values of T and k . However, due to symmetries, it might be possible that some values of Z be encountered by many different pairs (T, k) ; hence, there is an opportunity to save some profiling effort.

- Regarding the estimation of Σ , the attacker fixes Z to an arbitrary value, say 0, and then estimates Σ as the covariance matrix of the traces.

Therefore, in the sequel, we consider that the averages of the traces (\tilde{Y}) and the noise covariance matrix (Σ) are known. Notice that $N \sim \mathcal{N}(0, \Sigma)$ has zero mean, because the data (data = text and key) dependent part of the traces constitutes the expectation of the noise.

Also notice that we implicitly opted for a leakage decomposition basis known as the *canonical basis* [21]. However, any other choice of bases suits perfectly, insofar as the change to another basis is a multiplication by an invertible $2^n \times 2^n$ matrix, which can be applied on coalesced matrices (the noise is not impacted by the basis change, which concerns only the data). Such basis change could be interesting if the leakage happens to be concentrated on a smaller basis than the canonical one: The uninteresting dimensions can thus safely be dropped, thereby simplifying the equations.

3.1.2 Template attack

After the *identification* stage (profiling, as described in Sect. 3.1.1), the attacker can perform the *exploitation* stage. We have the following result:

Theorem 1 (Theorem 1 of [7]) *Template attacks guess the key as:*

$$\hat{k} = \operatorname{argmin}_k \operatorname{tr}((X - Y_k)^\top \Sigma^{-1} (X - Y_k)). \tag{4}$$

3.1.3 Standardization of traces

Theorem 2 (Template attack with standardized traces) *It is possible to standardize the templates (and the traces), by trading:*

- Y_k for $Y'_k = \Sigma^{-1/2} Y_k$, and
- X for $X' = \Sigma^{-1/2} X$.

Accordingly, the template attack simplifies as:

$$\hat{k} = \operatorname{argmin}_k \operatorname{tr}((X' - Y'_k)^\top (X' - Y'_k)) = \operatorname{argmin}_k \|X' - Y'_k\|_F^2, \tag{5}$$

where $\|\cdot\|_F^2$ is the square Frobenius norm of a matrix, that is the sum of all its elements raised to the power two.

Proof Notice that owing to the symmetry of Σ , one also has the following property $(\Sigma^{-1/2})^T = \Sigma^{-1/2}$. Hence,

$$\begin{aligned} (X - Y_k)^T \Sigma^{-1} (X - Y_k) &= (X - Y_k)^T \Sigma^{-1/2} \Sigma^{-1/2} (X - Y_k) \\ &= (\Sigma^{-1/2} (X - Y_k))^T \Sigma^{-1/2} (X - Y_k) \\ &= (X' - Y'_k)^T (X' - Y'_k). \end{aligned}$$

Besides,

$$\begin{aligned} &\text{tr}((X' - Y'_k)^T (X' - Y'_k)) \\ &= \sum_q (X'_q - Y'_{q,k})^T (X'_q - Y'_{q,k}) \\ &= \sum_q \|X'_q - Y'_{q,k}\|_2^2 \quad (\text{norm-2 of a } D\text{-dimensional vector}) \\ &= \|X' - Y'_k\|_F^2 \quad (\text{Frobenius norm of a } D \times Q \text{ matrix}). \end{aligned}$$

□

Notice that the standardized noise $N' = \Sigma^{-1/2}N$ has distribution $\mathcal{N}(0, I)$, where I is the $D \times D$ identity matrix.

Remark 1 In the expression of the template attack of Eq. (5), the covariance matrix Σ disappears: It is hidden half in the model (Y' is $\Sigma^{-1/2}Y$) and half in the matching traces (X' is $\Sigma^{-1/2}X$). Alternatively, we will show in Algorithm 3 that Σ can be completely hidden in the templates, and that there is no need to use Σ in the corresponding matching phase (Algorithm 4). This way of using Σ allows to minimize the computation time, in particular because Σ will need to be inverted only once, namely when building the model.

3.2 Template attack (with coalescence)

The trace operator in Theorem 1 is applied to a $Q \times Q$ matrix (resulted from the product of the raw matrices, without coalescence), which poses a problem of scaling as the number of traces grows. Therefore, we show that template attacks can be rewritten in a coalesced form:

Proposition 1 (Coalesced template attack) *Template attacks (as per (4)) guess the key as:*

$$\hat{k} = \underset{k}{\text{argmin}} \sum_t n_t (\tilde{X}_t - \tilde{Y}_{t,k})^T \Sigma^{-1} (\tilde{X}_t - \tilde{Y}_{t,k}), \quad (6)$$

where $n_t = \sum_{q:tq=t} 1$ is the number of traces corresponding to plaintext value t .

Proof By developing the argument to minimize in Theorem 1, there remain only two terms which depend on the key k :

$$\sum_q Y_{tq,k}^T \Sigma^{-1} X_q = X_q^T \Sigma^{-1} Y_{tq,k} \quad (7)$$

and

$$\sum_q Y_{tq,k}^T \Sigma^{-1} Y_{tq,k}, \quad (8)$$

because $\sum_q X_q^T \Sigma^{-1} X_q$ is independent from the key k .

Now,

$$\begin{aligned} \sum_q Y_{tq,k}^T \Sigma^{-1} X_q &= \sum_t \sum_{q:tq=t} Y_{tq,k}^T \Sigma^{-1} X_q \\ &= \sum_t \left(\sum_{q:tq=t} Y_{tq,k}^T \Sigma^{-1} X_q \right) \\ &= \sum_t \left(\sum_{q:tq=t} \tilde{Y}_{t,k}^T \Sigma^{-1} X_q \right) \\ &= \sum_t \tilde{Y}_{t,k}^T \Sigma^{-1} \left(\sum_{q:tq=t} X_q \right) \\ &= \sum_t n_t \tilde{Y}_{t,k}^T \Sigma^{-1} \left(\frac{\sum_{q:tq=t} X_q}{n_t} \right) \\ &= \sum_t n_t \tilde{Y}_{t,k}^T \Sigma^{-1} \tilde{X}_t \end{aligned}$$

and

$$\begin{aligned} \sum_q Y_{tq,k}^T \Sigma^{-1} Y_{tq,k} &= \sum_t \sum_{q:tq=t} Y_{tq,k}^T \Sigma^{-1} Y_{tq,k} \\ &= \sum_t \sum_{q:tq=t} \tilde{Y}_{t,k}^T \Sigma^{-1} \tilde{Y}_{t,k} \\ &= \sum_t \tilde{Y}_{t,k}^T \Sigma^{-1} \tilde{Y}_{t,k} \left(\sum_{q:tq=t} 1 \right) \\ &= \sum_t n_t \tilde{Y}_{t,k}^T \Sigma^{-1} \tilde{Y}_{t,k}. \end{aligned}$$

All in one, up to a key-independent term $\sum_t n_t \tilde{X}_t^T \Sigma^{-1} \tilde{X}_t$, we can rewrite the argument to minimize in Theorem 1 as:

$$\sum_t n_t (\tilde{X}_t - \tilde{Y}_{t,k})^T \Sigma^{-1} (\tilde{X}_t - \tilde{Y}_{t,k}).$$

□

Remark 2 Equation (6) reads as a trace on the plaintext space, weighted by the values n_t . Regarding n_t , they are the empirical estimators of $Q\mathbb{P}(T = t)$.

Remark 3 The attack presented in Proposition 1 is exactly the same as that of Theorem 1: It will succeed with the same number of traces. However, the attack in Proposition 1 is computationally more efficient than that of Theorem 1 as

soon as the number of traces Q is larger than the number of plaintexts involved in leakage model 2^n (for $2^n = 256$ for AES). The gain holds both for training and matching phases. However, we underline that training requires much more traces than matching; hence, most of the gain of using coalesced data arises from the template building phase.

3.3 State-of-the-art dimensionality reduction

It has been suggested in [1] that template attacks can be made more practical by:

1. first start with a dimensionality reduction,
2. then perform a template attack in a reduced space.

This approach deserves to be confronted with the innate dimensionality reduction power of template attacks, as expressed in Sect. 3.2 (where the complexity is related to cryptographic parameter $S = 2^n$ and not to the traces dimensionality D).

To be more accurate, two dimensionality reduction techniques have been put forward: principal components analysis (PCA [1]) and linear discriminant analysis (LDA [39]).

In this section, we contrast template attacks with and without such pre-processing (especially the PCA).

3.3.1 PCA

The PCA is a technique aiming at identifying linear projections which concentrate information in few directions, namely principal directions. In this respect, it is already pointed out in [1] that any set of training traces of high dimensionality D can always be reduced to lower dimensionality 2^n , as we also noted in Sect. 3.2.

