Side-Channel Countermeasures' Dissection

and the Limits of Closed Source Security Evaluations

Olivier Bronchain and François-Xavier Standaert

ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, Belgium.
firstname.lastname@uclouvain.be

Abstract. We take advantage of a recently published open source implementation of the AES protected with a mix of countermeasures against side-channel attacks to discuss both the challenges in protecting COTS devices against such attacks and the limitations of closed source security evaluations. The target implementation has been proposed by the French ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) to stimulate research on the design and evaluation of side-channel secure implementations. It combines additive and multiplicative secret sharings into an affine masking scheme that is additionally mixed with a shuffled execution. Its preliminary leakage assessment did not detect data dependencies with up to 100,000 measurements. We first exhibit the gap between such a preliminary leakage assessment and advanced attacks by exhibiting how a countermeasures' dissection exploiting a mix of dimensionality reduction, multivariate information extraction and key enumeration can recover the full key with less than 2,000 measurements. We then discuss the relevance of open source evaluations to analyze such implementations efficiently, by exhibiting that certain steps of the attack are hard to automate without implementation knowledge (even with machine learning tools), while performing them manually is trivial. Our findings are not due to design flaws but from the general difficulty to prevent side-channel attacks in COTS devices with limited noise. We anticipate that high security on such devices requires significantly more shares.

Keywords: Side-Channel Attacks · Security Evaluations · Certification · Affine Masking · Shuffling · Worst-Case (Multivariate) Analysis · Open Source Design

Introduction

General context. The main motivation of this work is to discuss and illustrate a worrying (and potentially increasing) gap between academic research and industrial practice in the field of embedded security. Our starting observation for this purpose is that the design of concretely relevant side-channel countermeasures is now widely accepted as a critical ingredient for the security of embedded devices that are deployed in an increasing amount of applications. The recent NIST competition stating this as an explicit goal is one good illustration of this trend.[1] Yet, the meaning of what is concretely relevant in this field of study is still a matter of quite subjective interpretation. We argue that this state of affairs is at least in part due to the very different (open source vs. closed source) models in which academic research and industrial practices perform their investigations.

On the one hand, recent academic research has been strongly focused on the understanding of the basic mechanisms that can lead to any kind of "physical security amplification", which is typically obtained in an open source setting. As generally accepted in cryptography, the hope of this line of works is to implement objects for which security can be argued based on a mix of physical assumptions and mathematical analysis, without relying

[1] https://csrc.nist.gov/projects/lightweight-cryptography

on the secrecy of the countermeasures implemented. The most prominent example of such investigations is masking [CJRR99], which has been formally analyzed in different models [ISW03, PR13, DDF14, DFS15, BDF+17]. Roughly, masking can amplify the noise in the side-channel measurements exponentially, at the cost of quadratic performance overheads. Yet, reaching high security levels using COTS components typically requires large number of shares (see [JS17, GPSS18] for recent discussions in software) and therefore performance overheads which may at some point exceed the limits imposed by industrial application constraints.[2] Various other countermeasures have been introduced and analyzed, ranging from heuristic to formal. We mention shuffling [HOM06, VMKS12] and re-keying [MSGR10, DKM+15] as popular examples.

On the other hand, public reports of side-channel analysis against real-world devices are actually scarce and mostly limited to unprotected (or weakly protected) implementations. In this line of works, most of the contribution generally lies in the understanding of the (protocols and algorithms) specifications and implementation, in order to exhibit an attack vector. Once such a (time-consuming) reverse engineering is performed, the actual side-channel attack is usually close to trivial. Typical examples include attacks against the Keeloq hopping scheme [EKM+08], against FPGA bitstream encryption [MBKP11], against (unprotected) SIM cards [ZYSQ13] or more recently against IoT devices [RSWO17, DK18]. While these works are very important to increase the awareness that side-channel attacks are quite accessible with limited equipment, they are not very informative regarding the level of security of more protected (possibly certified) products.

In this respect, an additional source of gap is that current certification schemes favor "security by obscurity". Namely, making implementations public currently may reduce the certification levels since the use of "sensitive information" in an attack actually increases its "complexity" (as perceived by certification schemes) [LR10]. In general, one may certainly wonder about the long-term relevance of relying on the secrecy of an implementation (in view of the history of failures of closed source cryptographic designs [BSW00, GdKGM+08]). Pragmatically, the impact of such a policy is that it prevents academic research to be of any direct use in improving the security of deployed products, since (*i*) the reverse engineering effort has usually little value from a scientific viewpoint, and (*ii*) the publication of any result of an attack against a certified product, even performed under responsible disclosure practices, has good chances to be blocked at a legal level. So while it is generally assumed that certified products are based on a mix of countermeasures, how these countermeasures are mixed and how their security is assessed can only be speculated.

Considering this general context, the recent publication of an AES implementation protected against side-channel attacks thanks to a mix of countermeasures, purposed for an accessible COTS device (i.e., an STM32 chip that is based on an ARM `Cortex-M4` architecture), by a team from the French ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information), is a very welcome initiative [BKPT].

Besides the effort in publishing an open source library mixing state-of-the-art countermeasures aimed at a minimum security level, the ANSSI implementation also comes with a preliminary leakage assessment that we can hope reflective of the first (minimum) steps that an evaluation laboratory would perform, namely leakage detection [SM16] / SNR computations [Man04] and basic (1st-, 2nd- and 3rd-order) divide-and-conquer attacks [SVO+10]. So while this implementation was not claimed to be designed to pass certification nor to be usable for commercial applications, it is at least a useful starting point for research purposes, since developed and analyzed by a team of experts with both academic and industrial experience. More specifically, it can be used to guide research towards a better understanding of three important questions, namely (*i*) how to evalu-

[2] Similar challenges appear in hardware, with slight variations. For example, better noise levels can usually be achieved but physical defaults such as glitches may lead to additional implementation constraints, especially in the composition of masked gadgets at high security orders [NRS11, GMK17, MMSS19].

ate such a mix of countermeasure in a (close to) worst-case manner; (*ii*) which security levels can be reached by software implementations with low number of shares in (e.g., ARM-based) COST devices (that are typical targets for deployment in IoT applications); and (*iii*) how much the knowledge of the target implementation is critical to reach such a worst-case understanding of the physical security level?

Contributions. Based on this state-of-the-art, we start by tackling the first two aforementioned questions related to the security evaluation of protected cryptographic implementations. Precisely, we use the ANSSI software library in order to discuss how much the preliminary leakage assessment that comes with this implementation can be used as an indicator of a quantitative (close to worst-case) security level. For this purpose, we investigate the problem of evaluating combined countermeasures and their effectiveness.

In this respect, preliminary leakage assessments based on simple detection tools are typically reflecting the informal expectation that the security of each countermeasure leads to a (strong) multiplicative impact on the security level. We mitigate this expectation thanks to methodological and technical tweaks.

At the high level, we use the knowledge of the implementation to perform countermeasures' dissection. By dissection, we mean a detailed analysis of each ingredient of a protected implementation, in order to limit their (combined) impact as much as possible. We note that from a complexity viewpoint, a dissection attack is not equivalent to a divide-and-conquer one. Divide-and-conquer attacks turn a multiplication of complexities into a sum. This is what happens when performing 16 independent DPAs against the 16 key bytes of the AES (which has a time complexity $\propto 16 \cdot 2^8$) rather than performing an exhaustive search on the full key (which has a time complexity of 2^{128}). In a dissection attack, the complexities of different countermeasures are still multiplied, but the adversary aims at keeping the attack factors as close to one as possible.

At the low-level, we use more advanced (multivariate) information extraction tools. Precisely, the ANSSI implementation is mixing an affine masking scheme [FMPR10] with a shuffled execution. We show that a PCA can be used to recover the multiplicative mask of the affine masking in full [APSQ06], that some parts of the implementation are weakly (or not) shuffled, that a multivariate higher-order template attack can significantly reduce the number of samples needed to circumvent the remaining shuffling and additive masks [CRR02], and that key enumeration can be used to additionally reduce the data complexity of the attack [PSG16]. Our tools remain simple (e.g., we do not exploit algebraic/analytical techniques [RSV09, VGS14, GS15, GRO18]) and the risks of precomputed tables in masking was put forward in [TWO13, BGNT18]). Yet, their combined impact is strong: preliminary leakage assessments did not detect data dependencies with up to 100,000 measurements. We recover a full key with less than 2,000 measurements. Our evaluations further indicate that reaching high security levels on a `Cortex`-like device would require (very) large number of shares (see the discussion in Section 3.4).

As a side-result, we briefly discuss the impact of using different implementations of the same `Cortex-M4` architecture, with different measurement setups. We observe that conclusions are similar (i.e., the same instructions can be targeted essentially in the same way) with some limited variations in the measurement noise and attack complexities.

