Time-Frequency Analysis for Second-Order Attacks

Pierre Belgarric[1,3], Shivam Bhasin[1], Nicolas Bruneau[1,4(✉)], Jean-Luc Danger[1,5], Nicolas Debande[1,6], Sylvain Guilley[1,5], Annelie Heuser[1], Zakaria Najm[1], and Olivier Rioul[2,7]

[1] TELECOM-ParisTech, Crypto Group, Paris, France
nicolas.bruneau@telecom-paristech.fr
[2] TELECOM-ParisTech, Digital Communications Group, Paris, France
[3] Orange Labs, Applied Cryptography Group, Issy-les-Moulineaux, France
[4] STMicroelectronics, AST Division, Rousset, France
[5] Secure-IC S.A.S., Rennes, France
[6] SERMA ITSEF, Pessac, France
[7] École Polytechnique, Palaiseau, France

Abstract. Second-order side-channel attacks are used to break first-order masking protections. A practical reason which often limits the efficiency of second-order attacks is the temporal localisation of the leaking samples. Several pairs of leakage samples must be combined which means high computational power. For second-order attacks, the computational complexity is quadratic. At CHES '04, Waddle and Wagner introduced attacks with complexity $\mathcal{O}(n \log_2 n)$ on traces collected from a *hardware* cryptographic implementation, where n is the window size, by working on traces auto-correlation. Nonetheless, the two samples must belong to the same window which is (normally) not the case for *software* implementations. In this article, we introduce preprocessing tools that improve the efficiency of bi-variate attacks (while keeping a complexity of $\mathcal{O}(n \log_2 n)$), even if the two samples that leak are far away one from the other (as in software). We put forward two main improvements. Firstly, we introduce a method to avoid losing the phase information. Next, we empirically notice that keeping the analysis in the frequency domain can be beneficial for the attack. We apply these attacks in practice on real measurements, publicly available under the DPA Contest v4, to evaluate the proposed techniques. An attack using a window as large as 4000 points is able to reveal the key in only 3000 traces.

Keywords: Bi-variate attacks · Zero-offset 2O-CPA · Discrete Hartley transform · Leakage in phase

This work is partially funded by ANR/JST project SPACES: https://spaces.enst.fr/.
Nicolas Debande – This work has been conducted while Nicolas Debande was with Morpho, Osny, France.
Annelie Heuser – Google European fellow in the field of privacy and is partially founded by this fellowship.

A. Francillon and P. Rohatgi (Eds.): CARDIS 2013, LNCS 8419, pp. 108–122, 2014.
DOI: 10.1007/978-3-319-08302-5_8,

1 Introduction

Side-Channel Attacks (SCA [1]) and corresponding protection techniques have been a hot research topic for over a decade now. Data masking [6] is one of few popular side-channel countermeasures, which motivates thorough investigations of higher-order SCA as e.g., in [12,15]. The following study deals mainly with *second-order SCA* which is used to break a first-order masking countermeasure. A particular case of second-order SCA is when the two shares used by the masking scheme are processed or leak simultaneously. In this case, Waddle and Wagner introduced an attack at CHES '04 [15], which consists in raising the traces to the power two. Such an attack, a so-called *zero-offset SCA*, is commonly used against hardware or parallel implementations. However, for software implementations, the two shares naturally leak at different dates or time samples. The second-order attacks which combine two different time samples are termed *bi-variate SCA*. The two different leakage samples are referred to as $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$ in the following. Despite bi-variate attacks may be powerful, a practical implementation might need a large amount of effort from the part of the attacker. The main problem of bi-variate attacks is to find the exact temporal localization (t_0, t_1) corresponding to leakages $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$. Incidentally, depending on the implementation, there might exist several such pairs.

To avoid finding the pair (t_0, t_1) explicitly, Waddle and Wagner introduced a method called FFT-2DPA, which only requires to find a window in which both leakages are included. More precisely, the attacker computes the auto-correlation on this window, which combines the two leakages $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$ multiplicatively. Thus, it is possible to utilize a regular zero-offset SCA on the auto-correlation trace. The authors of [15] suggest, to compute the auto-correlation as the inverse Fourier transform of the square modulus of the trace Fourier transform of the window of size n. This way, the preprocessing time has $\mathcal{O}(n \log_2 n)$ complexity, which is sub-quadratic.

Another category of second-order SCA are collision-based attacks. A particular case where *collision attacks* are efficient, is when the same mask is reused for each substitution box (S-box) of the crypto-algorithm. There exist two sub-categories of collision attacks: *correlation-collision* attacks and *collision-correlation* attacks. If the unmasked input of the S-box is biased, then correlation-collision attacks (see for instance [10]) can be applied. Otherwise, collision-correlation attacks [2] are more suitable. However, when the masking scheme does not reuse one mask to protect multiple unrelated sensitive variables, collisions attacks in general are not appropriate.

Summing up, apart from FFT-2DPA, bi-variate attacks usually require the knowledge of the samples $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$. If the leakage models $\mathcal{M}_0$ and $\mathcal{M}_1$ corresponding to the leakages $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$ are known, then the optimal strategy consists in combining them with a *centered product* [12]. We denote this attack as "2O-CPA". Note that, if the leakage can be approximated, then a *linear-regression* approach can mitigate the absence of accurate knowledge of the models $\mathcal{M}_0$ and $\mathcal{M}_1$ [4].