The PCA appears in two variants: *classical* and *class-based*, as tailored by Archambeau et al. in [1]. Both techniques require the definition of a *covariance matrix*. We analyze in the following lemma both techniques, which resort to the concept of *coalesced matrices* (Definition 1):

Definition 5 (*Variance of a vectorial random variable*) The variance of $X \in \mathbb{R}^D$ is: $\text{VAR}(X) = \mathbb{E}((X - \mathbb{E}X)(X - \mathbb{E}X)^\top)$.

Lemma 1 (Law of total variance)

$$\text{VAR}(X) = \mathbb{E}(\text{VAR}(X|T)) + \text{VAR}(\mathbb{E}(X|T)).$$

Proof Adaptation of the law of total variance to the multivariate case. □

Applied to $X = Y + N$, where Y depends on T (but N does not), we have:

$$\text{VAR}(X) = \text{VAR}(Y) + \Sigma.$$

Besides, as $Y|T$ is deterministic, we can apply Lemma 1:

$$\text{VAR}(Y) = \text{VAR}(\mathbb{E}(Y|T)).$$

Let us assume Y is centered, namely $\mathbb{E}(Y) = 0$. Hence,

$$\text{VAR}(Y) = \sum_t \mathbb{P}(T = t) \mathbb{E}(Y|T) \mathbb{E}(Y|T)^\top.$$

Lemma 2 (PCA) *We resort to the law of large numbers (LLN). The classical PCA is based on the estimation of the following $D \times D$ covariance matrix:*

$$\frac{1}{Q} \sum_{q=1}^Q \left(X_q - \frac{1}{Q} \sum_{q'} X_{q'} \right) \left(X_q - \frac{1}{Q} \sum_{q'} X_{q'} \right)^\top \xrightarrow[Q \rightarrow +\infty]{LLN} \text{VAR}(Y) + \Sigma. \tag{9}$$

The PCA of Archambeau et al. is based on the estimation of the following $D \times D$ covariance matrix:

$$\frac{1}{Q} \sum_{q=1}^Q \left(\tilde{X}_q - \frac{1}{Q} \sum_{q'} \tilde{X}_{q'} \right) \left(\tilde{X}_q - \frac{1}{Q} \sum_{q'} \tilde{X}_{q'} \right)^\top \xrightarrow[Q \rightarrow +\infty]{LLN} \text{VAR}(Y). \tag{10}$$

Regarding the PCA, we only consider the second case. As the traces are first averaged, the covariance arising from the noise disappears; therefore, the only contribution is the intra-class variability. As the matrix is a covariance matrix, it has only positive eigenvalues. We consider the set of corresponding eigen-vectors as a matrix V (matrix of eigen-vectors), which is such that:

$$\tilde{Y}_k (\tilde{Y}_k)^\top V = V \Delta,$$

where the matrix $\Delta = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{2^n})$ is the diagonal matrix of eigenvalues. Besides, it is known [25] that $V V^\top = I$, the $D \times D$ identity matrix.

Lemma 3 *Template attack with PCA is always less efficient (in terms of success probability) than the template attack.*

Proof The reason is simply because the maximum likelihood is the best attack in terms of success probability. As the PCA is a preprocessing, it is less efficient. □

Notice that in practice, it is hard to estimate models and noise accurately; therefore, reducing the dimensionality before attacking is a winning strategy. However, assuming that a perfect profiling is possible, any preprocessing reduces

the information (recall the “data processing theorem”); therefore, one may wonder how different efficiencies of template attacks *with* and *without* PCA are.

In the case $D < 2^n$, $\tilde{Y}_k(\tilde{Y}_k)^T$ is almost of full rank (with very high probability). However, as traces are centered, there is one relationship amongst all rows of \tilde{Y}_k , namely their sum is null. Hence, V is not invertible. There is actually no dimensionality reduction: The projection on any eigenvector carries information.

The PCA therefore consists in transforming the traces the following way:

- \tilde{Y} becomes $V^T \tilde{Y}$ and X becomes $V^T X$ (projection),
- the covariance matrix Σ becomes $V^T \Sigma V$. Indeed, the covariance matrix of the projected (centered) noise is $\mathbb{E}((V^T N)(V^T N)^T) = \mathbb{E}(V^T N N^T V) = V^T \mathbb{E}(N N^T) V = V^T \Sigma V$.

The argument to maximize over k in the optimal distinguisher (4) becomes:

$$\begin{aligned} & \text{tr} \left(\left((V^T X - V^T Y_k)^T (V^T \Sigma V)^{-1} (V^T X - V^T Y_k) \right) \right) = \\ & \text{tr} \left(\left((X - Y_k)^T V (V^T \Sigma V)^{-1} V^T (X - Y_k) \right) \right) = \\ & \text{tr} \left(\left((X - Y_k)^T (V (V^T \Sigma V)^{-1} V^T) (X - Y_k) \right) \right). \quad (11) \end{aligned}$$

Assume that V is invertible. Using the property that $(AB)^{-1} = B^{-1}A^{-1}$, one has:

$$V (V^T \Sigma V)^{-1} V^T = V V^{-1} \Sigma^{-1} (V^T)^{-1} V^T = \Sigma^{-1},$$

which is the same covariance matrix as for the optimal distinguisher (matrix V disappears).

But V is not invertible, and therefore, Eq. (11) cannot be simplified.

Finally, it is noticeable that all the improvements introduced in the current paper are compatible with any dimensionality reduction technique.

4 Efficiently computing templates with coalescence

4.1 Simplification by the LLN

As will be shown, when a high number of traces are needed (where SNR is less), an efficient computation of an LLN-based approximation of the optimal template attack can be carried out. Hence, when using the LLN the number of traces to recover the key is generally more than the exact template

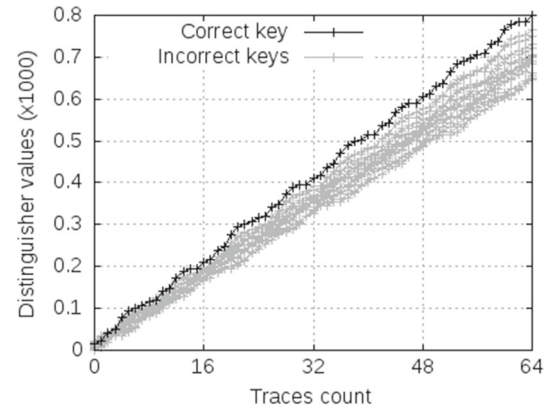


Fig. 1 Values of the distinguisher (Eq. (7)), where the $\tilde{Y}_{t,k}^T \Sigma^{-1} \tilde{Y}_{t,k}$ term (Eq. (8)) is dropped

Table 1 Dimensions of traces and models, seen as matrices in this paper

| | Raw | Coalesced |
|--------|------------------|----------------------------|
| Traces | $X : D \times Q$ | $\tilde{X} : D \times 2^n$ |
| Models | $Y : D \times Q$ | $\tilde{Y} : D \times 2^n$ |

attack. But (i) this is less and less a concern in situations where the SNR decreases (that is, the “real world” challenging scenarios), (ii) we do not need an efficient computation when number of traces is less (when the SNR is high).

In this section, we come back to genuine template attacks, as described in Sect. 3.

Definition 6 (*Equal Images under different Subkeys (EIS)* [37, Def. 2]) In this case, we have that for each pair $(k, k'), k \neq k'$, there exists a pair $(t, t'), t \neq t'$, such that $Y_{t,k} = Y_{t',k'}$.

In the case of the EIS, and in the specific case when $Y_{t,k}$ depends only on $t \oplus k$, we can drop the $\tilde{Y}_{t,k}^T \Sigma^{-1} \tilde{Y}_{t,k}$ term. The effect is represented in Fig. 1: notwithstanding that the value of the distinguisher (Eq. (7)) is always the largest for the correct key $k = k^*$, one can see “oscillations” occurring every $2^n = 16$ traces (here $n = 4$), where indeed the dropped (Eq. (8)) term is the same for all key hypotheses. We recall the dimension of matrices in Table 1.

The max-likelihood attack minimizes over k :

- either: $\text{tr} \left((X - Y_k)^T \Sigma^{-1} (X - Y_k) \right)$
- or equivalently $\sum_{t \in \mathbb{F}_2^n} n_t (\tilde{X}_t - \tilde{Y}_{t,k})^T \Sigma^{-1} (\tilde{X}_t - \tilde{Y}_{t,k})$.