Following these findings, our second goal is to analyze how much the knowledge of the implementation details are useful to gain such a better understanding of the worst-case security level. We link this question to recent (and numerous) attempts to improve security evaluations thanks to machine learning / deep learning. In this context, a recurrent claim is that machine learning / deep learning has the potential to simplify / automate some (most?) of the attacks steps. For example, it has been shown that machine learning / deep learning can limit the need to synchronize traces and to identify their points of interests (see [HGM+11, HZ12, LMBM13, LPB+15, CDP17, PSK+18, ZS19] for a few examples), while some works further argue that such tools can succeed in a fully black

box / automated manner and without knowledge / understanding of the underlying implementation [Tim19, BP19]. Our results suggest that despite the latter claim could be demonstrated in some previous studies where at most one countermeasure was implemented at a time, such an automation may not always succeed with low complexities in case of combined countermeasures. In particular, our experiments show that countermeasures dissection may benefit from manual expertise and highly improve the attack complexities. For example, our attacks recover the multiplicative mask of an affine masking scheme and therefore perform a field multiplication in order to retrieve sensitive information. Learning this field multiplication in a fully automated manner appears to be a challenging task for existing machine learning / deep learning tools (besides being a waste since this part of the attack is trivial to perform manually). So while our results do not rule out the possibility for such tools to succeed eventually, they question their efficiency for doing so. Concretely, we believe our work at least states an interesting challenge to machine learning / deep leaning research: can the ANSSI implementation be broken in less than 2,000 traces in a black box manner (and with similar time complexities and profiling efforts as ours)?[3]

1 Background

In this section, we first recall the countermeasures implemented in [BKPT]. Then, the statistical estimation tools used to perform the attack are listed and described.

1.1 Affine masking

The targeted library implements the affine masking scheme proposed in [FMPR10] to protect the AES. All the operations are performed over the Galois Field $\mathsf{GF}(2^8)$. In such a masking scheme, the sensible value x is encoded thanks to

$$C(x; r_m, r_a) = (r_m \otimes x) \oplus r_a, \tag{1}$$

where $C(\cdot)$ is the encoding function, r_m the multiplicative mask and r_a the additive mask. The multiplicative mask is uniformly distributed over $\{1, \ldots, 255\}$ allowing to revert the multiplication. The additive mask is uniformly distributed over $\{0, \ldots, 255\}$.

The main challenge in such a masking scheme is to implement the AES Sbox. Before every encryption, an alternative Sbox' is pre-computed depending on two additional additive masks r_{in} and r_{out} as well as r_m. During the encryption, the Sbox is applied to the encoding of x following

$$C(\mathsf{Sbox}(x); r_m, r_a) = \mathsf{Sbox}'(C(x; r_m, r_a) \oplus r_{in} \oplus r_a) \oplus r_a \oplus r_{out}. \tag{2}$$

That is, a look-up table to the alternative Sbox' is made while the additive mask r_a is locally canceled and replaced by the input and output masks r_{in} and r_{out} (that correspond to the pre-computed table). In the studied implementation, one r_a per byte is required and a single alternative Sbox' is pre-computed for all the bytes, meaning that a single r_m, r_{in} and r_{out} triple is used, leading to randomness requirements per encryption of 19 bytes. In the rest of the paper, $\vec{C}$ denotes the encoded 4x4 matrix of the AES state. Similarly, $\vec{R_a}$ corresponds to the 16 additives masks consistent with the encoding $\vec{C}$. The `MixColumns` operation can be performed on each column of $\vec{C}$ and $\vec{R_a}$ independently.

[3] We insist that our questioning is not about the interest of machine learning / deep learning algorithms as useful tools (among others) for side-channel analysis in general, but about the need to know and exploit implementation details and expert knowledge in order to reach a good quantitative understanding of the worst-case security level of an implementation efficiently.

1.2 Shuffling

On top of the previously exposed affine masking, the targeted implementation is using operations' shuffling [VMKS12]. Therefore, some sets of independent operations are performed in a randomized order. For the Sbox executions and for `ShiftRows`, permutations $p_{\vec{C}}$ and $p_{\vec{R_a}}$ over the set $\{0, \ldots, 15\}$ are generated based on a 16-bit random seed. For `MixColumns`, smaller permutations $p'_{\vec{C}}$ and $p'_{\vec{R_a}}$ over the set $\{0, \ldots, 3\}$ are generated based on a 2-bit random seed. In the following, we will assume that the first permutations cannot be enumerated while the second ones can trivially (making them similar to the weaker random start index shuffling discussed in [VMKS12]).

A high-level view of how the affine masking and shuffling countermeasures are combined in the ANSSI implementation is given in Figure 1.

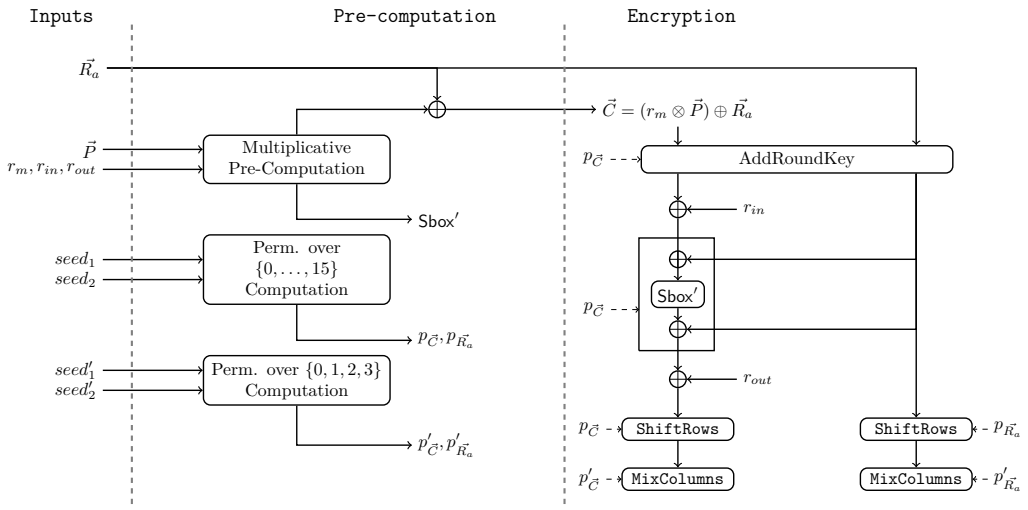


Figure 1: Combined (affine masking + shuffling) countermeasures: high-level view.

1.3 Statistical tools

One of the main challenge in side-channel analysis is to estimate a Probability Density Function (PDF) $\mathsf{f}[x|\vec{l}]$ where x is an internal state and $\vec{l}$ the observed (possibly multivariate) leakage sample. Such a PDF can (for example) be estimated with Gaussian templates [CRR02]. There, the adversary uses a clone device to estimate the distribution of the leakage conditioned on x under a Gaussian assumption, namely

$$\mathsf{f}[\vec{l}|x] = \frac{1}{\sqrt{(2\pi)^k |\vec{\Sigma}_x|}} \exp^{-\frac{1}{2}(\vec{l}-\vec{\mu}_x)' \vec{\Sigma}_x (\vec{l}-\vec{\mu}_x)} . \tag{3}$$

As a result, he has to estimate the co-variance matrix $\vec{\Sigma}_x$ and the mean $\vec{\mu}_x$ for each possible intermediate variable $x \in X$. Bayes' theorem is used to obtain the probability

$$\Pr[x|\vec{l}] = \frac{\mathsf{f}[\vec{l}|x]}{\sum_{x^* \in X} \mathsf{f}[\vec{l}|x^*]} , \tag{4}$$

which will be later used to mount the attack.

The Signal-to-Noise Ratio (SNR) metric can be used to determine the available (first-order) signal about x within the leakage traces [Man04]. In such a situation, the signal is

assumed to lie in the means of the leakage samples. The SNR can be estimated as

$$\hat{\mathrm{SNR}} = \frac{\mathsf{E}[\mu_x^2 - \mathsf{E}[\mu_x]]}{\mathsf{E}[\sigma_x^2]}\,, \tag{5}$$

where $\mathsf{E}[\cdot]$ is the expectation operator. The SNR is also useful in order to highlight all the so-called Points-Of-Interest (POIs) in the traces (i.e., the points that contain first-order information about x). Such a preliminary inspection allows determining the points for which it is useful to build a Gaussian template. So it can be viewed as a simple dimensionality reduction technique.

A slightly more advanced dimensionality reduction is Principal Component Analysis (PCA). In [APSQ06], the authors showed how to use PCA as a pre-processing before launching a template attack. In short, it takes as input a trace with a high number of dimensions and outputs a reduced trace with at most $|X|-1$ dimensions, while maximizing the inter-class variance (i.e., the upper term of Equation (5)).

Similarly to classical $g\mathsf{T}$s, Multi-Layer Perceptrons (`MLP`) can be used as an estimation tool allowing to compute a metric $\tilde{\mathsf{f}}[\vec{l}|x]$ used later as an alternative to a templat's PDF. An `MLP` is composed of layers each containing nodes. The nodes of sub-sequent layers are connected in a forward manner. Each of them outputs a weighted sums of their inputs applied to a non-linear function called activation function. The weights are estimated with the back-propagation algorithm during the profiling phase. The meta-parameters of the `MLP` must also be selected during the profiling phase. We refer to [Bis07] for the details.

2 Code inspection / countermeasures' dissection

In this section, we discuss different attack vectors that are exploitable in the protected implementation from [BKPT]. First, a generic distinguisher strategy is proposed. Second, the generic equation is simplified based on reasonable hypotheses (that we discuss in this section and will validate experimentally in the next sections). This allows a significant improvement of the attack time complexity. Third, we dissect the combination of countermeasures and describe how the different ingredients of the ANSSI implementations can be analyzed separately, in order to reduce the attack data complexity. Finally, four different attack strategies are proposed for comparison purposes.