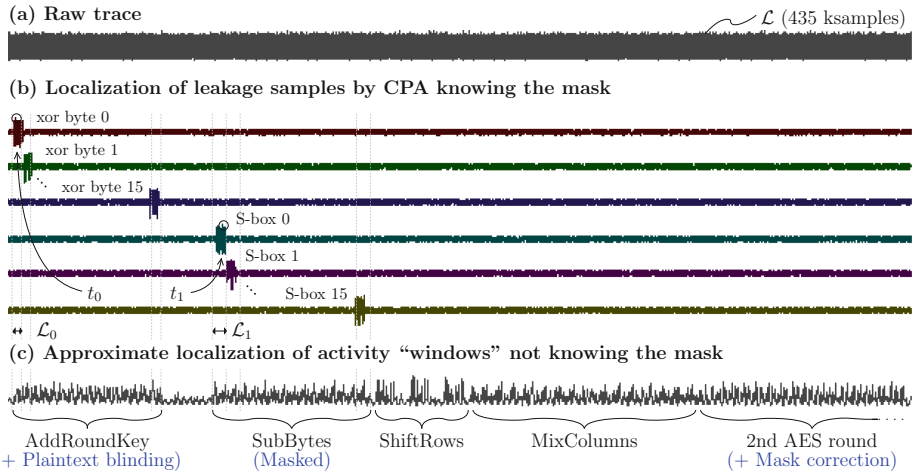

Fig. 1. Analyses on traces collected from the first round of a masked AES in software

Figure 1(a) shows the trace $\mathcal{L}$ of the beginning of an AES encryption on a smartcard. We see about 3100 clock cycles (435000 time samples). It is not possible to distinguish individual operations by visual inspection of the trace.

One way to identify the precise timing of individual operations, consists in using a clone device, where the masks can be set to zero or are known. In this case, several monovariate CPAs [1] can be computed to disclose the exact sample(s) in which each operation leaks as illustrated in Fig. 1(b). Such an analysis seems impossible, without the access to a clone device. However, without any information on the masks, an attacker can compute the several moments or filter the traces. Figure 1(c) plots the variance of the average of the traces computed over each clock cycle. It clearly reveals the structure of one AES round: AddRoundKey (16 identical operations), followed by SubBytes (16 identical operations), ShiftRows (3 identical operations on rows — indeed, the first row is unchanged by ShiftRows), MixColumns (4 identical operations on columns), and AddRoundKey again (corresponding to the second round). The notations in Fig. 1 are as follows: $\mathcal{L}_0$ and $\mathcal{L}_1$ ($\mathcal{L}_0, \mathcal{L}_1 \subset \mathcal{L}$; $\mathcal{L}(t_0) \in \mathcal{L}_0$ and $\mathcal{L}(t_1) \in \mathcal{L}_1$) are the windows in which the shares #0 and #1 are expected to leak (they correspond to the so-called *educated guesses* coined by Oswald et al. [11]); n_0 and n_1 are the width of windows $\mathcal{L}_0$ and $\mathcal{L}_1$, in terms of sample count. For the sake of simplicity, we assume $n_0 = n_1 = n$. Typically, $\mathcal{L}$ has few hundreds of thousand samples (e.g., 435000 in Fig. 1), whereas n_0 and n_1 may vary from a few hundreds to a couple of thousands.

Our Contributions. In this paper, we propose five practical methods to make 2O-CPA attacks feasible on first-order masking schemes. All five proposed methods are generic in nature and need no knowledge of leaking time samples. The common feature of our attacks is to turn a bivariate leakage into a monovariate

leakage (thanks to a combination that creates a sum of weighted products), that can be exploited by a classical zero-offset second-order attack. We base ourselves in the role of an attacker, who has a rough estimate of the zones in $\mathcal{L}$ where the leakages t_0 and t_1 are likely to be situated (that we call time intervals $\mathcal{L}_0$ and $\mathcal{L}_1$). In particular, our preprocessing methods convert two leakage windows of size n into a new window of size $2n$ or n, depending on the applied technique. Remarkably, these operations remain in complexity $\mathcal{O}(n \log_2 n)$, i.e., sub-quadratic. We show that our methods allow faster attacks (in terms of number of queries for the 2O-CPA to reach 80 % success rate) than the generalization of FFT-2DPA on two windows. This gain comes from two major factors:

1. The *phase information* is kept intact, and
2. The operation is performed in *frequency domain*.

As shown later, the leakage has a specific signature in terms of waveform shape, and in our implementation, there are multiple occurrences in time of the leakage. The representation in the frequency domain allows to regroup all these leakages, that combine constructively because they share the same waveform. Thus, the gain in terms of success rate is evident, since the signal is magnified at constant noise. Besides, from a computational point of view, the attack still stays on a linear number of points (n or $2n$).

Outline of the Paper. The rest of the paper is organized as follows. Preliminaries of tools related to time-frequency conversion are introduced in Sect. 2. Section 3 describes the five proposed preprocessing techniques, using time-frequency conversion tools. The attacks are then applied on a real masking implementation running on an 8-bit AVR smartcard (in Sect. 4). Section 5 provides further insights into the proposed attacks and their standing as compared to the state-of-the-art. Finally, conclusions and perspectives are drawn in Sect. 6.

2 Tools for Time-Frequency Analysis

This section provides a short background on common tools used in time-frequency analysis, which are then used in the proposed attacks in Sect. 3.

2.1 Discrete Fourier Transform

Definition 1 (DFT). *The* discrete Fourier transform *of a sequence* $Y \in \mathbb{R}^n$ *is another sequence* $\mathsf{DFT}[Y] \in \mathbb{C}^n$ *such as*

$$\mathsf{DFT}[Y](f) = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} Y(t) \cdot \exp(-2\imath\pi f t/n),$$

where $\imath$ *is one of the (square) roots of* 1 *in* $\mathbb{C}$ *that is different from* ± 1.

