Exact Template Attacks with Spectral Computation

1st Meriem MAHAR
Université Paris 8, LAGA, UMR 7539, France
Centre de Recherche sur l'Information Scientifique et Technique (CERIST),
Algiers, Algeria. meriem.mahar@etud.univ-paris8.fr

2nd Maamar OULADJ
CERIST, Algiers, Algeria.
ouladj.maamar@gmail.com
https://orcid.org/0000-0003-0976-312X

3rd Sylvain GUILLEY
Secure-IC S.A.S. Digital Park B,
801 avenue des Champs Blancs, Rennes, France,
École Normale Supérieure (ENS), Paris, France.
https://orcid.org/0000-0002-5044-3534

4th Hacène BELBACHIR
RECITS Laboratory, Mathematics faculty,
USTHB, Algiers, Algeria.
hacenebelbachir@gmail.com
https://orcid.org/0000-0001-8540-3033

5th Farid MOKRANE
Université Paris 8,
LAGA, UMR 7539,
France.
farid.mokrane@univ-paris8.fr

***Abstract*—The so-called Gaussian template attacks (TA) is one of the optimal Side-Channel Analyses (SCA) when the measurements are captured with normal noise. In the SCA literature, several optimizations of its implementation are introduced, such as coalescence and spectral computation. The coalescence consists of averaging traces corresponding to the same plaintext value, thereby coalescing (synonymous: compacting) the dataset. Spectral computation consists of sharing the computational workload when estimating likelihood across key hypotheses.**

State-of-the-art coalescence leverages the Law of Large Numbers (LLN) to compute the mean of equivalent traces. This approach comes with a drawback because the LLN is just an asymptotic approximation. So it does not lead to an exact Template Attack, especially for a few number of traces. In this paper, we introduce a way of calculating the TA exactly and with the same computational complexity (using the spectral approach), without using the LLN, regardless of the number of messages.

For the experimental validation of this approach, we use the ANSSI SCA Database (ASCAD), with different numbers of messages and different amounts of samples per trace. Recall that this dataset concerns a software implementation of AES-128 bits, running on an ATMEGA-8515 microprocessor.

***Index Terms*—Spectral approach, Template Attack, Multivariate analysis, Attack speed-up, Coalescence, Law of Large Numbers (LLN).**

I. Introduction

Embedded systems are increasingly present in everyday life. They require data protection by encryption protocols. These algorithms are a natural target of several analyses and attacks. They can exploit the mathematical weaknesses of these algorithms. Alternatively, they exploit their implementation weaknesses, including non-voluntary information leaks such as side-channels (electricity consumption, electromagnetic radiation, temperature, sound emanations, and time execution [7], etc.). This latter type of attack is very effective in recovering the encryption key, compared to the classic case [6].

Instead of carrying an SCA straightforwardly, that is on a side-channel trace-by-trace basis, one can first average each class of traces that correspond to the same message (or equivalently to the same sensitive value [11]). The advantage of this approach is the reduction of the complexity simultaneously in both the memory space and the computation. This approach was subsequently formalized and named "coalescence" in the SCA literature [8]. Then, coalescence has been extended from Linear Regression-Based Side-Channel Attacks (LRA) [9] to Correlation Power Analysis (CPA) [8, Chap.6] and Template Attacks [10].

In this last paper, the authors took advantage of the coalescence to introduce a spectral computation (using Fourier Transform) to speed up calculation. Namely, they progress from a quadratic to a quasi-linear complexity.

Nevertheless, they have only approximated the template attack [2] with coalescence, thanks to the law of large numbers. Indeed, the drawback of coalescence is that it leads to optimal SCAs asymptotically, but not for a limited amount of traces.

In what follows, we demonstrate, for the first time, that it is possible to calculate the template attack (TA) exactly (without approximation), and using the spectral computation, so as not to lose anything in complexity.

a) Notations: In this paper, random variables are denoted by a capital letter. A realization of a random variable (e.g. X) is denoted by the corresponding lower-case letter (e.g., x). A sample of several observations of X is denoted by $(x_i)_i$. It is sometimes referred to as a vector. The notation $(x_i)_i \hookleftarrow X$ means the initialization of the set of observations $(x_i)_i$ from X.

Calligraphic letters will denote matrices, so the elements of a $\mathscr{M}$ matrix will be denoted by $\mathscr{M}[i][j]$. In addition, its u^{th} column is denoted by $\mathscr{M}[u]$ and its i^{th} row is denoted by $\mathscr{M}^T[i]$.