Using the law of large numbers, for large Q , we have:

$$\frac{1}{Q} \sum_{t \in \mathbb{F}_2^n} n_t (\tilde{Y}_{t,k})^T \Sigma^{-1} \tilde{Y}_{t,k} \xrightarrow{Q \rightarrow +\infty} \sum_t \mathbb{P}(t) (\tilde{Y}_{t \oplus k})^T \Sigma^{-1} \tilde{Y}_{t \oplus k}$$

which does not depend on k if T is uniformly distributed, that is $\mathbb{P}(t) = 2^{-n}$. This is a mathematical justification for the observation made in Fig. 1. As already noted in Sect. 3.2, there is therefore only one key-dependent term which remains:

$$\sum_t n_t \tilde{X}_t^T \Sigma^{-1} \tilde{Y}_{t \oplus k} = \text{tr}_k(\tilde{X}^T \Sigma^{-1} \tilde{Y}), \tag{12}$$

where we recall that tr_k is the k -th trace operator introduced in Definition 2.

The templates are estimated as represented in Algorithm 1. In this formulation, it clearly appears that there is a unique noise covariance matrix Σ , which justifies the technique of ‘‘pooled covariance matrix’’ presented in [10]. Indeed, by the law of large numbers, we have that for all $t \in \mathbb{F}_2^n$:

$$\begin{aligned} \lim_{Q \rightarrow +\infty} \frac{1}{Q} \sum_{q:t_q=t} X_q &= \lim_{Q \rightarrow +\infty} \frac{1}{Q} \sum_{q:t_q=t} \tilde{Y}_{t_q} \\ &= \lim_{Q \rightarrow +\infty} \frac{1}{Q} \sum_{q:t_q=t} \tilde{Y}_t \\ &= \lim_{Q \rightarrow +\infty} \frac{\tilde{Y}_t}{Q} \sum_{q:t_q=t} 1 \\ &= \lim_{Q \rightarrow +\infty} \frac{n_t}{Q} \tilde{Y}_t \approx \frac{1}{2^n} \tilde{Y}_t, \end{aligned}$$

since $n_t \approx Q/2^n$ when Q is large. Besides, as $X_q = \tilde{Y}_{t_q} + N$, we have that:

$$\begin{aligned} \Sigma &= \frac{1}{Q} \sum_{q=1}^Q (X_q - \tilde{Y}_{t_q})(X_q - \tilde{Y}_{t_q})^T \\ &= \frac{1}{Q} \sum_{q=1}^Q X_q X_q^T \\ &\quad - \frac{1}{Q} \sum_{q=1}^Q X_q \tilde{Y}_{t_q}^T - \frac{1}{Q} \sum_{q=1}^Q \tilde{Y}_{t_q} X_q^T + \frac{1}{Q} \sum_{q=1}^Q \tilde{Y}_{t_q} \tilde{Y}_{t_q}^T \\ &\xrightarrow{Q \rightarrow +\infty} \frac{1}{Q} X X^T - \frac{1}{2^n} \tilde{Y} \tilde{Y}^T, \end{aligned}$$

assuming $n_t/Q \rightarrow 1/2^n$ when $Q \rightarrow +\infty$ (plaintext uniformity). This justifies line 9 of Algorithm 1.

4.2 Profiling and attack algorithms

The correct key is recovered using Algorithm 2. In this algorithm, the matrices \tilde{X} and \tilde{Y} are, respectively, $\tilde{X} = (\tilde{x}_0, \dots, \tilde{x}_{2^n-1})$ and $\tilde{Y} = (\tilde{y}_0, \dots, \tilde{y}_{2^n-1})$. Premise of formal presentation of template attacks, including the notion of Mahalanobis measure, is presented in a paper by Zhang et al. [44].

Input : Set of *training* traces X and corresponding plaintexts T for the estimation of the templates, for a known key k

Output : Templates:

- \tilde{Y} , the matrix of the templates, and
- Σ , its homoscedastic covariance matrix.

```

1 for  $t \in \mathbb{F}_2^n$  do // Initialization
2    $\tilde{y}_t \leftarrow 0$  // Mean trace per class
3    $n_t \leftarrow 0$  // Number of traces per class
4 for  $q \in \{1, \dots, Q\}$  do // Accumulation, as in [28]
5    $\tilde{y}_{t_q \oplus k} \leftarrow \tilde{y}_{t_q \oplus k} + X_q$ 
6    $n_{t_q \oplus k} \leftarrow n_{t_q \oplus k} + 1$ 
7 for  $t \in \mathbb{F}_2^n$  do // Normalization
8    $\tilde{y}_t \leftarrow \tilde{y}_t/n_t$ 
9 return  $(\tilde{Y} = (\tilde{y}_0, \dots, \tilde{y}_{2^n-1}), \Sigma = \frac{1}{Q} X X^T - \frac{1}{2^n} \tilde{Y} \tilde{Y}^T)$ 

```

Algorithm 1: Estimation of templates.

When the templates (\tilde{Y}, Σ) are obtained from Algorithm 1, then the attack is no longer optimal [15], but is termed template attack. However, template attacks tend to optimal attack when the profiling is carried out on a number of traces which tends to infinity.

Input : Set of *matching* traces X , corresponding plaintexts T and the templates (\tilde{Y}, Σ) obtained from Alg. 1

Output : Key guess k recovered by the optimal distinguisher

```

// ..... Accumulation, as in lines 1-8 of Alg. 1
1 for  $t \in \mathbb{F}_2^n$  do // Initialization
2    $\tilde{x}_t \leftarrow 0$  // Mean trace per class
3    $n_t \leftarrow 0$  // Number of traces per class
4 for  $q \in \{1, \dots, Q\}$  do // Accumulation
5    $\tilde{x}_{t_q} \leftarrow \tilde{x}_{t_q} + X_q$ 
6    $n_{t_q} \leftarrow n_{t_q} + 1$ 
7 for  $t \in \mathbb{F}_2^n$  do // Normalization
8    $\tilde{x}_t \leftarrow \tilde{x}_t/n_t$ 
// ..... Attack proper
9 return  $\text{argmax}_k \text{tr}_k(\tilde{X}^T \Sigma^{-1} \tilde{Y})$  // Eqn. (12)

```

Algorithm 2: Key recovery using optimal distinguisher, applied on profiling generated by Alg. 1.

4.3 Improved profiling and attack algorithms

In practice the profiling (Algorithm 1) should be carried out only once, whereas the matching (Algorithm 2) should be carried out every time when the attack outcome is needed. So the adversary can compute $\Sigma^{-1} \tilde{Y}$ once in the end of the first stage (Algorithm 3 instead of Algorithm 1), and he can use this result every times the attack needs to be estimated (Algorithm 4 instead of Algorithm 2). Finally, it is possible to

regroup the accumulations (lines 1–8) of Algorithms 1, 2, 3 and 4 in the same function.

```

Input : Set of training traces  $X$  and corresponding plaintexts  $T$  for the estimation of the templates, for a known key  $k$ 
Output : The matrix of templates left multiplied by the inverse of the covariance matrix  $\tilde{Y} = \Sigma^{-1}\tilde{Y}$ 

1 for  $t \in \mathbb{F}_2^n$  do // Initialization
2    $\tilde{y}_t \leftarrow 0$  // Mean trace per class
3    $n_t \leftarrow 0$  // Number of traces per class
4 for  $q \in \{1, \dots, Q\}$  do // Accumulation, as in [28]
5    $\tilde{y}_{t_q \oplus k} \leftarrow \tilde{y}_{t_q} \oplus X_q$ 
6    $n_{t_q \oplus k} \leftarrow n_{t_q} \oplus 1$ 
7 for  $t \in \mathbb{F}_2^n$  do // Normalization
8    $\tilde{y}_t \leftarrow \tilde{y}_t / n_t$ 
9  $\tilde{Y} \leftarrow (\tilde{y}_0, \dots, \tilde{y}_{2^n-1})$  // Same as in Alg. 1
10  $\Sigma \leftarrow \frac{1}{Q} X X^T - \frac{1}{2^n} \tilde{Y} \tilde{Y}^T$  // Same as in Alg. 1
11  $\tilde{Y} \leftarrow \Sigma^{-1} \tilde{Y}$  // Costly operation, factored here for all subsequent uses in Alg. 4
12 return  $\tilde{Y}$ 
    
```

Algorithm 3: Estimation of templates, improved by caching matrix Σ inversion at the end.

```

Input : Set of matching traces  $X$ , corresponding plaintexts  $T$  and the templates ( $\tilde{Y} = \Sigma^{-1}\tilde{Y}$ ) obtained from Alg. 3
Output : Key guess  $k$  recovered by the optimal distinguisher

// ..... Accumulation, as in lines 1-8 of Alg. 3
1 for  $t \in \mathbb{F}_2^n$  do // Initialization
2    $\tilde{x}_t \leftarrow 0$  // Mean trace per class
3    $n_t \leftarrow 0$  // Number of traces per class
4 for  $q \in \{1, \dots, Q\}$  do // Accumulation
5    $\tilde{x}_{t_q} \leftarrow \tilde{x}_{t_q} + X_q$ 
6    $n_{t_q} \leftarrow n_{t_q} + 1$ 
7 for  $t \in \mathbb{F}_2^n$  do // Normalization
8    $\tilde{x}_t \leftarrow \tilde{x}_t / n_t$ 
// ..... Attack proper
9 return  $\operatorname{argmax}_k \operatorname{tr}_k (\tilde{X}^T \tilde{Y})$  // Eqn. (12)
    
```

Algorithm 4: Improved key recovery using optimal distinguisher, applied on profiling as obtained from Alg. 3.