2.1 Generic distinguisher

In order to mount a successful attack, a side-channel distinguisher is interested in the probability of an intermediate variable X depending on the secret key (e.g., the output of the Sbox) given a fresh leakage sample. In the case of affine masking, the secret value x is protected with two random masks r_m and r_a, so we need to compute

$$\mathsf{f}[l|x^i] = \sum_{r_m}\sum_{r_a^i}\mathsf{f}[l|r_m, r_a^i, c^i = (x^i \otimes r_m) \oplus r_a^i] \cdot \Pr[r_a^i]\Pr[r_m], \tag{6}$$

where the exponent i denotes the targeted byte. This equation sums over all the possible masks weighted by their (uniform) probability. Given these masks, the likelihood to observe the leakage l conditionally to x^i is computed. This expression can latter be used by a distinguisher to apply the Bayes's law and recover the secret key.

If a shuffled execution is used on the top of the affine masking, the probabilities of the shuffled states also have to be computed by summing over the shuffled cycles (and possibly

permutations). A generic expression for the corresponding conditional PDF was given in [VMKS12]. Using our notations, it can be written as

$$\mathsf{f}[l|r_m, r_a^i, c^i] = \sum_{o_1} \frac{\mathsf{g}(o_1, i, l_1')}{\sum_{o_1'} \mathsf{g}(o_1', i, l_1')} \cdot \sum_{o_2} \frac{\mathsf{g}(o_2, i, l_2')}{\sum_{o_2'} \mathsf{g}(o_2', i, l_2')} \cdot \mathsf{f}[l_{o_1}, l_{o_2}|r_m, r_a^i, c^i, o_1, o_2]\,, \tag{7}$$

where o_1 and o_2 respectively correspond to the cycles/operations' indexes where c^i and r_a^i can be manipulated. The leakages about both permutations are denoted as l_1' and l_2'. The leakage about c^i (resp. r_a^i) is written as l_{o_1} (resp. l_{o_2}) when manipulated at index o_1 (resp. o_2). The function $\mathsf{g}(\cdot)$ depends on the permutation and the attacker strategy. We detail the functions used for this work in Subsection 2.3.

2.2 Time complexity improvements

The previous generic expression implies to estimate a large (and hardly realistic) number of PDFs (aka templates). Indeed, a direct/exhaustive profiling requires to estimate a template for every combination of r_m,r_a^i and c^i as well as for the indexes when they are manipulated, leading to a total of $255 \cdot 2^8 \cdot 2^8 \cdot |o_1| \cdot |o_2|$. Hereunder, we show how the profiling and attack time complexities can be reduced by exploiting an independence assumption in Equation (6). For this purpose, we first assume that the leakage l is composed of three leakages with independent noise (which is natural in the context of software implementations). The previous equation then becomes

$$\mathsf{f}[l|x^i] = \sum_{r_m} \sum_{r_a} \mathsf{f}[l_{r_m}|r_m] \cdot \mathsf{f}[l_{r_a^i}|r_a^i] \cdot \mathsf{f}[l_{c^i}|c^i] \cdot \Pr[r_a^i] \cdot \Pr[r_m], \tag{8}$$

where $l_\star$ is the leakage about the share $\star$. This simplification allows one to target the permutation on each of the shuffled shares independently and not jointly as performed in Equation (7). This results in equations

$$\mathsf{f}[l_{c^i}|c^i] = \sum_{o_1} \frac{\mathsf{g}(o_1, i, l_1')}{\sum_{o_1'} \mathsf{g}(o_1', i, l_1')} \mathsf{f}[l_{o_1}|c^i, o_1]\,, \tag{9}$$

and

$$\mathsf{f}[l_{r_a^i}|r_a^i] = \sum_{o_2} \frac{\mathsf{g}(o_2, i, l_2')}{\sum_{o_2'} \mathsf{g}(o_2', i, l_2')} \mathsf{f}[l_{o_2}|r_a^i, o_2] \cdot \tag{10}$$

Therefore, the profiling time complexity is greatly simplified (with direct impact on the data complexity). Namely, the number of templates to estimate is reduced to only $255 + 2^8 \cdot |o_1| + 2^8 \cdot |o_2|$. Additionally, in the direct/exhaustive profiling, a fresh profiling trace l can only be used to estimate a single template since it corresponds to a single combination of secret values (i.e., shares and permutations). Thanks to the independence assumption, that trace can be separated in independent $l_\star$'s, each being used to update a template. Consequently, for a fix number of available profiling traces, each of the PDFs will be estimated with more samples, leading to more accurate templates (i.e., less estimation errors) with less data complexity.

The introduced independence assumption also has a (more limited) positive impact on the complexity of the online attack. Namely, the computation of Equation (8) is slightly

improved compared to Equation (6). Indeed, the impact of the two permutations can be estimated separately, reducing the number of sums it induces from $|o_1| \cdot |o_2|$ down to $|o_1| + |o_2|$. In Subsection 2.3, we show how the attack complexity can be further (and significantly) improved by exploiting countermeasures' dissection.

As already mentioned, the proposed independence assumption is natural in the context of software implementations where the shares are manipulated in different clock cycles. Yet, it requires that the noise variables of the different shares are uncorrelated to a sufficient extent, which may of course not be perfectly fulfilled. Our following experimental results will show that this assumption leads to excellent attack results (yet that a machine learning based attack escaping this assumption leads to mild improvements).

2.3 Countermeasures' dissection

Based on the previous abstractions, we can now easily describe the goal of countermeasures' dissection. Namely, and in order to reduce the data complexity of the attacks, the goal of such a dissection is to try biasing the weights of each sum in Equation (8), ideally to the point where there is a single probability one event (i.e., mask, permutation) to consider. To do so, sub-attacks are performed against each single trace in order to (at least partially) recover ephemeral secrets such as the multiplicative mask r_m. For example, the previous equation can be rephrased by using Bayes's law as

$$\mathsf{f}[l|x^i] = \left(\frac{1}{|r_a||r_m|} \cdot \sum_{r'_m} \mathsf{f}[l_{r'_m}|r'_m] \right) \cdot \sum_{r_m} \sum_{r_a^i} \Pr[r_m|l_{r_m}] \cdot \mathsf{f}[l_{r_a^i}|r_a^i] \cdot \mathsf{f}[l_{c^i}|c^i] \tag{11}$$

$$= \alpha \cdot \sum_{r_m} \Pr[r_m|l_{r_m}] \cdot \sum_{r_a^i} \mathsf{f}[l_{r_a^i}|r_a^i] \cdot \mathsf{f}[l_{c^i}|c^i]. \tag{12}$$

There, the factor α does not need to be computed since it is independent of x and will therefore be canceled in Equation (4) (i.e., when the attack is performed). Expressed in that form, it appears that the probability of the multiplicative mask r_m given the leakage sample is first computed. This is equivalent to a partial attack on that secret value of which the output is used to attack the complete scheme. In the case this sub-attack is successful, $\Pr[r_m|l_{r_m}]$ will be close to one for the correct r_m and close to zero for all the remaining possibilities. In that case, only the sum over r_a must be performed for the correct r_m. This reduces the number of sums by a factor 255 and completely cancels the impact of this mask on the data complexity. In the following, we will show that such a partial attack on r_m is successful with an accuracy of 100% (thanks to the heavily leaking pre-computation that manipulates this mask).

A similar approach can be used to skip the sum over $|o_1|$ and $|o_2|$ in Equation (9) and Equation (10). The main observation in this case is that the ANSSI implementation contains two permutations: one over the 16 Sboxes, one over the 4 `MixColumns`. As a result, a natural target for an adversary is to try recovering the smaller permutation (and to sum over all the remaining candidates, which can be easily done). Using the terminology of [VMKS12], it means that we can anyway exploit a direct permutation leakage. In this case, we take $\mathsf{g}(o_1, i, l'_1) = \mathsf{f}[l'_1|o_1 = \mathsf{col}(i)]$, where $\mathsf{col}(i)$ corresponds to the index of the processed column in which the byte i is contained (the same holds with o_2 and l'_2). And in case more than one permutation remains possible, we can simply sum over the remaining

ones.[4] As a result, the previous equation can be rephrased as

$$\mathsf{f}[l_{c^i}|c^i] = \sum_{o_1} \frac{\mathsf{f}[l'_1|o_1 = \mathsf{col}(i)]}{\sum_{o'_1} \mathsf{f}[l'_1|o'_i = \mathsf{col}(i)]} \mathsf{f}[l_{o_1}|c^i, o_1] \tag{13}$$

$$= \sum_{o_1} \Pr[o_1 = \mathsf{col}(i)|l'_1] \cdot \mathsf{f}[l_{o_1}|c^i, o_1], \tag{14}$$

thanks to Bayes's law. Additionally, since a 2-bit seed is used to generate the `MixColumns` permutation, $o_1 = \mathsf{col}(i)$ is only true for a single seed value. Taking advantage of these observations, we will show in the following that a sub-attack against the seed of the permutation reaches a 98% accuracy, only leading to a very limited increase of the attack data complexity. As for the time complexity, the only remaining sum after the dissection presented here is over the 256 additive masks.