Property 1 (Inverse DFT). The DFT can be inversed with the inverse DFT such that $\mathsf{IDFT}\left[\mathsf{DFT}\left[Y\right]\right] = Y$, where $\mathsf{IDFT}\left[Z\right](t) = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} Z(f) \cdot \exp\left(+2\pi\imath ft/n\right)$.

Definition 2 (Cross-correlation). *The* (circular) cross-correlation *of two discrete sequences X and Y of n samples is defined by*

$$(X \star Y)(t) = \sum_{t'=0}^{n-1} X(t') \cdot Y(t' + t \mod n).$$

Theorem 1 (Cross-correlation theorem). *Again let X and Y be two discrete sequences of n samples in time domain, then*

$$(X \star Y)(t) = \sqrt{n} \cdot \mathsf{IDFT}\left[\overline{\mathsf{DFT}\left[X\right]} \cdot \mathsf{DFT}\left[Y\right]\right],$$

where $\overline{\cdot}$ denotes complex conjugation.

2.2 Discrete Hartley Transform

The application of a DFT on a sequence of real numbers results in a sequence of complex numbers. The discrete Hartley transform [7] (DHT) was proposed as a real-valued alternative to the DFT as DHT multiplies each real input by $\cos + \sin$ instead of $\cos - \imath \sin$ as in DFT:

Definition 3 (DHT). *The* discrete Hartley transform *of a sequence $Y \in \mathbb{R}^n$ is another sequence $\mathsf{DHT}\left[Y\right] \in \mathbb{R}^n$ such as:*

$$\mathsf{DHT}\left[Y\right](f) = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} Y(t) \cdot \left(\cos\left(2\pi\, ft/n\right) + \sin\left(2\pi\, ft/n\right)\right).$$

Property 2 (Link between Fourier and Hartley transforms). The DHT of the temporal signal Y can be obtained from the DFT by:

$$\mathsf{DHT}\left[Y\right](f) = \Re\mathrm{e}\,\mathsf{DFT}\left[Y\right](f) - \Im\mathsf{DFT}\left[Y\right](f).$$

Reciprocally, the DFT of the signal Y can be computed from the DHT with the formula:

$$\mathsf{DFT}\left[Y\right](f) =$$
$$\tfrac{1}{2}\left(\mathsf{DHT}\left[Y\right](f) + \mathsf{DHT}\left[Y\right](-f)\right) - \tfrac{\imath}{2}\left(\mathsf{DHT}\left[Y\right](f) - \mathsf{DHT}\left[Y\right](-f)\right).$$

Property 3 (DHT Involution). The DHT is its own inverse;
$\forall Y \in \mathbb{R}^n, \mathsf{DHT}\left[\mathsf{DHT}\left[Y\right]\right] = Y$. The proof is given in [7].

As such, the DHT avoids two computationally undesirable characteristics of the DFT:

1. the inverse DHT is identical with the direct transform — it is not necessary to keep track of $+\imath$ and $-\imath$ versions;
2. more importantly, the DHT has real rather than complex values. As a consequence, in a 2O-CPA, the computation of the correlation coefficient can be done in the frequency spectrum without any loss of information.

2.3 Fast Fourier Transform

The DFT (resp. IDFT) is directly obtainable from the FFT (resp. IFFT), that runs in $\mathcal{O}(n \log_2 n)$ complexity [5]. The computational complexity of DHT is also $\mathcal{O}(n \log_2 n)$, as it is simply obtained as the difference between the real and imaginary parts of the FFT.

3 New Second-Order Attacks with Time-Frequency Preprocessing

3.1 Why Do We Need New Attacks?

In first-order masking implementations, it is expected that each mask is reused (at least twice). Unfortunately, as shown in Fig. 1, the distance between two leakages using the same mask can be about 100000 samples. Therefore, the attacker, in practice, needs *two distinct* windows where the mask is reused, assuming for the sake of simplicity both of size n. Since the exact temporal localization of t_0 and t_1 corresponding to the leakages $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$ is unknown to the attacker, he would have to mount $\binom{n}{2}$ 2O-CPAs, resulting in $\mathcal{O}(n^2)$ complexity, which can become impractical for large n.

Another method would be to apply the approach of FFT-2DPA. However, one window in which $\mathcal{L}(t_0)$ and $\mathcal{L}(t_1)$ are included would be too large (e.g., 100000 time samples), therefore to overcome this problem we straightforwardly extend the idea of Waddle and Wagner to the case of two distinct windows $\mathcal{L}_0$ and $\mathcal{L}_1$. In particular, we consider two different approaches to treat $\mathcal{L}_0$ and $\mathcal{L}_1$. First, we use the concatenation:

Definition 4 (auto-corr). *Let us denote $\mathcal{L}_{01}$ as the concatenation in time of $\mathcal{L}_0$ and $\mathcal{L}_1$. Then* auto-corr $= (\mathcal{L}_{01} \star \mathcal{L}_{01}) = \mathsf{IDFT}\left[|\mathsf{DFT}\left[\mathcal{L}_{01}\right]|^2\right]$.

Second, if the size of the windows, $\mathcal{L}_0$ and $\mathcal{L}_1$ have equal width (i.e., $n_0 = n_1 = n$), the attacker can compute cross-correlation between $\mathcal{L}_0$ and $\mathcal{L}_1$, which we call x-corr.