In this paper, we shall consider that the attacker (the adversary) targets a single sensitive variable denoted by Z. The results can be directly extended to the general case, where several variables are targeted in parallel.

The sensitive variable Z depends on a public variable X (usually a plaintext or a ciphertext) which lives in $\mathbb{F}_2^n$ and a secret subkey k^* which also lives in $\mathbb{F}_2^n$, such that $Z = F(X, k^*)$

979-8-3503-6806-2/24/$31.00 ©2024 IEEE

where F is a known mapping $F : \mathbb{F}_2^n * \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ pertaining to the cryptographic algorithm specification. The lengths n and m (integer number of bits) depend on the targeted cryptographic algorithm and device architecture.

Let L denote the random variable's leakage model. An attack is carried out with N leakage traces $l_0, \ldots, l_{N-1}$. Each $(l_q \hookleftarrow L)_{1 \leq q \leq N}$ corresponds to the processing of $z_i = F(x_i, k^*)$, such that, $x_i \hookleftarrow X$, $z_i \hookleftarrow Z$. Besides, k^* denotes the real key the attacker is looking for. As a typical example, $Z = \mathsf{Sbox}(X \oplus k^*)$, where Sbox denotes a substitution box and $\oplus$ denotes the bit-wise addition (*xor*).

The number of samples per trace (leakage time points) is denoted by D (like in dimensionality).

Let us also consider the Gaussian leakage model, where $L \sim \mathcal{N}(M, \Sigma)$, such that M denotes the mean and Σ denotes the covariance matrix. In the sequel, Σ is assumed invertible, which is usually the case in practice. Elsewhere, a pseudo-inverse can be used instead. One can also denote the leakage as $L = M + noise$, where $noise \sim \mathcal{N}(0, \Sigma)$.

In summary, the model can be noted as follows:

$$X, k \longrightarrow Z \longrightarrow M(Z) = M \longrightarrow L = M + noise. \qquad (1)$$

The adversary should guess the true key by $\hat{k}$.

b) Contributions: Thanks to this work, we managed to optimize the Template Attack in terms of success rate without any loss in the computational complexity. Indeed, we show that we need to weight the model and the average of the traces class by the corresponding class's cardinality (Proposition 1). In addition, we demonstrate that we can draw profiles from the spectral approach without having to resort to the coalescence principle.

c) Outline: The rest of the paper is structured as follows. Our main result, namely the rewriting of the exact template attack, is given in Sec. II. Its optimal implementation leveraging a Fourier transform is the topic of Sec. III. The experimental validation is given in Sec. IV. Eventually, section V concludes the paper and opens some research perspectives.

II. Formal proof

From the Equ. (1) and According to the state of the art (more precisely [1, Theorem 2]), one can use the following formula for guessing the key during the optimal (exact) template attack:

$$\hat{k} = \underset{k}{\operatorname{argmin}}\ tr\left((L - M_k)^\mathsf{T} \Sigma^{-1} (L - M_k)\right) \ . \qquad (2)$$

After applying the coalescence principle [10], to improve the time of the exact calculation of the equation (2), the guessed key can be carried out by:

$$\hat{k} = \underset{k}{\operatorname{argmin}} \sum_{x=0}^{2^n-1} n_x (\tilde{L}_x - \tilde{M}_{x,k})^\mathsf{T} \Sigma^{-1} (\tilde{L}_x - \tilde{M}_{x,k}) \ . \qquad (3)$$

The equation (3) can be read as a matrix trace over the plaintext space (or the ciphertext), weighted by n_x values. Recalling that:

- n_x is the number of times the message x is involved,
- $\tilde{L}_x$ is the average trace over over all the traces corresponding to the same message x,
- $\tilde{M}_{x,k}$ is leakage model corresponding to the couple (x,k).

It is essential to point out that the attack presented by the last equation is exactly the same as that of the equation (2), and so they will succeed with the same number of traces. However, the attack (3) is more efficient in terms of computation, and memory space, than (2), as soon as the number of traces N is greater than the number of plaintexts involved in the leakage model (e.g., for *AES*, it is $2^n = 256$).

This gain applies to both the profiling and the matching phases. However, we emphasize that profiling requires many more traces than matching, so most of the gain from using coalesced data comes from the model-building phase.

In what follows, we will demonstrate how to calculate (3) without using the approximation by the LLN, contrary to the state of the art.