One may note that an extreme version of the countermeasures' dissection proposed here would be to directly express the ANSSI implementation as a factor graph and to take advantage of algebraic/analytical attacks [RSV09, VGS14, GS15, GRO18], where a similar independence assumption and a similar knowledge of the implementation details are also exploited, yet with more complex tools. We leave the investigation of such attacks as an interesting scope for further investigation.

2.4 Exemplary attack strategies

Before moving to concrete experiments, we finally specify a few representative (more or less powerful) attacks against the ANSSI implementation. Concretely, they are all based on the computation of Equation (8) exploiting the leakage of different operations.

Looking at Figure 1, it appears that an adversary at least has to target one operation involving $\vec{C}$ and one operation involving $\vec{R_a}$. Depending on this choice, the signal available from the leakage samples as well as the effectiveness of the shuffling countermeasure (e.g., if only `MixColumns` is considered) may change, leading to different attack complexities.

Our four instantiated adversaries are:

- $\mathcal{A}_1$ obtains information about $\vec{C}$ by targeting the XOR of the state with r_{out}. This operation is not shuffled making the estimation of $\mathsf{f}[l_c^i|c^i]$ straightforward. Information about $\vec{R_a}$ is extracted from the shuffled `MixColumns` with the complete sum of Equation (14). We note that this attack could be easily avoided by shuffling the XOR operation. The number of sums to perform is $2^8 \cdot 4$.
- $\mathcal{A}_2$ targets `MixColumns` for both shares but does not exploit the permutation leakage. Therefore, she has to perform the sum of Equation (9) and Equation (10) with $\mathsf{g}(\cdot) = 1/4$. Hence, the number of sums to perform is equal to $2^8 \cdot (4 + 4)$.
- $\mathcal{A}_3$ is similar to $\mathcal{A}_2$ but does exploit the `MixColumns` permutation leakage. She explicitly computes the sum over all the possible indexes with Equation (14).
- Finally, $\mathcal{A}_4$ is equivalent to $\mathcal{A}_3$ but only considers one (i.e., the most likely) event in Equation (14) to in order to limit the number of sums to 2^8.

All these attacks will be experimented, compared and discussed next. First, in Subsection 3.3, the PDFs will be estimated under a Gaussian assumption (possibly exploiting `PCA` for dimensionality reduction). Next, in Section 4, we show how a `MLP` can used to estimate the leakage on each of the shares individually leading to slightly improved performances. In between, we discuss the limitations of black box security evaluations.

[4] Which is in strong contrast with an attack against the permuted Sboxes where enumerating over $16! \approx 2^{44}$ is hardly practical. This is why it is advised in [VMKS12] to perform 12 dummy `MixColumns` operations so that this part of the AES implementation can be sufficiently shuffled as well.

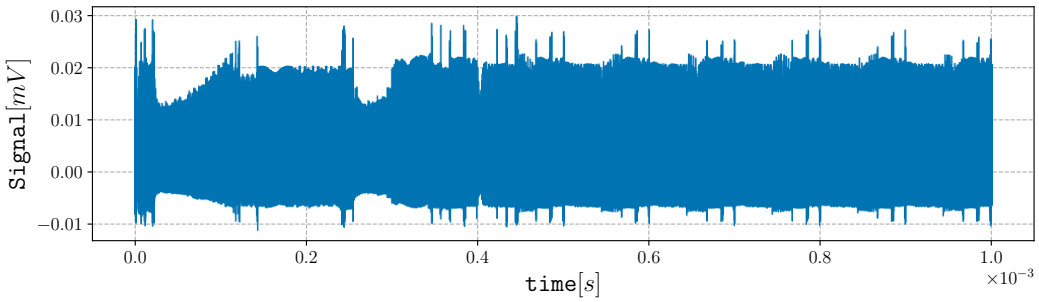


Figure 2: Mean trace obtained from the studied target.

3 Experimental results

In this section, we show how practical attacks can be performed against the ANSSI library. First, the measurement setup is described. Then, the profiling phase is explained during which the PDFs of each secret value is estimated. Finally, the performances of all the adversaries are reported and commented.

3.1 Measurement setup

The targeted library is running on a `ATSAM4LC4CA` that is a `Cortex-M4` based micro-controller. The circuit under test is placed on a commercial board `SAM4L Xplained Pro` that has been slightly modified by removing decoupling capacitors. The target is running at its maximum clock frequency of 48[MHz].[5]

Its power consumption is measured through an electromagnetic near field probe `R&S HZ-15` placed directly above the package of the micro-controller. The pre-amplified signal with a matched `R&S HZ-16` is sampled at 1[GSamples/s] with a `PicoScope 5244d` and an 8-bit resolution. The resulting averaged power trace is shown in Figure 2 where only the pre-processing and the 6 first rounds of the AES are recorded. The traces are 10^6 samples long and are obtained at a rate of about 35 traces per second (because of the interface with the scope). For this paper, $5 \cdot 10^5$ traces have been recorded with known random masks, seeds, keys and plaintexts for profiling purpose. In addition, $2 \cdot 10^5$ traces were also recorded with a fixed key in order to mount the attacks.

We stress that a particular care was paid to the quality of the measurements. More precisely, a large configuration space has been explored including probe position, sampling frequency/resolution as well as decoupling network. The traces used in this work are recorded in the setup maximizing the `SNR` on the multiplicative mask r_m. This choice is made because of its importance in countermeasures' dissection. No engineering on frequency filtering was performed, which could possibly lead to improved attacks.

Note that the implementation and setup described in this section are not exactly the same as used in the preliminary leakage assessments performed by the ANSSI team. The impact of these differences on our conclusions will be discussed in Section 3.5.

3.2 Templates' estimation

Our templates estimation use the tools from Subsection 1.3 and proceed in two steps:

[5] It turned out that the library was not constant time because of cached memory access on this platform. The experiment are performed with a table free `xtime` making the encryption constant time.

1. Dimensionality reduction by using the SNR to isolate the samples for which the mean contains information about the sensible variables. Then a PCA is optionally applied depending on the number of dimensions showing evidence of leakage.

2. Gaussian Templates ($g\mathsf{T}$) are built on the reduced dimensions. Ten of them are typically taken into account since it allows to cover most of the leakage samples.

We next highlight how a PDF can be estimated for all the secret variables, starting with the multiplicative mask, followed by the permutations' seed and the additive shares.

3.2.1 Multiplicative mask

In order to build a $g\mathsf{T}$ to obtain $\mathsf{f}[l_{r_m}|r_m]$, the pre-computation of the multiplicative table by r_m is targeted. Indeed, this is the only pre-computation that only involves r_m and no other secret value. The corresponding SNR is shown in Figure 3, and exceeds 200 on the peaks value. Such a high value is due to type of memory accesses targeted: words of 32 bits are stored in memory for each of the four bytes corresponding to four multiples of r_m. The stored values are 32-bit wide but only depend on 8 bit. This leads to a variance between the signal of the 255 classes that dominates the noise in our experimental setup.

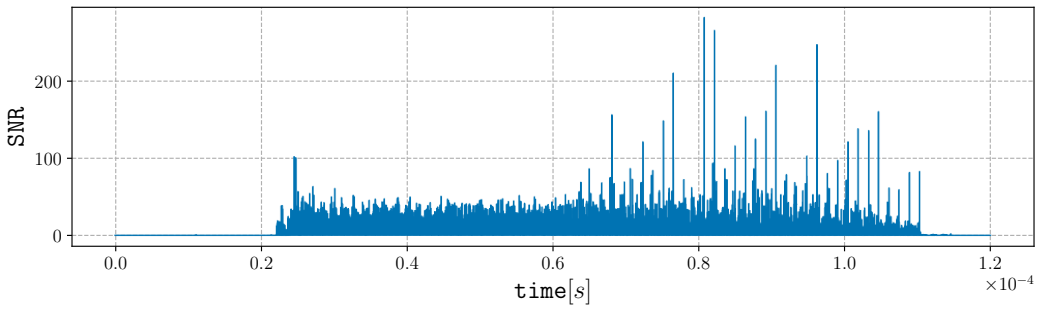


Figure 3: SNR on r_m during the multiplicative table pre-computation.

Because of the large number of samples depending on r_m, a PCA is additionally used as a dimensionality reduction tool on all the samples from the table pre-computation. Only the 10 most significant dimensions are used to build the $g\mathsf{T}$. The resulting model allows to perfectly re-identify the multiplicative mask with a single trace (i.e., with an accuracy of 100% and on all the $2 \cdot 10^5$ attack traces).

3.2.2 Permutations

The PDF $\mathsf{f}[l'_1|o_1 = \mathsf{col}(i)]$ can be derived from $\mathsf{f}[l'_1|seed'_1]$ where $seed'_1$ is a 2-bit used to generate the permutation (again, a similar equation holds with o_2, l'_2 and $seed'_2$). The SNR of both permutations is shown in Figure 4. The first peak shows the loading of the random bits. The second ones correspond to the permutations' computation. Then, the pattern repeated six times is the actual MixColumns. It clearly appears that the two shares are processed serially starting with $\vec{R_a}$ followed by $\vec{C}$. In this operation, the leakage depends on the memory address accessed, which depends on the studied seeds.