Definition 5 (x-corr). $\textit{x-corr} = (\mathcal{L}_0 \star \mathcal{L}_1) = \mathsf{IDFT}\left[\overline{\mathsf{DFT}\left[\mathcal{L}_0\right]} \cdot \mathsf{DFT}\left[\mathcal{L}_1\right]\right]$.

Interestingly, both auto-corr and x-corr can be computed in a complexity $\mathcal{O}(n \log_2 n)$, owing to the cross-correlation Theorem 1. Moreover, the *preprocessing* stage turns a *bi-variate* leakage into a *uni-variate* leakage. Indeed, the expressions auto-corr(t) and x-corr(t) contain the product $\mathcal{L}(t_0) \cdot \mathcal{L}(t_1)$, which is exploited by a 2O-CPA. So, the optimal prediction function is the same as in any bi-variate 2O-CPA. Thus, after the preprocessing with either auto-corr or x-corr, an attacker can simply perform a zero-offset SCA on the resultant trace to find the secret key.

However, we noticed two essential drawbacks when using the straightforward extension from Waddle and Wagner:

- First of all, as the DFT of the signals are processed via a *modulus* (See e.g., Definition 4), the *phase information is lost.*

– Second, returning in the timing domain is less efficient than staying in the frequency domain: indeed, as will be seen with on our practical examples (Sect. 4), the leaks in software usually feature *many peaks* in time domain, that nonetheless have a *common signature* in frequency domain.

3.2 New Attacks in Frequency Domain

Based on the previous definitions and observations, we introduce 5 new preprocessing methods, which intend to capture the leakage directly in frequency domain without transferring it back into time domain. Similar as for auto-corr and x-corr, we divide methods into two distinct classes. The first class consists of so-called "*one window*" methods, which utilizes the concatenated window $\mathcal{L}_{01}$ from two individual windows $\mathcal{L}_0$ and $\mathcal{L}_1$ resulting in an output of $2n$. The second class of methods ("*two windows*" methods) are capable to combine two windows of size n into a single window also of size n.

As analysis methods we use DFT and DHT (see Definition 3 in Sect. 2). The four resultant preprocessing techniques are summarized in Table 1.

Table 1. Variants of considered preprocessing attacks

Function \ ⟨name⟩	DFT [·]	DHT [·]
concat-⟨name⟩(f)	$\lvert\mathsf{DFT}\,[\mathcal{L}_{01}]\rvert^2$	$\mathsf{DHT}\,[\mathcal{L}_{01}]^2$
window-⟨name⟩(f)	$\lvert\overline{\mathsf{DFT}\,[\mathcal{L}_0]}\cdot\mathsf{DFT}\,[\mathcal{L}_1]\rvert$	$\mathsf{DHT}\,[\mathcal{L}_0]\cdot\mathsf{DHT}\,[\mathcal{L}_1]$

In order to reveal the secret key an attacker applies a zero-offset CPA on the output of these preprocessing techniques and the optimal prediction function $\mathcal{M}_{01}$, which we specify in Sect. 4.

Additionally as a "heuristic" method, we consider the max-corr attack to cope with a complex 2O-CPA (i.e., $\rho(\cdot,\cdot) \in \mathbb{C}$). More precisely,

$$\mathsf{max\text{-}corr} = \max(|\rho(\Re\mathrm{e}(\mathsf{DFT}\,[\mathcal{L}_{01}]), \mathcal{M}_{01})|, |\rho(\Im\mathrm{m}(\mathsf{DFT}\,[\mathcal{L}_{01}]), \mathcal{M}_{01})|).$$

Beware that the suffix "corr" in "max-corr" refers to the Pearson correlation coefficient "ρ" of the high-order CPA, and not to any auto- or cross-correlation.

Concluding, in total we proposed five new methods of the same complexity $\mathcal{O}(n \log_2 n)$ to mount second-order attacks on a first-order masking implementation. The described methods are applied on a real masked AES implementation running on a smartcard in the following section.

4 Experimental Validation

4.1 Software Implementation of the Protected AES

To test our methods, we use the publicly available traces of DPA contest v4 [14], which uses a low-cost masking protection applied on AES, called Rotating S-box Masking (RSM). RSM is a first-order countermeasure in which the S-boxes

$\mathbb{F}_{2^8} \rightarrow \mathbb{F}_{2^8}$ are (statically) precomputed. The same mask is XORed to one plaintext byte (T) and to some S-box output (corresponding to another plaintext byte T'). In this case, collision attacks might be applicable to the design. However, we considered an attack based on the combination of two "heterogeneous" leakage models. The applicable (*centered*) leakage models are given by: $\mathcal{M}_0 = w_H(T \oplus M) - 4$ and $\mathcal{M}_1 = w_H(\texttt{Sbox}[T' \oplus K] \oplus M) - 4$, where T, T', K are respectively two bytes of the plaintext and one byte of the key, and where $w_H(\cdot)$ is the Hamming weight function. Thus, the prediction function $\mathcal{M}_{01}$ for all our preprocessing methods is given by $\mathcal{M}_{01} = \mathbb{E}[(\mathcal{M}_0 \cdot \mathcal{M}_1)|T, T', K]$.

4.2 Leakage Detection

In the following we ensure that both leakage models $\mathcal{M}_0$ and $\mathcal{M}_1$ are suitable for our evaluation. We first perform a CPA on the traces, assuming the mask to be a known quantity in order to identify the most leaking points and to verify our assumed leakage models. The prediction functions knowing the mask are simply: $\mathcal{M}_0^m = \mathbb{E}[\mathcal{M}_0|T, M]$ and $\mathcal{M}_1^m = \mathbb{E}[\mathcal{M}_1|T', K, M]$.