4.4 Extension of our approach to masked implementations

Let d be a strictly positive integer. The protection by a masking of order d consists in dividing each sensitive variable Z into $(d + 1)$ shares $Z^{(0)}, \dots, Z^{(d)}$. In order to reveal this secret value, an adversary must carry out a so-called high-order SCA. It consists in combining the leakage corresponding to

all the $(d + 1)$ shares. For such high-order attack, we leverage same notations as in Sect. 2, with the following extensions:

- $(X^{(i)})_{0 \leq i \leq d}$ denote the measured leakage corresponding, respectively, to the $(d + 1)$ shares;
- $(Y^{(i)})_{0 \leq i \leq d}$ denote the leakage model corresponding, respectively, to the $(d + 1)$ shares.

4.4.1 State-of-the-art of high-order attacks

Let us first review the high-order attacks state of the art in the case where the adversary uses only one time-sample per share during the attack. To conduct higher-order attacks, many combination functions were proposed in the literature, such as product combination [38], absolute difference combination (possibly raised to some power) [26] and sine-based combination [35]. According to [40], even if the combination functions loose the information, this loss vanishes in practice for high noise. In such case, the second-order CPA with the normalized product function becomes (nearly) equivalent to the maximum likelihood distinguisher applied to the joint distribution. According to [36] the optimal combination technique against the first-order Boolean masking is the *normalized* product $((X^{(0)} - \mathbb{E}(X^{(0)}))(X^{(1)} - \mathbb{E}(X^{(1)})))$. The authors of this paper ([36]) show that this combination function should be accompanied with $\mathbb{E}[(Y^{(0)} - \mathbb{E}(Y^{(0)}))(Y^{(1)} - \mathbb{E}(Y^{(1)})) | Z]$ as an optimal model. A high-order optimal distinguisher is introduced in [8]. This paper proves that the CPA with *normalized* product combination is optimal for high noise, independently from the masking technique and its order (whether $d = 1, 2, \dots$). Let us now show how we can use these results in order to consider several time-sample per share.

4.4.2 Extension of our approach to higher-order masking

To extend our template approach to masking-based protected implementations, we take as a starting point the *normalized* product combination [36]. In fact one can see the measured trace as a set of $(d + 1)$ sub-traces, such that each sub-trace corresponds to one of the $(d + 1)$ shares. According to [36], an adversary which combines $(d + 1)$ relevant PoIs (one from each sub-trace) should be successful in his attack by carrying out a *normalized* product combination and by accompanying it with $\mathbb{E}[\prod_{i=0}^d (Y^{(i)} - \mathbb{E}(Y^{(i)})) | Z]$ as an optimal model. In fact, for combining, the adversary can choose any relevant PoI of the first share, with any relevant PoI of the second share... with any relevant PoI of the $(d + 1)$ th share.

Following this idea, to extend the template attack on a masked implementation, one can construct an artificial trace, such that any sample in the artificial trace is a *normalized* product combination of $(d + 1)$ PoIs (one from each sub-trace). This is illustrated in Fig. 2a. In this figure, only one

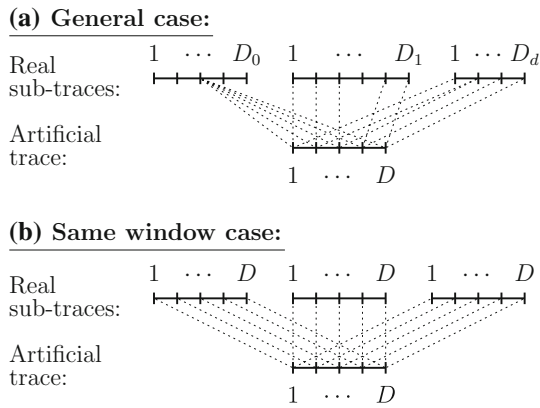


Fig. 2 Different methods to combine samples from $(d + 1)$ sub-traces, in the case of d th-order masking countermeasure. Attack is carried out on the multivariate combination shown in the artificial trace. Each sample amongst the D making up the artificial trace is a centered product of $(d + 1)$ samples taken each from a different sub-trace

PoI is selected from the first sub-trace, and it is combined to samples of the other d sub-traces, using evaluator’s expertise to get the best choice.

A simple way of combining is to take the same number of PoIs from each sub-trace and combine them in the same order (*i.e.*, the i th sample, for $1 \leq i \leq D$, of the artificial trace is the combination of all the PoI of the sub-traces that are in the i th position). Figure 2b describes the combination to create the artificial trace by such concatenation of D normalized products between $(d + 1)$ sub-traces. Finally, the adversary can profile the leakage model of this artificial leakage and carry out the template attack by matching the combination of current leakage (also by the normalized product) to the profiled model.

Thanks to this combination idea, one can carry out the template attack against high-order masking schemes, in the general way or following our particular approach (coalescence followed by the spectral approach in order to be significantly more efficient both in terms of processing time and of memory space complexities). One can see the experimental results of this technique, against a first-order Boolean masking [31], in the Sect. 5.5 (at scenarios 3 and 4).

4.4.3 Extension of our approach to disaligned traces

In fact the traces could not be aligned at the begging, due to random delays (jittering) or any other reason [13]. The impact of this countermeasure in performance has already been quantified in a formal framework for the evaluation of waveform resynchronization algorithms [22]. To carry out a template attack using such disaligned traces, one can use several techniques to first realign them [17,19,34,41]. All these techniques shall be applied before coalescence.

4.5 Computational performance analysis

4.5.1 Straightforward complexity analysis

Let us comment on the complexity of the algorithms. The body of Algorithm 1, that is lines 1–8, operates on vectors of size D . The same remark applies to the body, that is lines 1-8, of Algorithm 2. Hence, the overall complexity of these parts of the algorithm is $D \times Q$ additions.

The complex part of Algorithm 1, namely line 9, is computed only once. The overall complexity of this part equals to that of XX^T computation. So, that is equals to $D^2 \times Q$ multiplications.

The complex part of Algorithm 2, namely line 9, is also computed only once. Firstly, since the profiling stage should be done only one time, we can compute $\Sigma^{-1}\tilde{Y}$ once, as shown in Algorithm 3. The overall complexity of this multiplication equals $D^2 \times 2^n$. For inverting Σ efficiently one can use the optimized Coppersmith–Winograd-like algorithm that has a complexity of $\mathcal{O}(d^{2.373})$ [42].

Another advantage of our template analysis is that the overall complexity of the attack phase, namely Algorithm 4, does not depend on Q , no matter how large Q is. Indeed, the overall complexity of $\tilde{X}^T(\Sigma^{-1}\tilde{Y})$ computation is equal to $2^{2n} \times D$ multiplications. Table 2 shows the computing time of Algorithm 3 according to D . It can be seen that compute matrix produces to derive Σ takes more time than to the inverse of Σ . Since the time computation of Algorithm 3 depends only to lines 10 and 11 (asymptotically), we provide its duration in Table 2. Of course the overall time of Algorithm 3 is about the sum of that of lines 10 and 11.

In order to save memory space, we compute Σ in line 10 of Algorithm 3 from X and Y without using any temporary matrix. Finally, we can factor the code of lines 1–8 of Algorithms 1, 2, 3 and 4 in the same function.