Because of the large dimensionality of the leakage, we first selected 3,000 dimensions showing evidence of leakage based on the SNR. These are reduced using a PCA to only 3 dimensions (leveraging the small sample size variant of PCA proposed in [APSQ06]). Figure 5 shows the projected samples for the four possible $seed'_1$ values. Already by looking to the two first dimensions (Figure 5a), most of the classes are well separated. Only the pink and the green clusters are overlapping. By taking the third dimension into

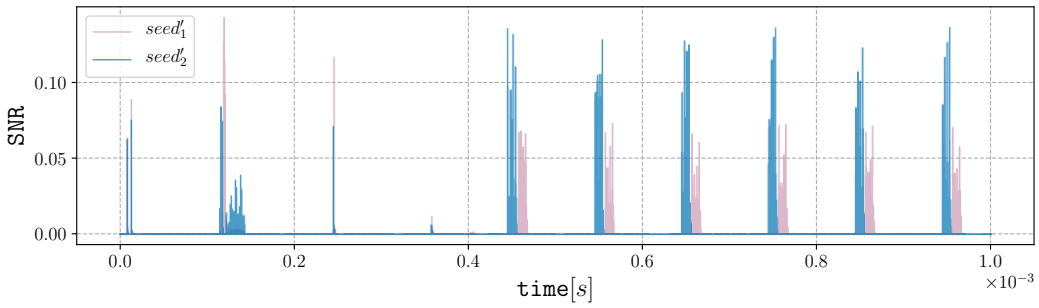
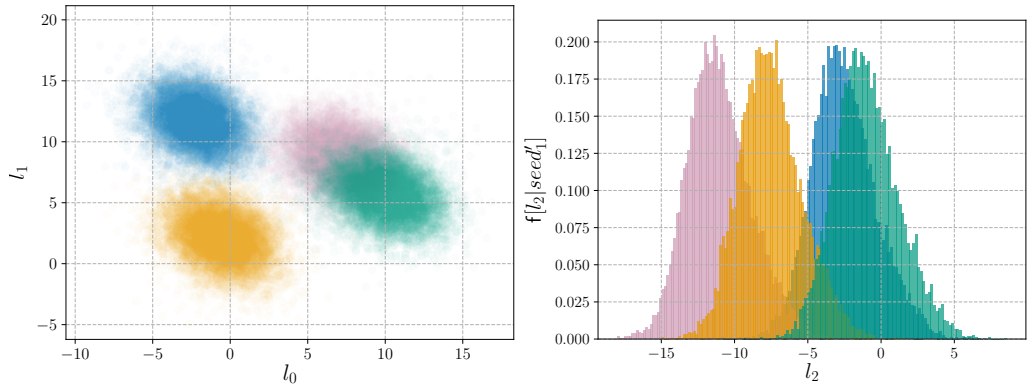


Figure 4: SNR on MixColumns permutations' seeds.

consideration, these two are also well separated. Four gTs are then fitted based on these clusters, and allow retrieving the correct seed with an accuracy of 98%.



(a) Dimensions 1-2.

(b) Dimension 3.

Figure 5: PCA output on $seed'_1$ for the three studied dimensions.

Intuitively, this accuracy can be explained by the fact that the same permutation is used at every round, increasing the number of leaking dimensions about the seeds. These distant samples have potentially uncorrelated noise while manipulating the same data and therefore allow reducing the noise thanks to averaging. Such an averaging is implicitly performed by the PCA that is a weighted sum of samples. Additionally, the seed (and its bijectively derived data) are involved in different operations such as loads, permutation computations or memory accesses, each of them leaking in a different way, therefore providing additional independent information about the seed as well.

3.2.3 Additive shares

We finally highlight how information can be extracted on the additive shares given a permutation. For this purpose, one single template is computed for each index o_1, omitting the dependency to the actual manipulated byte i. In total, 16 gTs are estimated independently of the shuffling strategy. Because the leakage about a share is typically contained in a few cycles, only the SNR is used to isolate the POIs. In the following, we restrict our profiling to the operations exploited by the previously mentioned adversaries.

First, in Figure 6, the SNR on each byte of $\vec{C}$ is reported and color-grouped by column during the addition with r_{out}. This operation is implemented with a few sequential instructions. An entire column is loaded from memory, then it is bitwise XORed with r_{out}

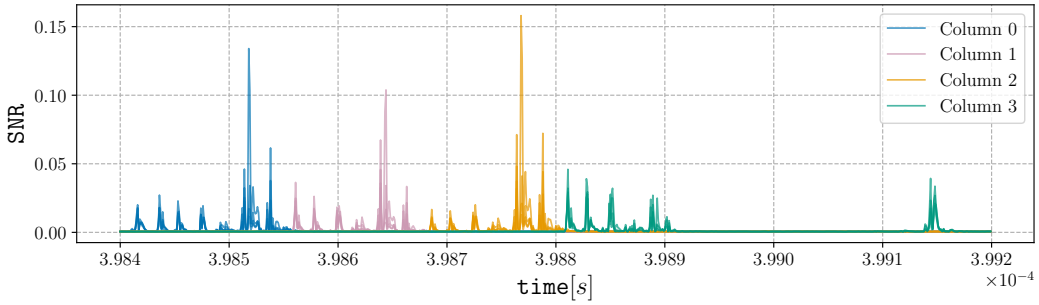


Figure 6: SNR on $\vec{C}$ during r_{out} addition.

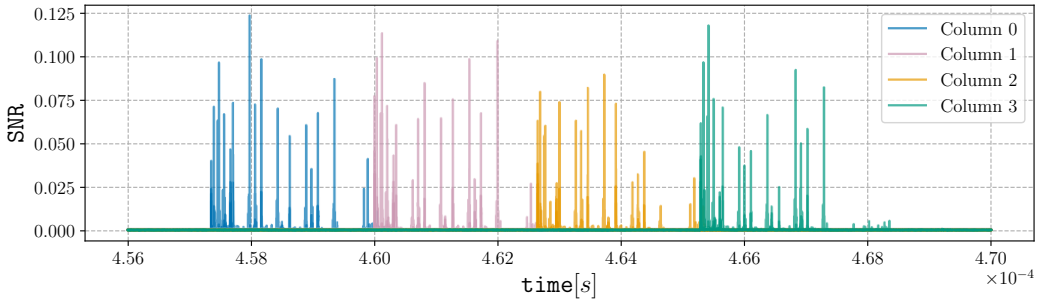


Figure 7: SNR on $\vec{C}$ during MixColumns.

(duplicated 4 times) and finally it is stored in memory. Therefore, all the four bytes of the same columns leak at the same time. To our understanding, the first peaks correspond to the XOR operation while the larger (and last) peak corresponds to the memory write. The resulting gTs are used only by $\mathcal{A}_1$.

Second, information about $\vec{C}$ can also be extracted from the MixColumns operation as exploited by $\mathcal{A}_2$,$\mathcal{A}_3$ and $\mathcal{A}_4$. Because this operation is shuffled, a profile for each index o_1 is built. The SNR is plotted in Figure 7 where the color-grouping corresponds to a constant o_1. Contrarily to the previous case, the bytes of the column are serially loaded from memory and stored in registers. Then, these bytes are combined one with each other within registers to perform the matrix multiplication of MixColumns. Therefore, the four bytes of the processed column do not always leak at the same point in time.

A similar behavior is observed for the template on $\vec{R}_a$ in Figure 8. Additionally, we see smaller peaks before the MixColumns, generated by ShiftRows. Each of them corresponds to the shuffled loading of a byte or the store of one of them. These smaller SNR values would typically be the ones that would be exploited by an adversary targeting the (well) shuffled Sbox executions in a black box manner.

3.3 Adversaries comparisons

Based on the previous profiling efforts, we can now compare the different adversaries introduced in Subsection 2.4. For each of them, the Guessing Entropy (GE) for each subkey byte is first reported [SMY09]. Then, the key rank of the full key is also estimated thanks to the rank estimation algorithm of [PSG16].