Figure 2(a) shows the correlation between the leakage $\mathcal{L}_0$ and the model $\mathcal{M}_0^m$ using 10000 measurements. We additionally marked the time instants when the correct key takes the highest correlation (i.e., $k^* = \max_k \rho(\mathcal{L}_1, \mathcal{M}_1^m)$), which amounts in 433 time instants over the window of 6000 points. Figure 2(b) shows the correlations using $\mathcal{M}_1^m$, where in 94 time instants the correct key takes the highest correlation, moreover, these instants are less spread than for the XOR operation. Further, Fig. 3 shows the mean consumption of each class of the highest correlation peak around the time instant ≈3000. One can clearly detect that the classification according to $\mathcal{M}_0^m$ (resp. $\mathcal{M}_1^m$) is reasonable. We therefore maintain our models $\mathcal{M}_0$ and $\mathcal{M}_1$ capturing the XOR and the $\texttt{Sbox}[\cdot]$ operation. The average number of traces to break the key using $\mathcal{M}_1^m$ is about 15 (*very low!*) for a success rate $\geqslant$80 %, as can be seen in Fig. 4(a).

4.3 Empirical Evaluation

First of all, we confirm that a direct application of a 1O-CPA (Brier et al. [1]) using model $\mathcal{M}_0$ or $\mathcal{M}_1$ on the whole trace $\mathcal{L}$ does not allow to retrieve any key byte using 100000 traces. No preprocessing was applied on the traces before the attack. Then, we applied a bi-variate 2O-CPA by multiplying the two most leaking samples for models $\mathcal{M}_0^m$ and $\mathcal{M}_1^m$. The success rate is given in Fig. 4(b). About 300 traces are sufficient to break the key with probability $\geqslant$80 %.

For our empirical evaluation we choose 3 different sets of window sizes n: small $n = \{50, 200\}$, medium $n = \{500, 2000\}$, large $n = \{4000, 6000\}$. So, auto-corr, concat-dft & concat-dht are calculated on a window of size $2n$, whereas x-corr, window-dft & window-dht utilize two windows each with size n. Since only a fixed number of measurement traces (100000) are provided by the DPA contest v4, we were restricted in the number of retries. More precisely, for small windows we computed the success rate using up to 2000 traces and we were therefore able

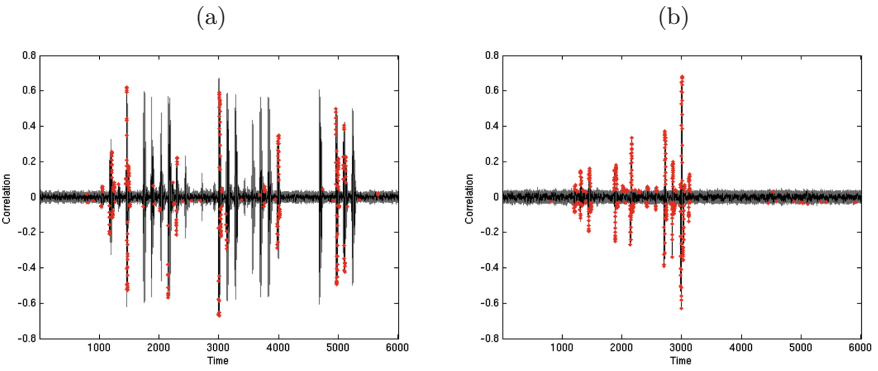

Fig. 2. Correlation $\rho(\mathcal{L}_0, \mathcal{M}_0^m)/\rho(\mathcal{L}_1, \mathcal{M}_1^m)$ knowing the mask M; correlation of k^* is displayed in black; time instants when $k^* = \max_k \rho(\mathcal{L}_0, \mathcal{M}_0^m)/k^* = \max_k \rho(\mathcal{L}_1, \mathcal{M}_1^m)$ are marked with a red diamond (Color figure online)

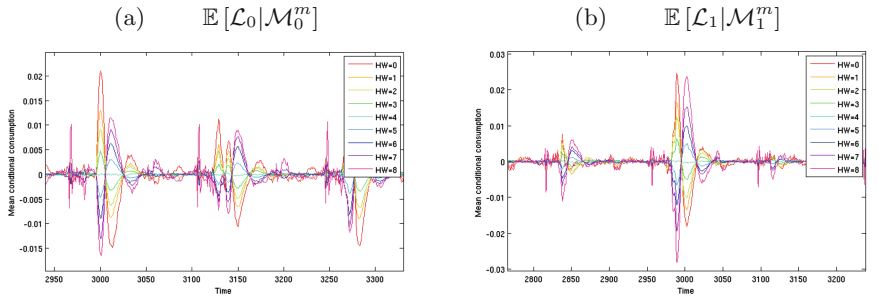

Fig. 3. Mean consumption conditioned by each leakage model class $\mathcal{M}_0^m$ and $\mathcal{M}_1^m$