4.5.2 Complexity improvement with spectral analysis

We improve the complexity of the template attack using results from [23]. To compute $\text{tr}_k(\tilde{X}^T(\Sigma^{-1}\tilde{Y}))$ more efficiently, let us first denote $(\Sigma^{-1}\tilde{Y})$ as $\tilde{\tilde{Y}}$, such that:

$$\text{tr}_k(\tilde{X}^T(\Sigma^{-1}\tilde{Y})) = \text{tr}_k(\tilde{X}^T\tilde{\tilde{Y}}).$$

Recall the dimension of \tilde{X}^T is $2^n \times D$ and that of $\tilde{\tilde{Y}}$ is $D \times 2^n$. The (i, j) element of the matrix $\tilde{X}^T\tilde{\tilde{Y}}$ is thus:

$$\tilde{X}^T\tilde{\tilde{Y}}[i][j] = \sum_{l=0}^{D-1} \tilde{X}^T[i][l]\tilde{\tilde{Y}}[l][j].$$

Table 2 Computing time (in seconds) of Algorithm 3 according to D

| | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|------------------------------------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|
| Line 10 | 0.252 | 0.869 | 1.904 | 3.354 | 5.197 | 7.488 | 10.147 | 13.259 | 16.769 | 20.740 |
| Line 11 | 0.031 | 0.177 | 0.528 | 1.155 | 2.103 | 3.486 | 5.362 | 7.794 | 10.903 | 16.021 |
| Inversion (Σ^{-1}) | 0.012 | 0.093 | 0.314 | 0.751 | 1.472 | 2.575 | 4.133 | 6.183 | 8.814 | 12.112 |
| Product ($\Sigma^{-1}\tilde{Y}$) | 0.019 | 0.084 | 0.214 | 0.404 | 0.631 | 0.911 | 1.229 | 1.611 | 2.089 | 3.909 |

Consequently,

$$\begin{aligned} \text{tr}_k \left(\tilde{X}^T \tilde{Y} \right) &= \sum_{i=0}^{2^n-1} \sum_{l=0}^{D-1} \tilde{X}^T[i][l] \tilde{Y}[l][i \oplus k] \\ &= \sum_{l=0}^{D-1} \sum_{i=0}^{2^n-1} \tilde{X}^T[i][l] \tilde{Y}[l][i \oplus k] \\ &= \sum_{l=0}^{D-1} \tilde{X}^T[.][l] \otimes \tilde{Y}[l][.](k) \\ &= \sum_{l=0}^{D-1} \text{WHT}^{-1} \left(\text{WHT}(\tilde{X}^T[.][l]) \bullet \text{WHT}(\tilde{Y}[l][.]) \right) (k) \end{aligned}$$

So,

$$\text{tr}_k \left(\tilde{X}^T \tilde{Y} \right) = \text{WHT}^{-1} \left(\sum_{l=0}^{D-1} \text{WHT}(\tilde{X}^T[.][l]) \bullet \text{WHT}(\tilde{Y}[l][.]) \right) (k), \tag{13}$$

- $\tilde{X}^T[.][l]$ denotes the l^{th} column of the matrix \tilde{X}^T ,
- $\tilde{Y}[l][.]$ denotes the l^{th} line of the matrix \tilde{Y} , and
- \otimes denotes the convolution product between them.
- \bullet denotes the coordinate wise product between them.

Thanks to the (normalized) Walsh–Hadamard transform (WHT) that allows us to compute, for all $l = 0, \dots, D - 1$, the convolution product $\tilde{X}^T[.][l] \otimes \tilde{Y}[l][.]$ with a complexity of $n2^n$ instead of 2^{2n} . Thereby the overall complexity of $\text{tr}_k \left(\tilde{X}^T(\Sigma^{-1}\tilde{Y}) \right)$ computation becomes $n2^n \times D$ instead of $2^{2n} \times D$. In applications such as key extraction from the AES ($n = 8$), the computation time of the attack phase is indeed divided by $2^n/n = 32$, which is a significant gain. Table 3 shows the computing time of the line 9 of the Algorithm 4 according to D , with and without the spectral analysis (resp. Eqs. 13 and 12).

It is noteworthy that this result that assume the group operation \oplus over the set $\{0, 1\}^n$ (i.e., $Z = F(t \oplus k)$) hold true for any other group operation \odot as long as Walsh–Hadamard Transform is replaced by Fourier Transform (FFT)

on this group ($\{0, 1\}^n, \odot$). For example, since in TEA (a Tiny Encryption Algorithm) and several AES candidates [12], the operation is $+(\text{mod}(2^n))$, so cyclical Fourier transform must replace Walsh–Hadamard transform (WHT).

5 Experiments

In this section, we assess the efficiency of this template attack. We first analyze its success rate with variable numbers of samples per trace. We compare it with the success rate of the traditional CPA. In the sequel a comparison between this template attack and the monovariate CPA attack over PCA is done.

5.1 Traces used for the case study

An ATmega 163 smart card, involving an 8-bit AVR type processor, has been programmed to process a software AES. The analysis consists in measuring the power consumption of the smart card, when the AES is running. The power measurements are done with a PicoScope 3204A on the first round. The sampling rate equals 256 MSamples/second.

We first characterize the traces managed in the following experiments. The average trace, SNR (Definition 3), NICV (Definition 4), and the principal direction of PCA (recall Sect. 3.3.1) are computed in all possible samples of these traces. Figure 3 shows those characterizations over $D = 700$ samples.

In the following template attacks, a set of traces is collected for templates building, whilst another set is collected for the attack phase. As the two sets of traces are measured from the same smart card, the results are optimistic in terms of attacker potential. We analyze the success rate of the template attack according to the size of a chosen samples window.

5.2 Template attacks with windows of increasing size

We analyzed the success rate with different numbers of samples (from $D = 1$ to $D = 700$ samples). The samples are selected around a chosen central point I_0 . This point is the sample time where the traditional CPA yields the highest

Table 3 Computing time (in seconds) of the line 9 of Algorithm 4 according to D , with and without the spectral analysis (WHT)

| | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Without WHT | 0.0619 | 0.1242 | 0.2030 | 0.2858 | 0.3447 | 0.4126 | 0.4763 | 0.5453 | 0.6151 | 0.6854 |
| With WHT | 0.0022 | 0.0042 | 0.0065 | 0.0091 | 0.0116 | 0.0135 | 0.0154 | 0.0165 | 0.0176 | 0.0201 |
| Improvement (ratio) | 28.625 | 29.752 | 30.969 | 31.373 | 29.652 | 30.534 | 30.849 | 32.969 | 34.891 | 34.183 |

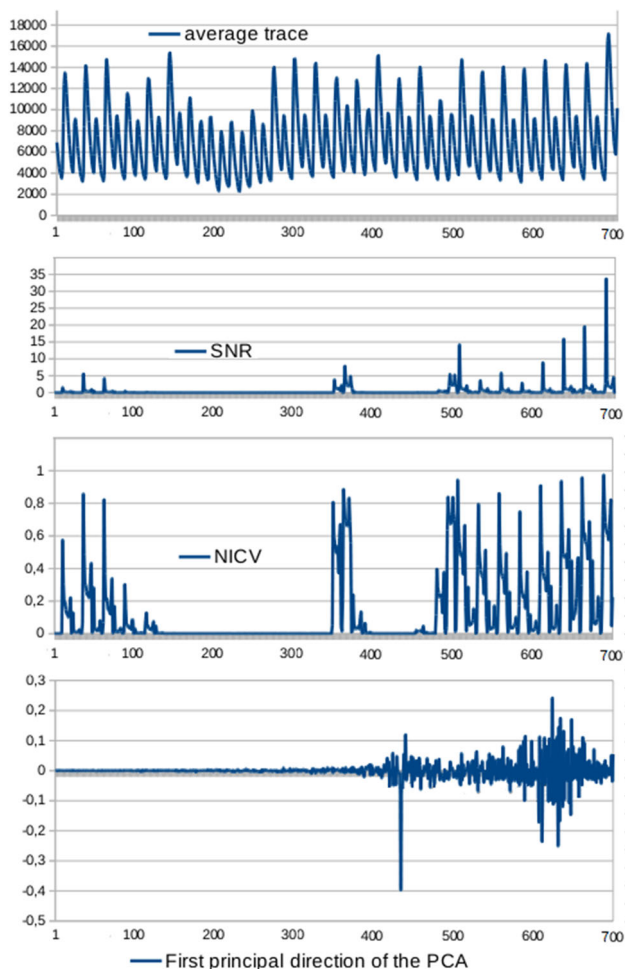


Fig. 3 Average trace, SNR, NICV and first principal direction of PCA, according to the index of samples

peak. The window of growing size D gathers samples belonging to the interval $[I_0 - D/2, I_0 + D/2)$.

Figure 4 shows the success rate according to the number of measurements for different sizes of the window (from $D = 200$ to $D = 700$). From these results, one can deduce that the larger the dimensionality, the fewer the number of traces to recover the secret key. Asymptotically, for very large dimensionalities $D \rightarrow +\infty$, only a limited few number of traces suffices to extract the key.