The first adversary $\mathcal{A}_1$ exploits jointly the leakage from the addition of $\vec{C}$ with r_{out} and the leakage of MixColumns on $\vec{R}_a$. As shown on Figure 9, such an adversary requires

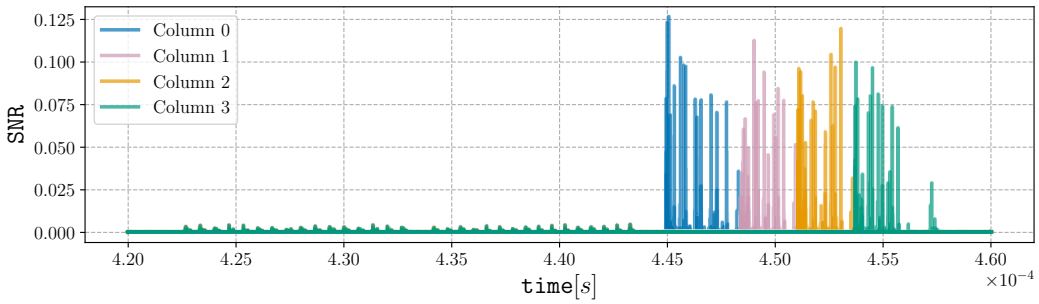
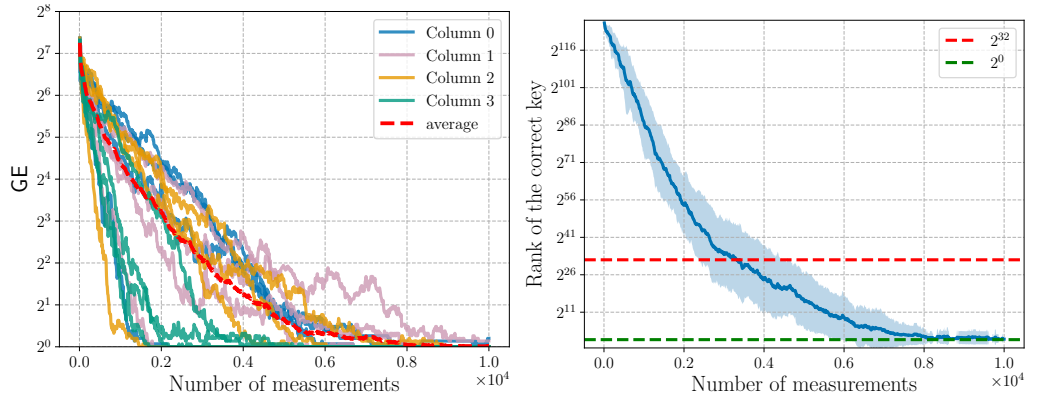


Figure 8: SNR on $\vec{R_a}$ during MixColumns.



(a) GE of all the bytes.

(b) Rank of the correct key.

Figure 9: Attack results for $\mathcal{A}_1$.

about $10 \cdot 10^3$ measurements to recover the full encryption key. The data complexity is reduced to $3 \cdot 10^3$ if key enumeration is performed.

The second adversary $\mathcal{A}_2$ targets MixColumns for both shares but does not exploit information about the permutation used. The resulting attack performances are given in Figure 10. With a total amount of $50 \cdot 10^4$, the full key is not recovered without post-processing. The latter is due to some "harder to recover" subkey bytes, as illustrated in Figure 10a. Yet, key enumeration still allows recovering the full key with $20 \cdot 10^4$ traces.

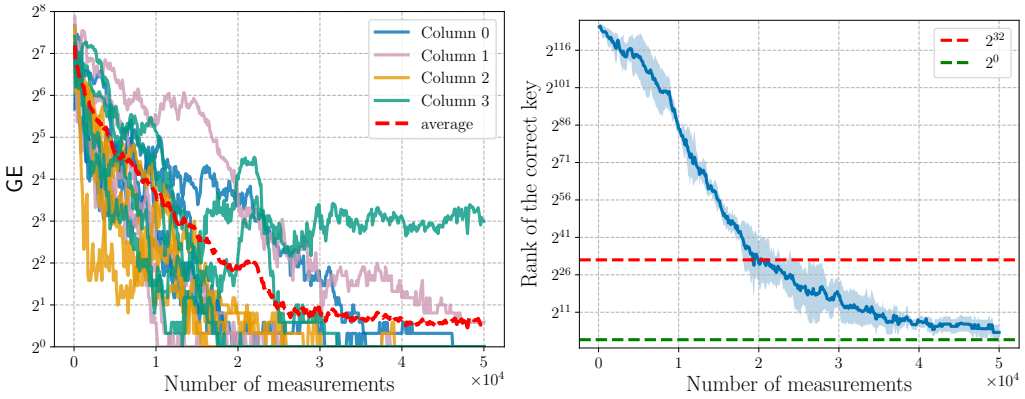
The third attack performed by $\mathcal{A}_3$ exploits the MixColumns leakage for both shares as well as the leakage on the permutations. The attack results are shown in Figure 11. With around $1.1 \cdot 10^3$ traces, the entropy of the full key is reduced to less than 32 bit, while it is directly recovered with $3 \cdot 10^3$. One can also notice that one byte per column is more difficult to recover, slowing done the full attack.

The last adversary $\mathcal{A}_4$, is reported in Figure 12. It requires about $1.2 \cdot 10^3$ traces with key enumeration and $4 \cdot 10^3$ without. Note that the GE of some of the bytes is not to 1 (but close to it) with that amount of traces (so the outputted key is not necessarily ranked first but closed to it).

3.4 Discussion

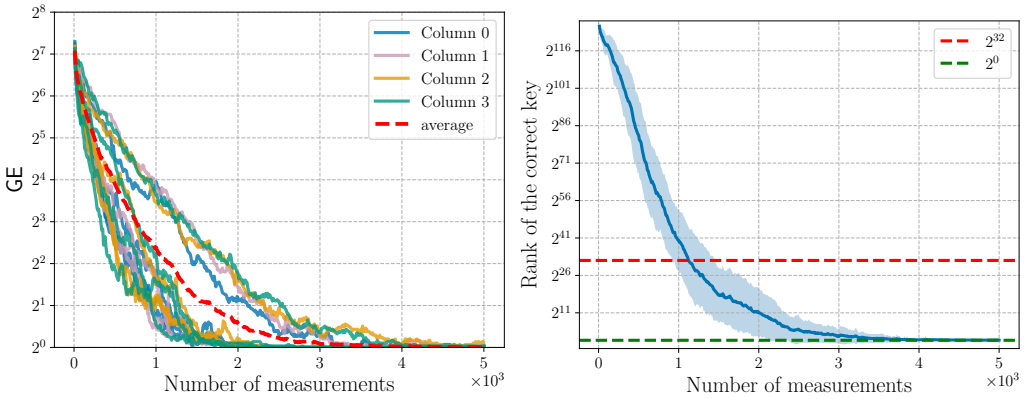
Our different results are summarized in Table 1, which allows evaluating the influence of the shuffling and targeted operations on the attacks' complexity.

First, by moving from $\mathcal{A}_1$ to $\mathcal{A}_3$, only the targeted operation to gain knowledge about $\vec{C}$ is changed. Hence, the reduced complexity is inherent to the targeted operation.



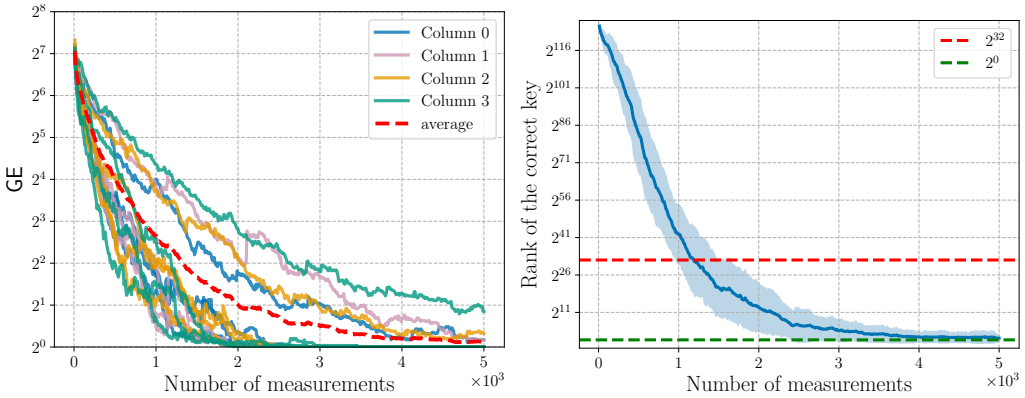
(a) GE of all the bytes. (b) Rank of the correct key.

Figure 10: Attack results for $\mathcal{A}_2$.



(a) GE of all the bytes. (b) Rank of the correct key.

Figure 11: Attack results for $\mathcal{A}_3$.



(a) GE of all the bytes. (b) Rank of the correct key.

Figure 12: Attack results for $\mathcal{A}_4$.

Table 1: Summary of all data complexity for all the investigated adversaries.

Adversary	Direct key recovery	Key enumeration
$\mathcal{A}_1$	$10 \cdot 10^3$	$3 \cdot 10^3$
$\mathcal{A}_2$	$> 50 \cdot 10^4$	$20 \cdot 10^4$
$\mathcal{A}_3$	$3 \cdot 10^3$	$1.1 \cdot 10^3$
$\mathcal{A}_4$	$4 \cdot 10^3$	$1.2 \cdot 10^3$
$\mathcal{A}_5$	$2 \cdot 10^3$	$0.9 \cdot 10^3$

Namely, the information exploited by $\mathcal{A}_1$ is coming from a few cycles with 24 random bits manipulated in parallel (Figure 6). The information exploited by $\mathcal{A}_3$ comes from multiple operations in `MixColumns`, leading to more extractable and independent signal.

The effect of shuffling can be measured by moving from $\mathcal{A}_2$ to $\mathcal{A}_3$. If no information is exploited about the permutation, the information per share is (roughly) divided by the number of possible indexes (i.e., $|o_1|$ and $|o_2|$). As a result, the noise on each share is increased by a factor 4, leading to a data complexity 16 larger (because of masking). This factor 16 fits the difference between both attacks (that respectively require $1.1 \cdot 10^3$ and $2 \cdot 10^4$ traces) reasonably well.