Proposition 1 (Exact Template Attack – Expression of the Maximum Likelihood Distinguisher)**.**

$$\hat{k} = \underset{k}{\operatorname{argmin}} \sum_{x=0}^{2^n-1} n_x \tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1} \tilde{M}_{x\oplus k} - 2 \sum_{x=0}^{2^n-1} (n_x \tilde{L}_x^\mathsf{T})(\Sigma^{-1} \tilde{M}_{x\oplus k}) \ .$$

Proof. Let us rewrite (3) by developing the terms:

$$\begin{aligned}
\hat{k} &= \underset{k}{\operatorname{argmin}} \sum_{x=0}^{2^n-1} n_x (\tilde{L}_x - \tilde{M}_{x,k})^\mathsf{T} \Sigma^{-1} (\tilde{L}_x - \tilde{M}_{x,k}) \\
&= \underset{k}{\operatorname{argmin}} \Big[\sum_{x=0}^{2^n-1} n_x \tilde{L}_x^\mathsf{T} \Sigma^{-1} \tilde{L}_x + \sum_{x=0}^{2^n-1} n_x \tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1} \tilde{M}_{x\oplus k} \\
&\qquad - \sum_{x=0}^{2^n-1} n_x \tilde{L}_x^\mathsf{T} \Sigma^{-1} \tilde{M}_{x\oplus k} - \sum_{x=0}^{2^n-1} n_x \tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1} \tilde{M}_x \Big].
\end{aligned}$$

Given that the term $\sum n_x [\tilde{L}_x^\mathsf{T} \Sigma^{-1}] \tilde{L}_x$ is independent of the key k, then finding $\hat{k}$ is equivalent to minimizing

$$\sum_{x=0}^{2^n-1} n_x \left[\tilde{L}_x^\mathsf{T} \Sigma^{-1} \tilde{L}_x + \tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1} \tilde{M}_{x\oplus k} - \tilde{L}_x^\mathsf{T} \Sigma^{-1} \tilde{M}_{x\oplus k} - \tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1} \tilde{L}_x \right].$$

Notice that:

$$\sum_{x=0}^{2^n-1} (n_x \tilde{L}_x^\mathsf{T})(\Sigma^{-1} \tilde{M}_{x\oplus k}) = \sum_{x=0}^{2^n-1} (\tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1})(n_x \tilde{L}_x),$$

So, finally:

$$\hat{k} = \underset{k}{\operatorname{argmin}} \sum_{x=0}^{2^n-1} n_x \tilde{M}_{x\oplus k}^\mathsf{T} \Sigma^{-1} \tilde{M}_{x\oplus k} - 2 \sum_{x=0}^{2^n-1} (n_x \tilde{L}_x^\mathsf{T})(\Sigma^{-1} \tilde{M}_{x\oplus k}) \ .$$

□

III. Spectral expression

Recalling that, for any pair of pseudo-Boolean functions f and g, we have:

$$\sum_{x=0}^{2^n-1} f(x).g(x \oplus k) = (f \otimes g)(k) \quad = WHT(WHT(f) \bullet WHT(g))(k),$$

where

1) "$\bullet$" denotes the direct product between two pseudo-Boolean functions (that is, the term-to-term product),
2) "$\otimes$" denotes the convolution product between two pseudo-Boolean functions,
3) WHT denotes the Walsh-Hadamard Transform. This transform is defined as:

$$WHT(f)(u) = \sum_x (-1)^{u \cdot x} f(x).$$

The convolution product can be computed naïvely in $\mathscr{O}(n^2)$ complexity, but also efficiently thanks to computing the WHT by a butterfly algorithm in quasi-linear $\mathscr{O}(n \log_2 n)$ complexity [4].

Hence, let us note: $\mathscr{M}(x) \doteq \tilde{M}_x^{\mathsf{T}} \Sigma^{-1} \tilde{M}_x$; i.e. $\mathscr{M}(x)$ is a scalar (dimension 1×1); Let $L_{cumul}(x) \doteq n_x \tilde{L}_x^{\mathsf{T}}$; i.e. $L_{cumul}(x)$ is of dimension $(1 \times D)$; $\tilde{\mathbb{M}}(x) \doteq \Sigma^{-1} \tilde{M}_x$; that is $\tilde{\mathbb{M}}(x)$ is of dimension $(D \times 1)$.