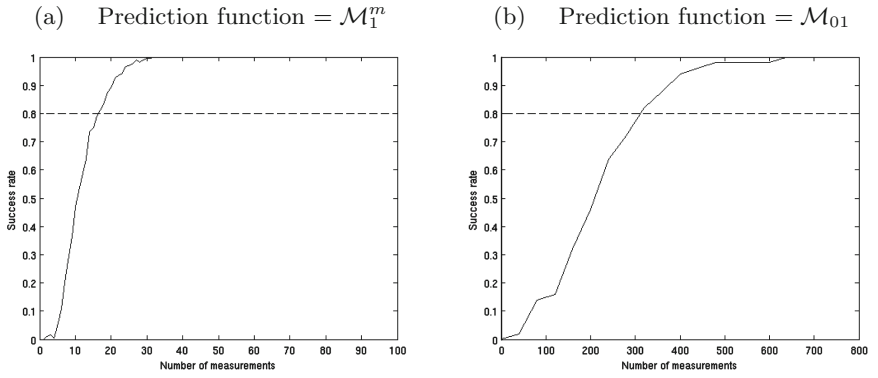

Fig. 4. Success rate of (a) univariate CPA attack on $\mathcal{L}(t_1)$ knowing the mask and (b) bi-variate 2O-CPA attack on $\mathcal{L}(t_0) \cdot \mathcal{L}(t_1)$ knowing (t_0, t_1) but ignoring the mask

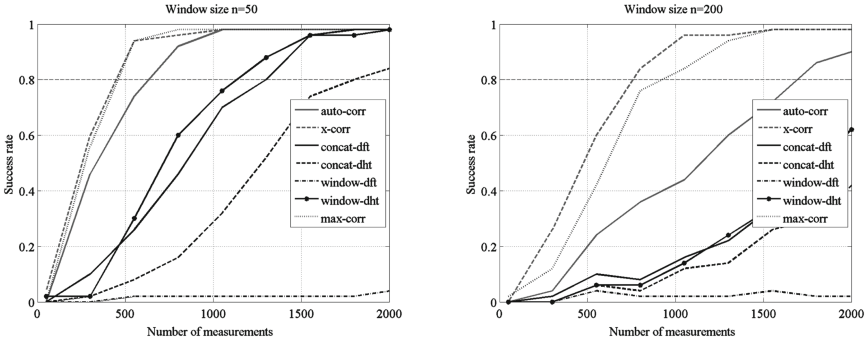

Fig. 5. Success rate when using a small window size

to compute $^{100000}/_{2000} = 50$ retries, accordingly, for medium windows 25 retries, and for large windows 10 retries were possible.

The success rate for a window of smaller size ($n = 50$ and $n = 200$) is shown in Fig. 5. In both cases, auto-corr and x-corr are the most efficient preprocessing methods, followed by the window-dht, concat-dft, and concat-dht, whereas window-dft is not able to retrieve the correct key. This confirms that the preprocessing of Waddle and Wagner is relevant when the time instants of the leakages are well known a priori. However, we also note that for such small windows, an exhaustive search of the interesting (t_0, t_1) is not deterrent (computationally speaking), and would yield better success rates (recall Fig. 4(b)).

The efficiency of the attacks is changed when using a window of medium size (see Fig. 6). The usage of x-corr seems only reasonable when the window size is sufficiently small, whereas the efficiency of window-dft and concat-dht increases when provided with more time instants. Interestingly, one can observe that window-dht is more efficient when using a window size of 500 as x-corr with smaller window size. This is an illustration that the attack manages to properly combine constructively the plurality of leakage instants in the trace (recall the multiple leakage peaks in Fig. 2(a) and (b)).

When increasing the window size up to $n = 4000$ and $n = 6000$ the difference between window-dht, concat-dht, and concat-dft becomes greater. Remarkably, even for large window sizes (two windows with each 6000 time instants), window-dht is still able to efficiently reveal the secret key. It is about equivalent in terms of efficiency with max-corr. Thus, this confirms that *attacks remain very practical, even though the attacker does not have a precise idea about the leakage location.*

From Table 2, we can deduce that when the attacker knows the leakage samples, i.e., a small window size, x-corr is the best attack. Moving from small to medium windows, window-dht proves to be the best attack. Finally, max-corr seems to be the best attack for large window size. This means that max-corr is well suited for practical cases because only a minimum assumption on the knowledge of leakage samples is required, thus, the attacker is able to choose a large window. As already underlined, another noteworthy observation from Table 2 is