Eventually, omitting some terms in the sums (i.e., moving from $\mathcal{A}_3$ to $\mathcal{A}_4$) only marginally increases the data complexity since the secret permutation is recovered with 98% of accuracy in our experiments. So it mostly improves the attack time complexity by a factor 8. (In case of less accurate information extraction about the permutation, this change would be more of a tradeoff between an improved time or data complexity).

We recall that all the attacks described in this section are exploiting the (full) knowledge of the multiplicative mask r_m that is recovered in a single trace. With this knowledge, the affine masking is turned in a Boolean masking with two shares (given a multiplication by r_m in $\mathsf{GF}(2^8)$ is performed). Overall, the success of these attacks is mostly due to the limited amount of noise on the target device, which naturally implies that the "noise amplification" that masking provides only has a limited impact on the final complexities.

We insist that such a low noise level was already pointed out in the technical report of [BKPT] where the analysis was made on a different target (`STM32` with a `Cortex-M4`). Given the noise level that we measured, extrapolations based on [DFS15] show that 18 shares would be necessary to prevent attacks with up to 2^{64} measurements. So our results mostly show that ensuring security in such low-cost devices is challenging, and a first step would probably be to try increasing the – algorithmic or physical – noise so that masking becomes more effective (which would also have a positive impact on shuffling [VMKS12]).

3.5 Comparison with the ANSSI measurements

In view of the previous results, a natural question is whether they are due to implementation and setup differences with the ones used in the ANSSI leakage assessments. After we communicated our results to them, the ANSSI team kindly accepted to share their measurements so that we could answer this question. It turns out their (STM32) implementation has some minor difference with the one we analyzed (in particular the non-constant time behavior mentioned in Footnote 5). But for the rest, it gives rise to the same POIs as we exploit in our attacks. We therefore repeated the same attacks as in this section using the traces from [BKPT] and could recover the full key with approximately 5,000 traces and an enumeration power of 2^{32}. The only noticeable difference is a slightly lower SNR, especially for the same four bytes as we observed (so enumeration is actually more critical in the case of the measurements used by the ANSSI team). We assume this difference is mostly due to an improved measurement setup.

4 Machine learning based evaluations

As mentioned in introduction, machine learning / deep learning algorithms are increasingly popular tools to analyze side-channel resistance. In this section, we question whether such tools can improve the attacks in the previous section. Since one of the claimed advantages of machine learning / deep learning algorithms is their potential for automation, we will consider two experimental settings. First, a close to black box setting where the neural network is fed with the same POIs as the attacks in the previous section. Next a dissection attack where the neural network is used to extract information about specific targets. We analyze the performance of a `MLP` as a first step and leave the investigation of other machine learning / deep learning tools as a scope for further investigations.

4.1 Methodology

In contrast with $g\mathsf{T}$s that only require one pass on the data to estimate means and covariances, meta-parameters must be tuned for a `MLP`, which is typically done with (k-fold) cross-validation. Parameters such as the number of layers, the learning rate and the activation function were selected accordingly (with $k = 5$). We allowed a maximum number of 3 hidden layers with up to 500 nodes, a learning rate between 0.01 and 0.2, and *sigmoid*, *relu* and *tanh* activation layers. We selected the best set of parameters with a random grid search performed independently for each of the investigated problems. The number of epochs was fixed to 250. Once the meta-parameters are found, the `MLP` is trained again with all the training samples. Practically, the open source library `Keras` was used as front end to `TensorFlow`.[6] The training was performed on 48 available cores.

4.2 Close to black-box evaluations

Using the previous methodology, we tried to perform a close to black box analysis of the ANSSI implementation. However, despite feeding an `MLP` with the same POIs as for the previous attacks, none of the attacks succeeded. To gain intuition about this failure, we therefore ran some simulated experiments. Namely, we compared a $g\mathsf{T}$ adversary similar to the previous section to an `MLP` attack in two scenarios:

(a) Boolean masking.

(b) Affine masking.

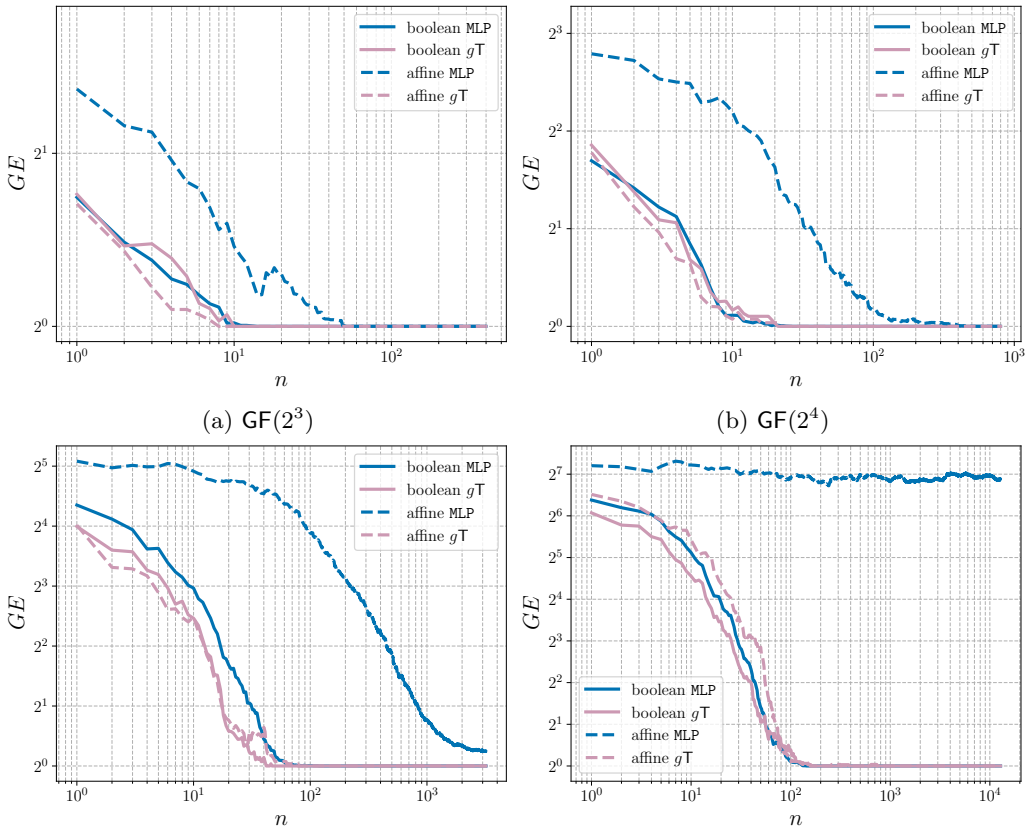
Figure 13: Simulation settings.

1. The first scenario corresponds to a Boolean masking (as illustrated in Figure 13a). There, the secret value x is XORed with a single random byte r. In order to perform the attack, the $g\mathsf{T}$ adversary needs knowledge of x and r associated to the leakage samples $l_\star$. He can then estimate the distributions of l_1 and l_2 individually. The attack is performed by summing over all the possible r's as shown in Equation (6). The training of the `MLP` is performed without the knowledge of the masks and therefore called black box. It only needs the label x corresponding to the $l_\star$'s.

[6] https://keras.io

2. The second scenario corresponds to an affine masking with *known* multiplicative mask r_m (Figure 13b). It is therefore similar to the previous practical experiments where r_m was recovered with a SPA. Despite trivariate, the $g\mathsf{T}$ adversary in this setting is essentially the same as the one targeting the previous Boolean masking scheme: she just needs to compute $x \otimes r_m$ in the attack phase. Indeed, if r_m is known the only remaining protection is a Boolean masking with two shares. By contrast, the MLP adversary does not change, raising the question whether she will be able to automatically learn the field multiplication.

Note that in case (1), an additional useless dimension is added so that the MLP can take exactly the same inputs in both scenarios (for the $g\mathsf{T}$, this has no impact given an accurate profiling [LPB+15]). Furthermore, and in order to help interpreting the results, we investigated attacks against encodings in Galois fields of sizes ranging from three to eight bits (hoping to observe a gradual degradation of the MLP attack in the second scenario as the field size increases). The number of traces used for the profiling is fixed to $2,000$ per intermediate values, which is a similar profiling effort as in the previous practical experiments. We use the Hamming weight leakage model and the SNR is set to 10 (i.e., a high value since we are interested in the hardness of learning the fied multiplication).



(a) GF(2^3) (b) GF(2^4)

(c) GF(2^6) (d) GF(2^8)

Figure 14: GEs of MLP and $g\mathsf{T}$ in front of Boolean masking and affine masking with known multiplivative mask for SNR =10 with n traces (estimated over 100 experiments).

The GEs obtained for each of the scenarios and adversaries are shown in Figure 14. The first observation is that for all the experiments, the $g\mathsf{T}$ attacks against the Boolean

masking and the ones against the affine masking with known multiplicative masks show similar performances. So as expected, a Boolean masking scheme and an affine masking scheme with known multiplicative mask are equivalent for such an adversary (who can easily perform the multiplication by r_m with perfect knowledge of r_m).