So,

$$\begin{aligned}\sum_{x=0}^{2^n-1} (n_x \tilde{L}_x^{\mathsf{T}})(\Sigma^{-1} \tilde{M}_{x \oplus k}) &= \sum_{x=0}^{2^n-1} L_{cumul}(x) \tilde{\mathbb{M}}(x \oplus k) \\ &= \sum_{u=1}^{D} L_{cumul}[u] \otimes \tilde{\mathbb{M}}[u](k) \\ &= \sum_{u=1}^{D} WHT^{-1}\big[WHT(L_{cumul}[u]) \bullet WHT(\tilde{\mathbb{M}})[u]\big](k) \ .\end{aligned}$$

Since WHT is a linear mirror function (equal to the inverse of itself), one has that:

$$\begin{aligned}&\sum_{x=0}^{2^n-1} (n_x \tilde{L}_x^{\mathsf{T}})(\Sigma^{-1} \tilde{M}_{x \oplus k}) \\ &\qquad = WHT\big[\sum_{u=1}^{D} WHT(L_{cumul}[u]) \bullet WHT(\tilde{\mathbb{M}}[u])\big](k) \ .\end{aligned}$$

As a result,

$$\begin{aligned}\hat{k} &= \operatorname*{argmin}_k \sum_{x=0}^{2^n-1} n_x \tilde{M}_{x \oplus k}^{\mathsf{T}} \Sigma^{-1} \tilde{M}_{x \oplus k} - 2 \sum_{x=0}^{2^n-1} (n_x \tilde{L}_x^{\mathsf{T}})(\Sigma^{-1} \tilde{M}_{x \oplus k}) \\ &= \operatorname*{argmin}_k \ n(.) \otimes \mathscr{M}(.)(k) - 2 \sum_{u=1}^{D} L_{cumul}[u] \otimes \tilde{\mathbb{M}}[u](k) \\ &= \operatorname*{argmin}_k \ WHT\Big[WHT(n) \bullet WHT(\mathscr{M}) \\ &\qquad - 2 \sum_{u=1}^{D} WHT(L_{cumul}[u]) \bullet WHT(\tilde{\mathbb{M}}[u])\Big](k).\end{aligned}$$

From this formula, we can carry out an exact template attack using the two algorithms Alg. 1 and Alg. 2 presented below.

A. New and exact profiling Algorithm

The exact model learning algorithm is given in Alg. 1.

Input: Profiling traces set L and the corresponding messages X for the model estimation, according to the known key k^*
Output: The covariance matrix inverse multiplied by the model matrix $\tilde{\mathbb{M}} = \Sigma^{-1} \tilde{M}$

1 **for** $x \in \mathbb{F}_2^n$ **do** // Initialisation
2 $\quad \tilde{m}_x \leftarrow 0$ // Average trace per class
3 $\quad n_x \leftarrow 0$ // Number of traces per class
4 **for** $q \in \{1, \ldots, N\}$ **do** // Accumulation
5 $\quad \tilde{m}_{x_q \oplus k^*} \leftarrow \tilde{m}_{x_q \oplus k^*} + L_q$
6 $\quad n_{x_q \oplus k^*} \leftarrow n_{x_q \oplus k^*} + 1$
7 **for** $x \in \mathbb{F}_2^n$ **do** // Normalisation
8 $\quad \tilde{m}_x \leftarrow \tilde{m}_x / n_x$
9 $\tilde{M} \leftarrow (\tilde{m}_0, \cdots, \tilde{m}_{2^n-1})$
10 $\Sigma \leftarrow \frac{1}{N} LL^{\mathsf{T}} - \frac{1}{2^n} \tilde{M} \tilde{M}^{\mathsf{T}}$
11 $\tilde{\mathbb{M}} \leftarrow \Sigma^{-1} \tilde{M}$
12 **for** $x \in \mathbb{F}_2^n$ **do** // ..... **the** $\tilde{M}_{x \oplus k^*}^{\mathsf{T}} \Sigma^{-1} \tilde{M}_{x \oplus k^*}$ **processing**
13 $\quad \mathscr{M}[x] \leftarrow \tilde{M}^{\mathsf{T}}[x] \, \tilde{\mathbb{M}}[x]$
14 **return** $\tilde{\mathbb{M}}$, $\mathscr{M}$.

Algorithm 1: The new and exact model estimation algorithm.

B. New and exact matching algorithm

Accordingly, the exact matching algorithm, used to extract the most likely key, is given in Alg. 2.