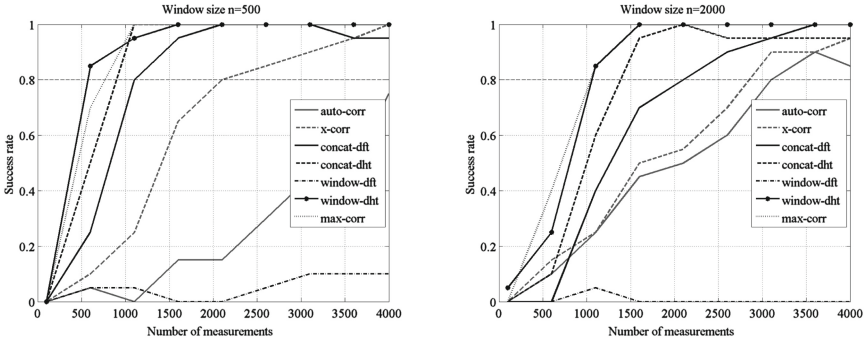

Fig. 6. Success rate when using a medium window size

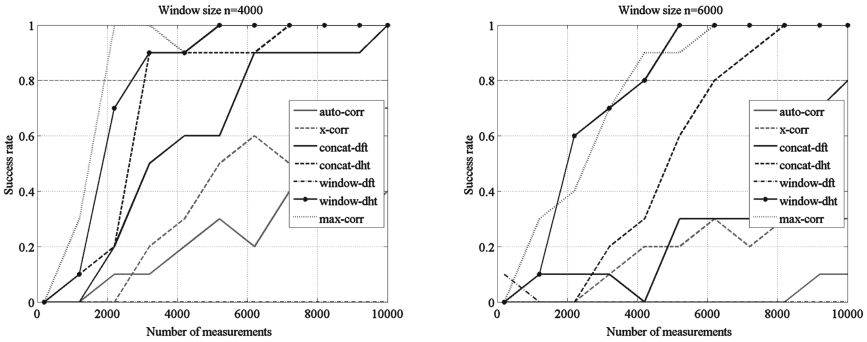

Fig. 7. Success rate when using a large window size

that, x-corr takes more traces to disclose the key for a window of 200 points as compared to window-dht for a window of 500 points.

5 Discussion

5.1 Benefits of the Proposed Attacks

Preprocessing Speed-up. Turning bi-variate into mono-variate leakage is actually a matter of trade-off:

- the computational power is lowered while exploiting the traces (because the research of (t_0, t_1) is skipped);
- at the expense of a greater noise in the estimation of the distinguisher (hence more traces to guess the key), due to the inaccurate location of the leakages in the window(s).

The use of our methods can be justified for software traces, that can be so long (millions of samples) that a complexity in $\mathcal{O}(n^2)$ is prohibitory. For instance,

Table 2. Comparison of performance of proposed methods against attack efficiency.

Window size	Best attack	Number of traces for SR $\geqslant 0.8$
50	x-corr	450
200	x-corr	750
500	window-dht	550
2000	window-dht max-corr	550
4000	max-corr	1950
6000	max-corr	3000

with window size $n = 6000$, the complexity of our preprocessing (in terms of "multiplications" count) is roughly $n \log_2 n \approx 75300$ or 0.0753×10^6, whereas an exhaustive search of pairs (t_0, t_1) requires $\frac{n(n-1)}{2} \approx 18 \times 10^6$ tries. So our attack method is very light in computation time. Now, in terms of number of traces to break the key, our method requires about 3000 traces instead of 300 knowing the most leaking samples, which remains reasonable.

Resilience to Traces Desynchronization. Our techniques can withstand a global desynchronization in the acquisition of the traces. It can happen that the traces are offset one w.r.t. the others, due to the lack of a reliable synchronization signal. It is already known that DFT based techniques (if the phase is ignored) can work even in this case [8]. (We do not consider here countermeasures like dummy cycles addition [3].) So concat-dft, window-dft and max-corr resist traces disalignment.

5.2 Explanation of the Results: Why are Attacks in Frequency Domain More Efficient when the Window Width is Large?

When the correlation is computed on auto-corr or x-corr signals, i.e., in the time domain, the leakage $\mathcal{L}(t_0) \cdot \mathcal{L}(t_1)$ is "dissimulated" into the numerous other terms $\mathcal{L}(t) \cdot \mathcal{L}(t')$, for $(t, t') \neq (t_0, t_1)$. Thus, when the window becomes too large, the signal-to-noise ratio at each point of the auto-corr/x-corr becomes very small. Of course, when the size of the window is small, it is possible to distinguish efficiently.

On the contrary, we see from Fig. 8 that the leakage is well localized in a few frequencies[1]. Those frequencies are around 20 MHz, which corresponds to the dynamic of the CMOS logic (see the duration of the *bounces* in Fig. 3: it is about 25 samples, i.e., 50 ns). The clock frequency is equal to 3.57 MHz, which is much smaller. Interestingly, the leakage *is not* modulated by the periodic clock signal.

[1] Three or four frequencies are especially leaky, which is much less than the tens of leakages dates in the time domain – cf. Fig. 2.

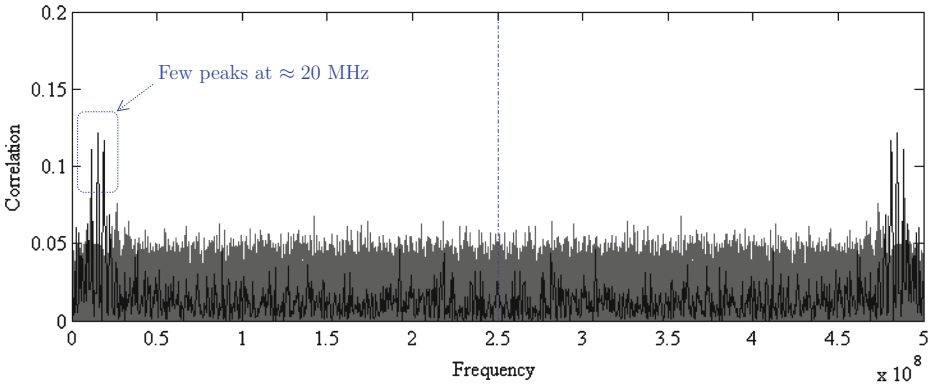

Fig. 8. Correlation coefficient on a 2O-CPA on concat-dft in frequency domain when using $n = 6000$ and 10000 traces (we recall that the sampling rate is $F_S = 500$ Msample/s)

When the window size n is large, the frequency resolution of the DFT or the DHT is high, so it is more likely that the signal is decomposed close to the main leaking frequencies (i.e., the 20 MHz frequency value is well approximated in the domain of the DFT/DHT — recall that frequencies are *quantified*, i.e., discrete variables $f \in F_S/n \times [\![0, n-1]\!]$, where F_S is the sampling rate). Additionally, there are many leaking samples in the timing window (recall Fig. 2), but the Fourier transform manages to constructively sum them up.