For reference, Fig. 4 also shows the success rate of univariate CPA (which corresponds to the maximum value of

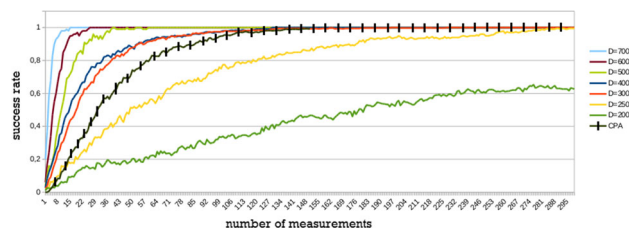


Fig. 4 Values of success rate, according to the number of samples

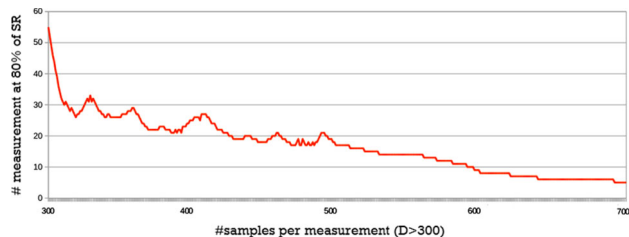


Fig. 5 Number of measurements at 80% of success rate, according to the number of samples (zoom for $D > 300$)

the correlation coefficient along all $D = 700$ samples). The CPA uses Hamming weight model. It can be seen that (model-agnostic) template attack becomes more efficient than CPA starting from about $D \in [250, 300]$, and keeps being better for larger dimensionality.

Indeed, as shown in Fig. 5, this monotony of the success rate is approximately respected above $D = 300$ samples/trace. But, as shown in Fig. 6, this monotony is not respected below $D = 300$. One can deduce that in this interval the success rate depends of the signal wave form; in the interval $D \in [1, 300]$, it can happen that when increasing the window size, more *noise* than *signal* is injected in the template attack, thereby making it less efficient. In order to lead this attack more efficiently one can choose only the samples where the curve in Fig. 6 decreases, because these samples really depend on $T \oplus k$ contrary to the samples where the curve increases.

As shown in Fig. 4, from around 280 samples/trace this multivariate analysis will be more efficient than the traditional CPA. So one can conclude that there is a venue to capture more information using part or all of the samples rather than conducting a monivariate attack.

In order to validate our study independently from the targeted device, the same experiment is carried out on a more

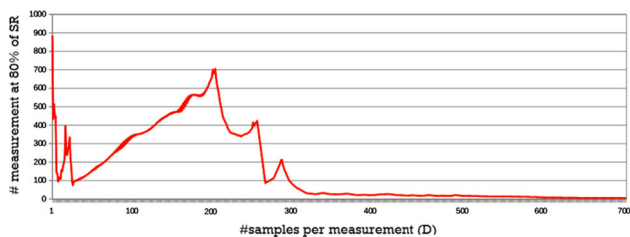


Fig. 6 Number of measurements at 80% of success rate, according to the number of samples (full scale, that is $D \in [1, 700]$)

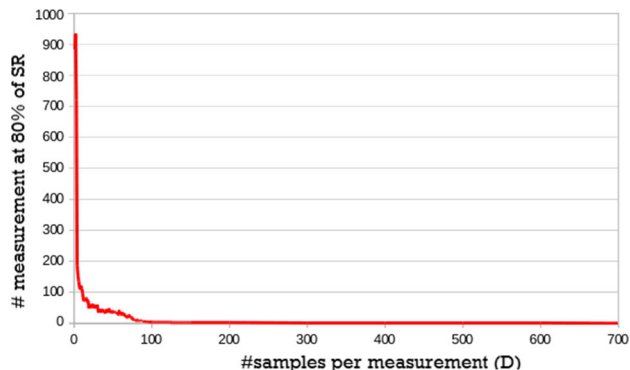


Fig. 7 Number of measurements at 80% of success rate, according to the number of samples (full scale, that is $D \in [1, 700]$) for the TI MSP430G2553 card

recent smartcard, namely TI MSP430G2553. Figure 7 shows similar results on this device, comparing to the right part of Fig. 6 ($D \geq 300$). One can see that the more samples are taken into account, the faster the attack. This figure shows also that it is very important to choose the relevant PoIs, and that template attacks benefit from the great multiplicity of PoI. The gain is huge: more than $100\times$ less traces to recover the key when using $D = 1$ vs $D = 700$.

It is noticeable that the using sampling frequency during the MSP430G2553's (resp. the Atmega163's) attack is 512 MHz (resp. 256 MHz), where the operating frequency is 16 MHz (resp. 8 MHz). So the two campaigns are comparable in terms of number of samples per clock cycle. The difference is in the noise, since the two circuits have not been designed in the same technology and the side-channel acquisition system differ (MSP430 LaunchPad and Smartcard reader, respectively).

It is also noteworthy that both of PoI-based and PCA-based template attacks [20] can straightforwardly benefit from our approach.

5.3 Comparison with PCA

In order to compare the efficiency of this method with other multivariate analyses, the PCA analysis is carried out on the same device with the same number of samples (700 samples/trace). Figure 8 shows that the template attack is more

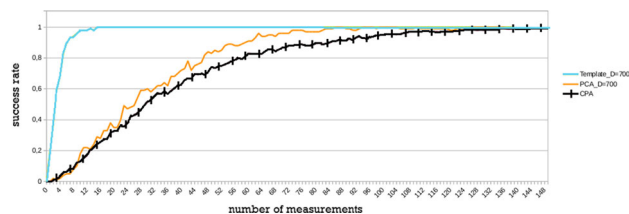


Fig. 8 Values of success rate, according to the number of samples

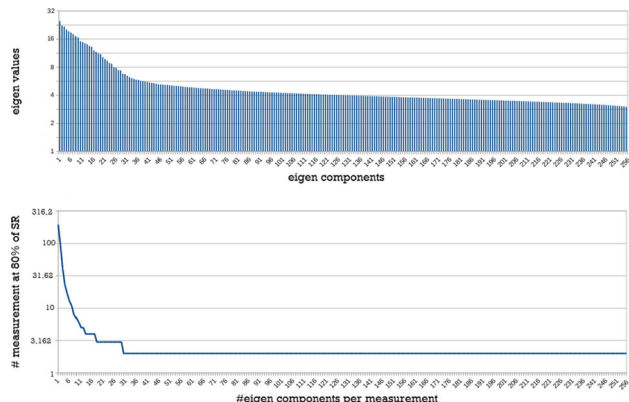


Fig. 9 PCA with multiple directions (from 1 to $2^n - 1$ directions). Top: scree plot of eigenvalues (log scale). Bottom: number of measurements at 80% of success rate (log scale)

efficient in practice than both the PCA and the traditional CPA. Recall that Fig. 3 shows the average trace, SNR, NICV and the first principal direction of PCA, according to the index of samples.

5.4 Template attack after dimensionality reduction (over first eigen-components)

In this stage we assess the effect of PCA over the success rate of our template attack. Instead of assessing the efficiency of our attack according to the number of samples per trace it is assessed according to the number of directions of PCA ordered by decreasing eigenvalue. Figure 9 shows the number of measurements required to reach 80% of success rate, according to the number of eigen-components considered per measurement. This figure shows also the ordered eigenvalues according to their corresponding eigen-components. It is clear that a quasi-linear relation exists between them: the success rate increases about as fast as the cumulative eigenvalues.

5.5 Study of our approach with simulated traces

In order to do a fair comparison under different aspects (PoIs, noise levels and masking) one can resort to simulated traces. In this paper we present four scenarios with different noise levels. In each scenario we increase a window size around a

central PoI and we show the number of the required traces to reach 80% of success rate. So, we carry out, at once, a study with different PoI numbers, different noise levels and using the masking technique or not. The four scenarios are:

1. In the first scenario, we simulate a leakage that follows a half cycle of a sine function, during 700 samples (or equivalently any 700 PoIs that follows the same law). More exactly, if the sensitive variable is Z , then the leakage at the s th time sample is:

$$X_z(s) = w_H(z) \sin\left(2\pi \frac{s}{2 \times 700}\right) + N,$$

such that N denotes a Gaussian noise and the leakage model is $Y_z(s) = w_H(z) \sin\left(2\pi \frac{s}{2 \times 700}\right)$. In those equations, $w_H(z)$ is the Hamming weight of $z \in \mathbb{F}_2^n$, that is $w_H(z) = \sum_{i=1}^n z_i$. We used such leakage simulation in order to be close to the real leakage (for example, see the real leakage of a smart card as shown in [33, Fig.4]). During this experiment we vary the window size around the central PoI ($s = 350$). Figure 10 shows the number of the required traces to reach 80% of success rate according to the number of samples, for different standard deviations (σ) of the noise, in the first scenario.