Input: Matching traces set L, the corresponding messages X and the model ($\tilde{\mathbb{M}} = \Sigma^{-1} \tilde{M}$ et $\mathscr{M} = \tilde{M} \Sigma^{-1} \tilde{M}$), obtained by Alg. 1
Output: The guessed key $\hat{k}$ provided by the optimal distinguisher (Proposition 1)

1 **for** $x \in \mathbb{F}_2^n$ **do** // Initialisation
2 $\quad l_{cumul\,x} \leftarrow 0$ // Average trace per class
3 $\quad n_x \leftarrow 0$ // Number of traces per class
4 **for** $q \in \{1, \ldots, N\}$ **do** // Accumulation
5 $\quad l_{cumul\,x_q} \leftarrow l_{cumul\,x_q} + L_q$
6 $\quad n_{x_q} \leftarrow n_{x_q} + 1$
// ..... Matching
7 **return** $\hat{k} = \operatorname{argmin}_k \left(n \otimes \mathscr{M}(k) - 2 \sum_{u=1}^{D} L_{cumul}^{\mathsf{T}}[u] \otimes \tilde{\mathbb{M}}[u](k)\right)$

Algorithm 2: The new and exact spectral computation-based matching algorithm.

C. Discussion (complexity and comparison)

The body of the algorithm Alg. 1, i.e., lines 1 to 8, operates on traces of size D. The same remark applies to the body of Alg. 2, i.e., lines 1 to 6. Consequently, the overall complexity of these parts of the algorithm involves $\mathscr{O}(N \times D)$ additions (the processing of n_x scalars is negligible compared to the processing of traces of D samples, thus we safely ignore them). The complex part of the Alg. 1, namely the line 10, is calculated only once. So the overall complexity of this part is equal to that of the calculation of LL^{T}, which is $\mathscr{O}(D^2 \times N)$

(that of $\tilde{M}\tilde{M}^{\mathsf{T}}$ is $\mathscr{O}(D^2 \times 2^n)$). This yields an overall total of $\mathscr{O}(D^2 \times N)$ multiplications.

Line 11 is also calculated once. The overall complexity of this multiplication is equal to $\mathscr{O}(D^2 \times 2^n)$. To efficiently invert the Σ matrix, one can use Coppersmith Winograd's optimized algorithm [5], which has a complexity of $\mathscr{O}(D^{2.373})$.

Then, lines 12 and 13 of the Alg. 1 consist in calculating $\mathscr{M}$, and have a complexity equal to $\mathscr{O}(2^n \times D)$. The purpose of computing $\tilde{\mathbb{M}}$ at line 11 and $\mathscr{M}$ at line 13 is to avoid calculating them again and again each time an attack is carried out by the Alg. 2.

Regarding the attack proper, described in Alg. 2, the bottleneck is the calculation of line 7. The complexity of $n \otimes \mathscr{M}$ is $\mathscr{O}(n2^n)$, which is negligible compared to that of $\sum_{u=1}^{D} L_{cumul}^{\mathsf{T}}[u] \otimes \tilde{\mathbb{M}}[u]$, which is $\mathscr{O}(Dn2^n)$.

From the above, it has been shown that the complexity of the new and exact profiling and matching algorithms (Alg. 1 and Alg. 2) is equal to that of the two old ones (Alg. 3 and Alg. 4) of [10]). Let us recall that these "old" algorithms boil down asymptotically to using $N/2^n$ instead of n_x, for all values of x, with all the simplifications that are subsequently entailed in the formulas. As a matter of fact, the added computations have negligible complexity compared to the old ones, which leads to a marginal increase in computation time, but an important increase in terms of success ratio, as we will show in the next section.

IV. Results and experimental validation

In order to validate the results, we employed raw traces from the SCA database (ASCAD) of the French National Agency for Information Systems Security (ANSSI) [3]. The target of our attacks is a protected software implementation of AES encryption algorithm (with 128-bit key), running on an ATMEGA-8515 microprocessor, which has an AVR-8 bit architecture. The software aims to protect against first-order SCAs, using a Boolean secret-sharing scheme based on the table recalculation method, although the first two bytes of the AES state are unprotected, to enable for comparison (refer to [3, §2.5.1]).

Typically, the target variable on our tests is the second byte of the state, at the output of the SBox in the first round, i.e., $Z = \mathsf{SBox}(x[2] \oplus k[2])$.

The algorithms are coded using C language, compiled with maximal level of optimization (`-O3` flag) by GNU GCC (version 4.8.1)[1]. The profiling and matching codes are running on a 32-bit Intel i3 personal computer. The Walsh-Hadamard Transform is implemented efficiently by the butterfly algorithm.