5.3 Comparison with the State-of-the-Art

There are several existing methods to evaluate the resistance against second-order attacks in the state-of-the-art. Among the most recent published methods that can be applied to evaluate our masking scheme, we can consider a direct 2O-CPA with pointwise multiplication of $\mathcal{L}(t_0) \cdot \mathcal{L}(t_1)$ by using the detection method proposed in [13]. As explained in Sect. 4.1, two heterogeneous leakage variables that share information about the mask can be extracted from the power traces. In our case these two leakages depend respectively on $\{T, M\}$ and $\{T', K, M\}$. Formally, in a fixed chosen plaintext scenario it is possible to identify the leakage points by searching the couples of points that maximizes the quantity: $\hat{I}(\mathcal{L}(t); \mathcal{L}(t'))$, where $\hat{I}$ denotes the estimator of the mutual information.

This method, although more efficient than performing $\binom{n}{2}$ 2O-CPA, remains of quadratic, i.e., $\mathcal{O}(n^2)$, complexity. Besides, it cannot be applied directly to the context of known plaintexts (*random, not chosen*) scenario. In [13], an extension of this method is presented. It is possible to consider the couples of points that maximize: $\hat{I}(\mathcal{L}(t); \mathcal{L}(t'); \mathcal{M}_1)$, where $\mathcal{M}_1$ is a model of the leakage. This value is high when the variation of the leakage depends on $\{T, M\}$. In our case (DPA contest v4), the variation of the leakage also depends on another plaintext byte T', thus this method will be less practical. This method could be extended by

using: $\hat{I}(\mathcal{L}(t); \mathcal{L}(t'); \mathcal{M}_0; \mathcal{M}_1)$. In this case, we have to consider a quadrivariate mutual information analysis that is likely to be little efficient in the presence of noise, and would require more traces to identify the leakage points. Our methods (cf. Sect. 3.2) basically skip the detection step, and perform a direct 2O-CPA on larger windows than in [15].

Among the state-of-the-art methods, Moradi and Mischke reported at CHES '13 [9] a similar approach as [15], where the attack is performed in time basis after point combination. In the case they report, the two leaking time samples are close in time (a few tens of clock cycles), and the low-pass filtering of the acquisition system mixes the two signals. The scenario of the attack is thus the same. The difference is however that the "overlapping" of the two leaking signals is done for free in Moradi and Mischke's setup, whereas it is forced by a preprocessing in our case. Indeed, in our masking scheme, the two sensitive variables masked with the same mask M are not used consecutively.

6 Conclusions and Perspectives

We present five preprocessing techniques that turn a bi-variate attack into a second-order zero-offset attack. Our technique applies even if the two leakage samples to be combined are far from each other. Remarkably, the proposed methods need only a rough estimate of the location of two windows (around t_0 and t_1), where the two leaks can be found purportedly. The regularity of encryption algorithms, such as the AES, facilitates the identification of the elementary operations, like plaintext blinding and S-box calls.

In addition, we notice that our techniques have the potential to scale for higher-order attacks. For instance, imagine $d+1$ shares that are leaking at time samples $t_0, t_1, \ldots, t_d$. If the attacker is only able to know an approximate window $\mathcal{L}_i$ containing t_i ($i \in [\![0, d]\!]$), then window-dht becomes simply $\prod_{i=0}^{d} \mathsf{DHT}[\mathcal{L}_i]$. The working factor of this dth-order CPA attack method is that this product, once expanded, contains terms of the form $\prod_{i=0}^{d} \mathcal{L}(t_i)$, which indeed combines multiplicatively the leakage from *all* the shares.

References

1. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
2. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved collision-correlation power analysis on first order protected AES. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 49–62. Springer, Heidelberg (2011)
3. Coron, J.-S., Kizhvatov, I.: Analysis and improvement of the random delay countermeasure of CHES 2009. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 95–109. Springer, Heidelberg (2010)
4. Dabosville, G., Doget, J., Prouff, E.: A new second-order side channel attack based on linear regression. IEEE Trans. Comput. **62**(8), 1629–1640 (2013)

5. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proc. IEEE **93**(2), 216–231 (2005). doi:10.1109/JPROC.2004.840301
6. Goubin, L., Patarin, J.: DES and differential power analysis. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
7. Hartley., R.V.L.: A more symmetrical Fourier analysis applied to transmission problems. Proc. IRE **30**(3), 144–150 (1942)
8. Mateos, E., Gebotys, C.H.: A new correlation frequency analysis of the side channel. In: Proceedings of the 5th Workshop on Embedded Systems Security, WESS '10, pp. 4:1–4:8, ACM, New York (2010)
9. Moradi, A., Mischke, O.: On the simplicity of converting leakages from multivariate to univariate. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 1–20. Springer, Heidelberg (2013)
10. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
11. Oswald, E., Mangard, S., Herbst, C., Tillich, S.: Practical second-order DPA attacks for masked smart card implementations of block ciphers. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 192–207. Springer, Heidelberg (2006)
12. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
13. Reparaz, O., Gierlichs, B., Verbauwhede, I.: Selecting time samples for multivariate DPA attacks. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 155–174. Springer, Heidelberg (2012)
14. TELECOM ParisTech SEN Research Group. DPA Contest, 4th edn. (2013–2014). http://www.DPAcontest.org/v4/
15. Waddle, J., Wagner, D.: Towards efficient second-order power analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)