2. In the second scenario, we simulate a leakage that follows the same leakage function as the first scenario but only for the even values of s . For the odd values of s we simulate non-relevant points (non-informative points [27]) by random values. Figure 11 shows the number of required traces to reach 80% of success rate according to the number of samples, for different noise standard deviations (σ), in the second scenario. One can show the inconvenience of considering non-relevant points as PoIs.
3. In the third scenario, we simulate the leakage of a protected device by first-order Boolean masking [32, §4]. For each share (masked sensitive value and the mask) the leakage is simulated by a sine function as in the first scenario. Before carrying out our approach, we combine the sub-traces in one artificial trace as described in Sect. 4.4. Figure 12 shows the number of required traces to reach 80% of success rate according to the number of samples, for two different standard deviations (σ) of the noise, in the third scenario. One can see that it is possible to reveal the secret by applying our approach even against masking.
4. In the fourth scenario, we simulate a leakage that follows the same leakage function than the third scenario but only for the even values of s . For the odd values of s we simulate non-relevant points (non-informative points) by random values. This alternation between relevant and dummy samples holds for both shares. Figure 13 shows the number of the required traces to reach 80% of suc-

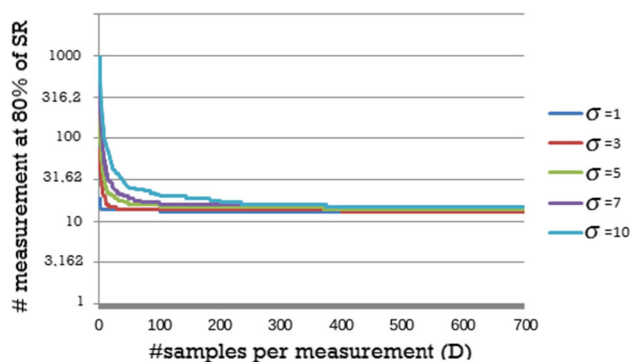


Fig. 10 Number of measurements at 80% of success rate (log scale), according to the number of samples, in scenario 1

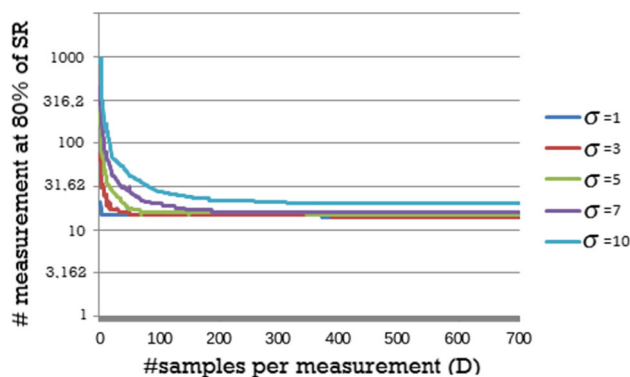


Fig. 11 Number of measurements at 80% of success rate (log scale), according to the number of samples, in scenario 2

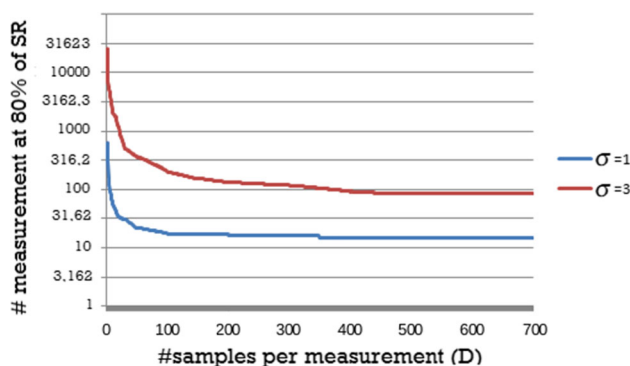


Fig. 12 Number of measurements at 80% of success rate (log scale), according to the number of samples, in scenario 3

cess rate according to the number of samples, for two different standard deviations (σ) of the noise, in the last scenario. This figure shows the same trend as Fig. 12. The convergence rate with respect to dimensionality (#samples) is similar. But the asymptotic value (#samples to recover the key with 80% of success) is of course higher, especially for high noise.

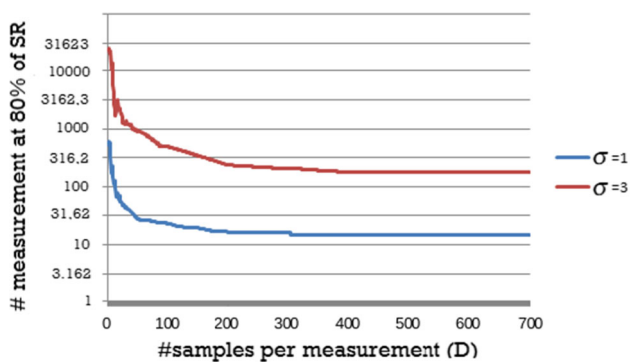


Fig. 13 Number of measurements at 80% of success rate (log scale), according to the number of samples, in scenario 4

6 Conclusion and perspectives

In this paper, we have provided an analytical formula for the template attack in a multivariate Gaussian setting. We have applied it to highly multivariate traces, and we have shown that template attacks outperform state-of-the-art heuristics, such as traces dimensionality reduction followed by monovariate distinguishers. Template attacks without prior dimensionality reduction can be applied to traces of dimensionality D of several hundreds without any effectiveness loss: The success rate increases as D increases. Therefore, this study reveals that the high sampling rate of oscilloscopes can help increase the success rate of the attacks.

Furthermore, we extend the approach to the masking-based protected implementations. We also exhibit a spectral approach for template attack which allows an exponential computational improvement in the attack phase (with respect to data bitwidth), which turns out to be a speed-up by $32\times$ in the case of AES. Recall that both of PoI-based and PCA-based template attacks can straightforwardly benefit from our approach, both in terms of processing time and in term of the needed space memory. Our approach is validated both by simulated and real-world traces.

References

1. Archambeau, C., Peeters, É., Standaert, F.-X., Quisquater, J.-J.: Template Attacks in Principal Subspaces. In: CHES, Vol. 4249 of LNCS, pp. 1–14. Springer, Yokohama, Japan, October 10–13 (2006)
2. Bär, M., Drexler, H., Pulkus, J.: Improved Template Attacks. In: COSADE, pp. 81–89. Darmstadt, Germany, February 4–5 (2010) http://cosade2010.cased.de/files/proceedings/cosade2010_paper_14.pdf
3. Bhasin, S., Danger, J.-L., Guilley, S., Najm, Z.: NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. In: International Symposium on Electromagnetic Compatibility (EMC '14 / Tokyo). IEEE, Session OS09: EM Information Leakage. Hitotsubashi Hall (National Center of Sciences), Chiyoda, Tokyo, Japan, May 12–16 (2014)
4. Bhasin, S., Danger, J.-L., Guilley, S., Najm, Z.: Side-channel Leakage and Trace Compression Using Normalized Inter-class Variance. In: Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy, HASP '14, pp. 7:1–7:9. New York, NY, USA, ACM (2014)
5. Brier, É., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds), Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings, vol. 3156 of Lecture Notes in Computer Science, pp. 16–29. Springer (2004)
6. Bruneau, N., Guilley, S., Heuser, A., Marion, D., Rioul, O.: Less is More - Dimensionality Reduction from a Theoretical Perspective. In: Güneysu, T., Handschuh, H. (eds), Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13–16, 2015, Proceedings, vol. 9293 of Lecture Notes in Computer Science, pp. 22–41. Springer (2015)
7. Bruneau, N., Guilley, S., Heuser, A., Marion, D., Rioul, O.: Optimal side-channel attacks for multivariate leakages and multiple models. *J. Cryptogr. Eng.* 7(4), 331–341 (2017)
8. Bruneau, N., Guilley, S., Heuser, A., Rioul, O.: Masks Will Fall Off – Higher-Order Optimal Distinguishers. In: Sarkar, P., Iwata, T. (eds), Advances in Cryptology – ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014, Proceedings, Part II, vol. 8874 of Lecture Notes in Computer Science, pp. 344–365. Springer (2014)
9. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Burton, S., Kaliski, Jr., Koç, Ç.K., Paar, C. (eds), Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13–15, 2002, Revised Papers, vol. 2523 of Lecture Notes in Computer Science, pp. 13–28. Springer (2002)
10. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds), Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27–29, 2013. Revised Selected Papers, vol. 8419 of LNCS, pp. 253–270. Springer (2013)
11. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Koç, Ç.K., Paar, C. (eds), CHES, vol. 1965 of Lecture Notes in Computer Science, pp. 252–263. Springer (2000)
12. Coron, J.-S., Goubin, L.: On Boolean and Arithmetic Masking against Differential Power Analysis. In: CHES, vol. 1965 of Lecture Notes in Computer Science, pp. 231–237. Springer, Worcester, MA, USA, August 17–18 (2000)
13. Coron, J.-S., Kizhvatov, I.: An efficient method for random delay generation in embedded software. In: Clavier, C., Gaj, K. (eds), Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6–9, 2009, Proceedings, vol. 5747 of Lecture Notes in Computer Science, pp. 156–170. Springer (2009)
14. Coron, J.-S., Vadnala, P.K., Giraud, C., Prouff, E., Renner, S., Rivain, M.: Conversion of Security Proofs from One Model to Another: A New Issue. In: COSADE, Lecture Notes in Computer Science. Springer, Darmstadt, Germany, May 3–4 (2012)
15. de Chérisey, É., Guilley, S., Heuser, A., Rioul, O.: On the optimality and practicability of mutual information analysis in some scenarios. *Cryptography and Communications* (Jul 2017) <https://dblp.uni-trier.de/rec/journals/ccds/CheriseyGHR18.html?view=bibtex>
16. Debande, N., Souissi, Y., Abdelaziz Elaabid, M., Guilley, S., Danger, J.-L.: Wavelet transform based pre-processing for side channel analysis. In: 45th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2012, Workshops Proceedings, Van-