A. Comparison

To assess the effectiveness of our improved template attacks, we compared both approaches, namely the old versions (the Alg. 3 and Alg. 4 from [10]), and our improved ones (the Alg. 1 and Alg. 2), in terms of success rates according to the number of traces, for different window sizes D (the number of temporal samples per trace) from 1 to 50. The success rates obtained by using a training set of $512,000$ traces and by averaging over $10,000$ attacks which are represented by the graphs in Figure 1, zooming on an attack on $N < 200$ traces (and in Figure 2 for the same attack on $N < 10,000$ traces).

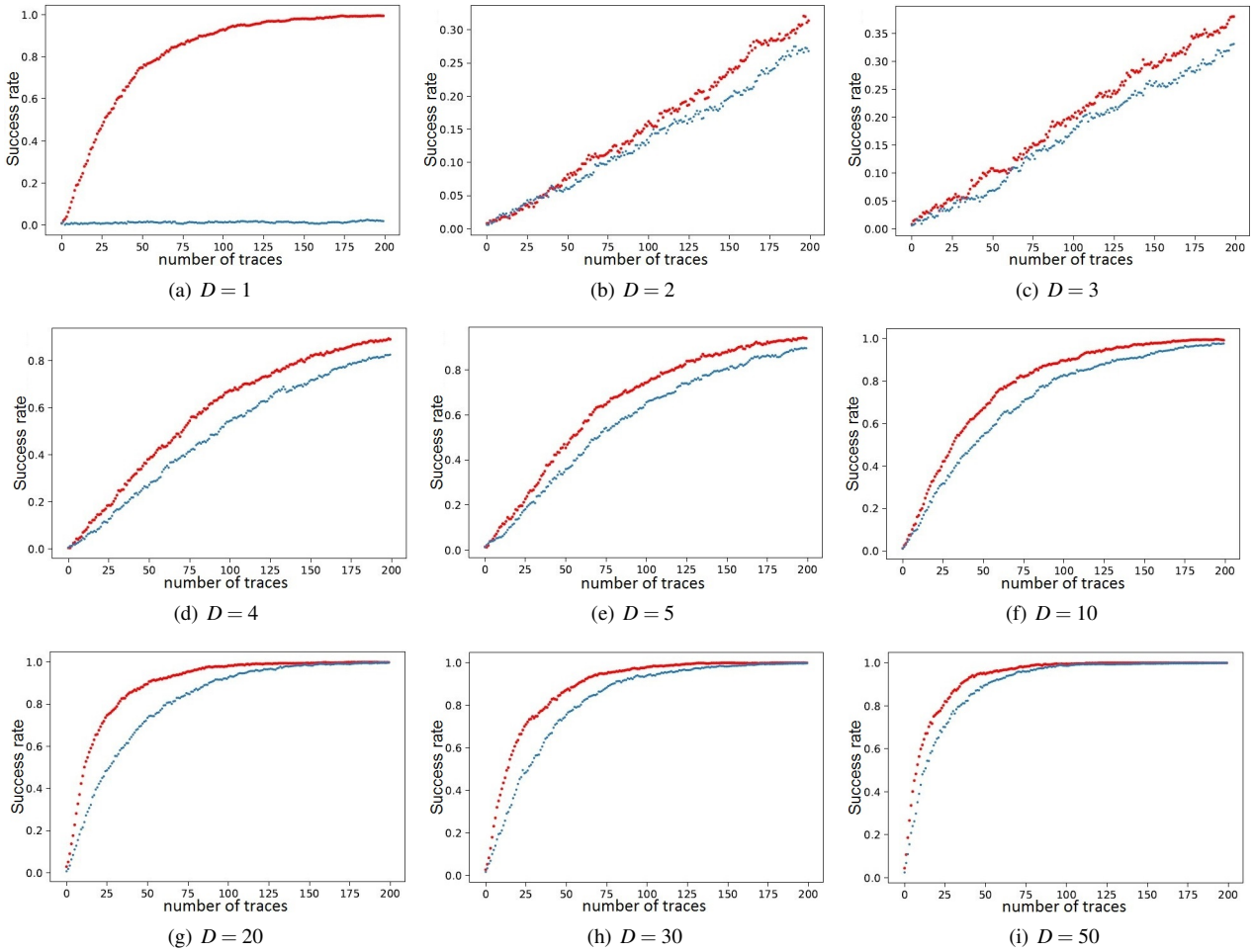
From these graphs, we can clearly see the effectiveness of the proposed improvement, for different values of D. Obviously, this improvement is clearer for small values of D, where the convergence of the success rate to 100% is slower.

Of course, for large values of N (number of traces), our improvement becomes marginal, as the number of traces per class approaches each other (all ending up being equal to $N/2^n$). This is shown in Figure 2.

In practice and interestingly, when $D = 1$ (mono-sample attack), we notice that the success rate with coalescence requires a lot of traces to start differing from 0%. On the other hand, with our exact computations, the success rate in the mono-sample case is increasing fast to 100%. It is even faster than the case of exact computation when $D > 1$. This fact can be explained by the selection of the (unique) point of attack. We selected the point where our attack succeeds the best (i.e., with the best signal-to-noise ratio). So, adding more points comes down to adding (relatively) less information. For clarity, we represented the value of the success rate (after an arbitrary number of 100 matching traces) for attacks with coalescence and with our improvement (exact template computation) in figures 3 and 4. The sample of attack for the graphs in Fig. 2 corresponds to the best sample (at position 350). This sample has a very higher SNR compared to its neighboring samples, which makes it very singular. This accounts for the fact the exact attack at $D = 1$ exhibits a better success rate than for $D > 1$ (until approximately $D = 5$). From $D = 5$ on (i.e., large dimensionality), we recover the fact that the higher D, the higher the success rate (since there is no longer a big discrepancy between the added points, in terms of SNR). Noting that each trace contains 700 samples.

In order to study and compare the computation time for the two approaches (coalescence and our improvement), we run them by varying the window D from 1 to 50, while setting the number of traces to a given value (200 in our case). The execution times obtained are given in Table I and plotted in Figure 5.

[1]The source code is freely available at https://github.com/<anonymous>/Exact_Template_Attacks_With_Spectral_Computation/tree/master (the correct URL is hidden owing to the blind review process).

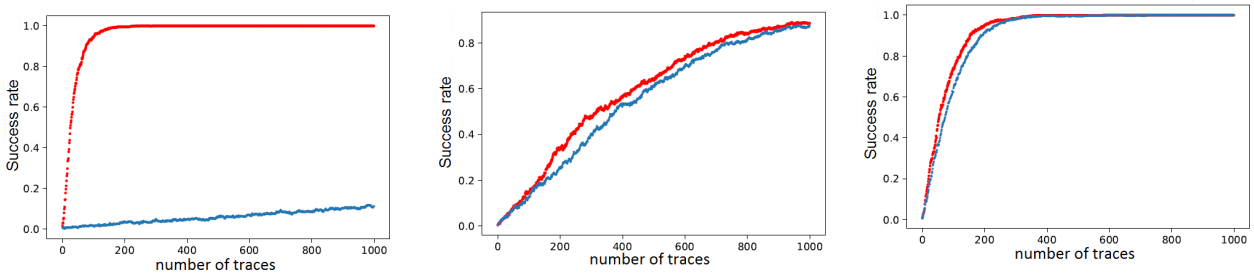


(a) $D = 1$ (b) $D = 2$ (c) $D = 3$

(d) $D = 4$ (e) $D = 5$ (f) $D = 10$

(g) $D = 20$ (h) $D = 30$ (i) $D = 50$

Figure 1. Success rate according to the number of traces (until $N = 200$), for different values of D, with: (•): coalescence and (•): our improvement



(a) $D = 1$ (b) $D = 2$ (c) $D = 5$

Figure 2. Success rate according to the number of traces (until $N = 1000$), for different values of D, with: (•): coalescence and (•): our improvement

Table I
COMPUTATION TIME (SECONDS) ACCORDING TO THE TRACES DIMENSIONALITY D.

	1	**2**	**3**	**4**	**5**	**10**	**20**	**30**	**50**
Coalescence	26.649	29.658	31.980	35.021	37.158	55.814	75.392	105.150	159.486
Improvement	31.365	33.128	35.801	37.964	39.754	54.016	82.036	106.676	170.333

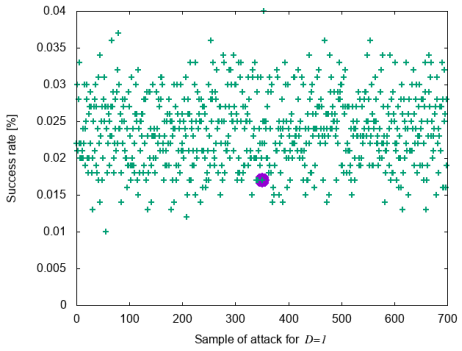


Figure 3. Success rate after 100 traces for $D = 1$ in the (legacy) coalescence approach, as a function of the sample chosen for the attack.