- couver, BC, Canada, December 1-5, 2012, pp. 32–38. IEEE Computer Society (2012)
17. Durvaux, F., Renaud, M., Standaert, F.-X., van Oldeneel tot Oldenzeel, L., Veyrat-Charvillon, N.: Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models. In: Mangard, S. (ed) CARDIS, vol. 7771 of Lecture Notes in Computer Science, pp. 123–140. Springer (2012)
 18. Abdelaziz Elaabid, M., Guilley, S.: Practical Improvements of Profiled Side-Channel Attacks on a Hardware Crypto-Accelerator. In: Bernstein, D.J., Lange, T. (eds), Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings, vol. 6055 of Lecture Notes in Computer Science, pp. 243–260. Springer (2010)
 19. Facon, A., Guilley, S., Lec’Hvien, M., Marion, D., Perianin, T.: Binary Data Analysis for Source Code Leakage Assessment, In: 11th International Conference, SecITC 2018, Bucharest, Romania, November 8–9, 2018, Revised Selected Papers, pp. 391–409. 01 (2019)
 20. Fan, G., Zhou, Y., Zhang, H., Feng, D.: How to Choose Interesting Points for Template Attacks More Effectively? In: Yung, M., Zhu, L., Yang, Y. (eds), Trusted Systems - 6th International Conference, INTRUST 2014, Beijing, China, December 16-17, 2014, Revised Selected Papers, vol. 9473 of Lecture Notes in Computer Science, pp. 168–183. Springer (2014)
 21. Guilley, S., Heuser, A., Tang, M., Rioul, O.: Stochastic Side-Channel Leakage Analysis via Orthonormal Decomposition. In: Farshim, P., Simion, E. (eds), Innovative Security Solutions for Information Technology and Communications - 10th International Conference, SecITC 2017, Bucharest, Romania, June 8-9, 2017, Revised Selected Papers, vol. 10543 of Lecture Notes in Computer Science, pp. 12–27. Springer (2017)
 22. Guilley, S., Khalfallah, K., Lomné, V., Danger, J.-L.: Formal Framework for the Evaluation of Waveform Resynchronization Algorithms. In: Ardagna, C.A., Zhou J. (eds), Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings, vol. 6633 of Lecture Notes in Computer Science, pp. 100–115. Springer (2011)
 23. Guillot, P., Millérioux, G., Dravie, B., El Mrabet, N.: Spectral Approach for Correlation Power Analysis. In: El Hajji, S., Nitaj, A., Souidi, E.M. (eds), Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet, vol. 10194 of Lecture Notes in Computer Science, pp. 238–253. Springer (2017)
 24. Hajra, S., Mukhopadhyay, D.: Reaching the limit of nonprofiling DPA. IEEE Trans. CAD Integr. Circuits Syst. **34**(6), 915–927 (2015)
 25. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics (2002). ISBN: 0387954422
 26. Joye, M., Paillier, P., Schoenmakers, B.: On Second-Order Differential Power Analysis. In: CHES, vol. 3659 of LNCS, pp. 293–308. Springer, Edinburgh, UK, August 29 – September 1st (2005)
 27. Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., Standaert, F.-X.: Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In Mangard, S., Poschmann A.Y. (eds), Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers, vol. 9064 of *Lecture Notes in Computer Science*, pp. 20–33. Springer (2015)
 28. Lomné, V., Prouff, E., Roche, T.: Behind the Scene of Side Channel Attacks. In: Sako, K., Sarkar P., (eds), ASIACRYPT (1), vol. 8269 of Lecture Notes in Computer Science, pp. 506–525. Springer (2013)
 29. Maghrebi, H., Prouff, E.: On the Use of Independent Component Analysis to Denoise Side-Channel Measurements. In: Fan, J., Gierlichs, B. (eds), Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings, vol. 10815 of Lecture Notes in Computer Science, pp. 61–81. Springer (2018)
 30. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. <http://www.springer.com/Springer> (December 2006). ISBN 0-387-30857-1, <http://www.dpabook.org/>
 31. Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Fast Software Encryption’00, pp. 150–164. Springer-Verlag, New York (April 2000)
 32. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Schneier, B. (ed) Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings, vol. 1978 of Lecture Notes in Computer Science, pp. 150–164. Springer (2000)
 33. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining Smart-Card Security under the Threat of Power Analysis Attacks. IEEE Trans. Comput. **51**(5), 541–552 (2002)
 34. Di Natale, G., Flottes, M.-L., Rouzeyre, B., Valka, M., Réal, D.: Power consumption traces realignment to improve differential power analysis. In: Kraemer, R., Pawlak, A., Steiner, A., Schölzel, M., Raik, J., Vierhaus, H.T. (eds), DDECS, pp. 201–206. IEEE (2011)
 35. Oswald, E., Mangard, S.: Template Attacks on Masking — Resistance Is Futile. In: Abe, M. (ed), CT-RSA, vol. 4377 of Lecture Notes in Computer Science, pp. 243–256. Springer (2007)
 36. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
 37. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: LNCS, (ed), CHES, vol. 3659 of LNCS, pp. 30–46. Springer, Edinburgh, Scotland, UK (Sept 2005)
 38. Schramm, K., Paar, C.: Higher Order Masking of the AES. In: Pointcheval, D. (ed), CT-RSA, vol. 3860 of LNCS, pp. 208–225. Springer (2006)
 39. Standaert, F.-X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In: CHES, vol. 5154 of Lecture Notes in Computer Science, pp. 411–425. Springer, Washington, D.C., USA. August 10–13 (2008)
 40. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World is Not Enough: Another Look on Second-Order DPA. In: ASIACRYPT, vol. 6477 of LNCS, pp. 112–129. Springer, Singapore, December 5-9 (2010) <http://www.dice.ucl.ac.be/~fstandae/PUBLIS/88.pdf>
 41. van Woudenberg, J.G.J., Witteman, M.F., Bakker, B.: Improving Differential Power Analysis by Elastic Alignment. In: Kiayias, A. (ed), CT-RSA, vol. 6558 of Lecture Notes in Computer Science, pp. 104–119. Springer (2011)
 42. Williams, V.V.: Multiplying matrices faster than coppersmith-wograd. In: STOC’12 Proceedings of the forty-fourth annual ACM symposium on Theory of computing, New York, USA — May 19-22, 2012, pp. 887–898, 05 (2012)
 43. Zhang, H., Zhou, Y.: How many interesting points should be used in a template attack? J. Syst. Softw. **120**, 105–113 (2016)
 44. Zhang, H., Zhou, Y., Feng, D.: Mahalanobis distance similarity measure based distinguisher for template attack. Security Commun. Netw. **8**(5), 769–777 (2015)
 45. Zheng, Y., Zhou, Y., Yu, Z., Hu, C., Zhang, H.: How to Compare Selections of Points of Interest for Side-Channel Distinguishers in

Practice? In: Chi Kwong Hui, L., Qing, S.H., Shi, E., Yiu, S.-M. (eds), Information and Communications Security - 16th International Conference, ICICS 2014, Hong Kong, China, December 16-17, 2014, Revised Selected Papers, vol. 8958 of Lecture Notes in Computer Science, pp. 200–214. Springer (2014)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.