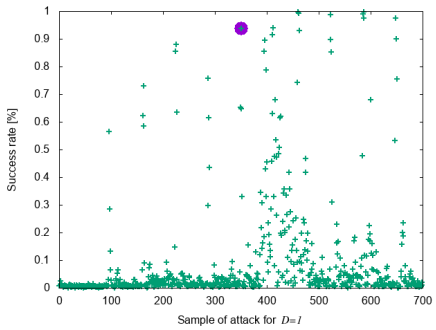


Figure 4. Success rate after 100 traces for $D = 1$ in our new exact approach, as a function of the sample chosen for the attack. It shows that the selection of the sample 350 is definitely singular, as the attack is significantly better at this sample than at others, even its neighbors.

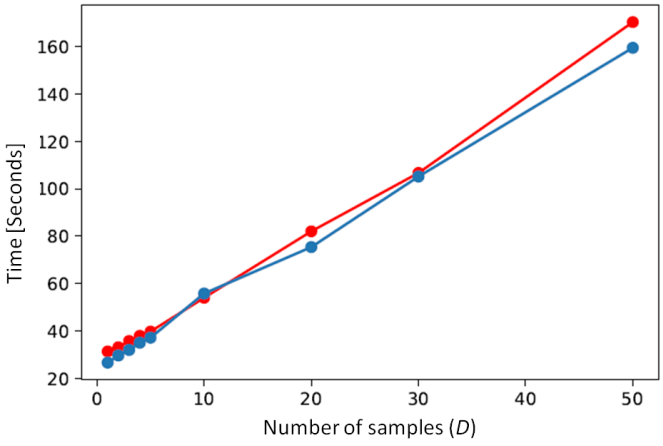


Figure 5. Computation time according to D with: (●): coalescence and (●): our improvement

From the above experimental results, we can see that our approach offers a considerable improvement in terms of success rate at the expense of a marginal delay in terms of computation time. This shows the importance of this improvement, as in general an improvement of say 10% in data (number of traces complexity is much more valued than an improvement of 10% in computational complexity).

V. Conclusion and Perspectives

In this paper, we described a new improvement in template attacks' success rate and computational speed. We took advantage of the properties of the spectral approach and were inspired by the coalescence principle, while mathematically demonstrating the optimality of the proposed improvement.

Furthermore, we presented a validation consisting of practical results obtained in the form of graphs, clearly showing the improvement in success rate compared with the previous version of coalescence-based template attacks. This considerable gain in success rate comes at the expense of a marginal loss in computation time, which is explained in terms of complexity.

Besides, the application of the coalescence principle was introduced in attacks based on linear regression (as early as the first LRA paper [11]). Recall that it causes accuracy errors, with template attacks, due to class imbalance. So, it seems that our improvement of template attacks could be extended to the linear regression-based one, by taking into account the class sizes and getting inspiration from the spectral computation [9]. This work is planned as a future improvement, as thus left as a perspective.

References

[1] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Optimal side-channel attacks for multivariate leakages and multiple models. *J. Cryptographic Engineering*, 7(4):331–341, 2017.

[2] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[3] Prouff Emmanuel, Strullu Remi, Benadjila Ryad, Cagli Eleonora, and Dumas Cecile. Study of deep learning techniques for side-channel analysis and introduction to ascad database. *CoRR*, pages 1–45, 2018.

[4] A. Samad Hedayat and Walter D. Wallis. Hadamard matrices and their applications. *Ann. Statist.*, 6(6):1184–1238, 11 1978.

[5] Howard Karloff. *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*. ACM, 2012.

[6] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO'99*, pages 388–397. Springer-Verlag, 1999.

[7] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[8] Maamar Ouladj and Sylvain Guilley. *Side-Channel Analysis of Embedded Systems*. Springer, 2021. ISBN: 978-3-030-77221-5.

[9] Maamar Ouladj, Sylvain Guilley, and Emmanuel Prouff. On the implementation efficiency of linear regression-based side-channel attacks. In *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, October 5-7, 2020, Proceedings (LNCS 12244)*, pages 147–172, 2020.

[10] Maamar Ouladj, Nadia El Mrabet, Sylvain Guilley, Philippe Guillot, and Gilles Millérioux. On the power of template attacks in highly multivariate context. *Journal of Cryptographic Engineering - JCEN*, 2020.

[11] Werner Schindler. On the optimization of side-channel attacks by advanced stochastic methods. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings*, volume 3386 of *Lecture Notes in Computer Science*, pages 85–103. Springer, 2005.