The second observation is that the efficiency of the MLP attacks compared to the one of $g\mathsf{T}$ differs in function of the scenario. In the context of a Boolean encoding, the MLP adversary performs as good as the $g\mathsf{T}$ without requiring knowledge of the randomness during the profiling. This makes the MLP an appealing solution when the randomness cannot be controlled by the evaluator. It implies that the MLP is able to efficiently estimate the Gaussian mixture model with a low noise (i.e., a SNR of 10).

By contrast, and contrary to what is observed for a $g\mathsf{T}$ adversary, the MLP requires more traces to recover a key word of the affine masking scheme than for the Boolean masking, despite the known multiplicative mask. Furthermore, this trend clearly increases with the size of the field in which the multiplications are performed. For example, if the operations are performed in $\mathsf{GF}(2^4)$, the secret can be recovered with 5 times more traces than with a Boolean masking; in $\mathsf{GF}(2^8)$, hundred times more samples do not allow to recover the secret byte. We conclude that the problem of learning a field multiplication based on side-channel leakages is not trivial (despite it is trivial to perform manually).

Overall, our results suggest the attack of the ANSSI implementation with less than 2,000 traces (and similar time complexities and profiling efforts as ours) as an interesting challenge for black box machine learning based evaluations. While such experiments are admittedly no proofs, we believe the general intuition they illustrate is correct and important: even if the field multiplications of an affine masking scheme can eventually be learned in a black box manner, it is likely that it will imply significant profiling overheads while manually exploiting the knowledge of an additive mask is actually trivial.

So these simulated experiments allow an interesting illustration of the pros and cons of machine learning in the context of side-channel security evaluations. On the positive side, they are handy to automate some parts of the attacks such as the learning of a Gaussian mixture with low noise (which is interesting since performed without mask knowledge). On the negative side, they remain limited in a fully black box setting. In this respect, the ANSSI implementation provides a realistic example where the gap between a black box security evaluation and a countermeasures' dissection is actually wide.

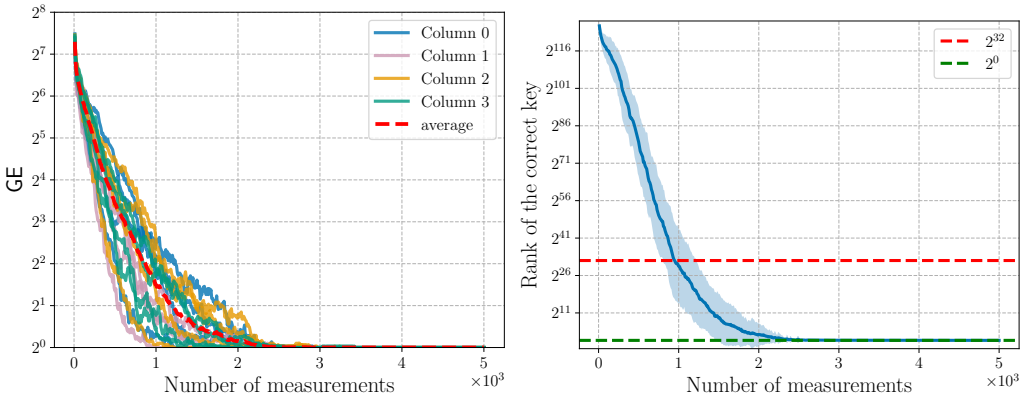
Based on this observation, the next section concludes the paper by illustrating how to integrate the MLP distinguisher as a useful ingredient to improve our countermeasures' dissection while still exploiting the knowledge of the target implementation.

4.3 Dissection with MLP

As discussed in Section 2, performing an efficient countermeasures' dissection based on $g\mathsf{T}$ requires a few (independence) assumptions. Despite the experiments in Subsection 3.3 show that these assumptions hold to a sufficient extent so that powerful attacks can be mounted based on them, it remains that avoiding such hypotheses may lead to improved attacks (e.g., a better modeling of the leakages).

In the following, we run a MLP on each of the shares individually, so that the MLP replaces the computation of Equation (9) and Equation (10) given the same leakage samples. As a result, it estimates independently the probability of each share while taking into account the leakage of the permutation on MixColumns. This adversary is denoted as $\mathcal{A}_5$.

The resulting attack performances are reported in Figure 15 where only $2 \cdot 10^3$ traces are needed to recover the full 128-bit key. Furthermore, the correct key is ranked in the 2^{32} most probable ones with only 900 measurements, which improves over $\mathcal{A}_3$. This gain presumably comes from the absence of leakage assumption for the MLP. In particular, it



(a) GE of all the bytes.

(b) Rank of the correct key.

Figure 15: Attack results for $\mathcal{A}_5$.

does not assumes independence between the leakage of the permutation and the data, nor a Gaussian distribution. Interestingly, we note from Figure 15a that all the bytes are now roughly equivalent in front of $\mathcal{A}_5$, while it was not the case for $\mathcal{A}_3$ (Figure 11a), so we assume that some bytes were hard to recover due to assumption errors by $\mathcal{A}_3$.

This last attack confirms that machine learning algorithms can be used in complement with other approaches to evaluate protected implementations in a close to worst-case manner. It also shows that claims of better automation do materialize in advanced scenarios, since the manipulation of the permutation and shares leakages in this section is simpler than in Section 3, which is in line with previous results (e.g., [CDP17]). Yet, these positive results have to be moderated by the inherent limitations of black box security evaluations that experiments such as in Subsection 4.2 illustrate.

5 Conclusion (how not to interpret our results)

Considering the low-complexity of the attacks put forward in this paper, a tempting conclusion could be that the open approach chosen for the ANSSI implementation is too risky. In our opinion, the correct conclusion is the exact opposite. Namely, the ANSSI implementation puts forwards the (significant) risks of overstated security that closed source designs evaluated in a black box manner carry.

Concretely, the fine grain analysis we provide was only possible (under the time constraints of evaluators and researchers) because of the knowledge of the target implementation. Hence, as our understanding of side-channel countermeasures progresses, preventing such a fine grain analysis can still postpone the finding of some attacks (by the time needed to reverse engineer a single implementation), but it is more and more at the cost of the long-term security guarantees that an open approach leveraging sound physical assumptions and mathematical amplification can offer (see [Sta19] for a discussion).

As far as designs are concerned, we believe it is an important long-term goal to evolve towards physical security based on well-understood solutions surviving public audits. The collaboration of public agencies is of great help for this purpose, in order to gradually increase the security level of public implementations and identify relevant targets.

As far as evaluations are concerned, we believe the analogy with cryptographic research is worth consideration. Namely while automation (e.g., with machine learning / deep learning) is perfectly valid as an attack tool and has lead to several positive results in recent years, such tools may have limited coverage (and therefore limited relevance) in the

context of black box evaluations. In order words, while an adversary is perfectly happy to see a single attack succeeding, a designer has to argue that no attack is possible: a challenge for which a theoretically founded approach is likely to remain the way to go. Interestingly, a similar view has been recently put forward in a CHES 2019 invited talk by Helena Handschuh (on *RISCV and Security: how, when and why?*) [Han19].

Finally, we insist that the weak security levels we put forward originate from the general difficulty to secure a COTS (e.g., ARM Cortex) device rather than from flaws in the ANSSI library. Precisely, the main weakness of the implementation we analyzed is its limited noise, which strongly reduces the impact of both masking and shuffling against countermeasures' dissection. We see no obvious fix to improve the worst-case security level in similar 32-bit devices without significantly increasing the number of shares and leveraging algorithmic noise as much as possible. Given that such architectures are generally considered as good candidates for deployment in the IoT, this confirms that physical security is unlikely to be possible without significant overheads in such COTS devices.

Acknowledgement The authors thank the members of the ANSSI team for their constructive and useful feedback about our results, as well as for providing us the traces they used for their preliminary leakage assessment. François-Xavier Standaert is a Senior Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the European Union (EU) through the ERC project 724725 (acronym SWORD) and the H2020 project 731591 (acronym REASSURE).

References

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, 2017.

[BGNT18] Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Teglia. Multivariate high-order attacks of shuffled tables recomputation. *J. Cryptology*, 31(2):351–393, 2018.

[Bis07] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

[BKPT] Ryad Benadjila, Louiza Khati, Emmanuel Prouff, and Adrian Thillard. https://github.com/ANSSI-FR/SecAESSTM32.

[BP19] Elie Burszstein and Jean-Michel Picot. A hacker guide to deep learning based side channel attacks , Defcon 2019. pages https://elie.net/talk/a--hackerguide--to--deep--learning--based--side--channel--attacks/, 2019.

[BSW00] Alex Biryukov, Adi Shamir, and David A. Wagner. Real time cryptanalysis of A5/1 on a PC. In *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[DK18] Daniel Dinu and Ilya Kizhvatov. EM analysis in the iot context: Lessons learned from an attack on thread. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):73–97, 2018.

[DKM+15] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2015.

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2008.

[FMPR10] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 262–280. Springer, 2010.

[GdKGM+08] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE classic. In *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2008.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An efficient side-channel protected AES implementation with arbitrary protection order. In *CT-RSA*, volume 10159 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2017.

[GPSS18] Benjamin Grégoire, Kostas Papagiannopoulos, Peter Schwabe, and Ko Stoffelen. Vectorizing higher-order masking. In *COSADE*, volume 10815 of *Lecture Notes in Computer Science*, pages 23–43. Springer, 2018.

[GRO18] Joey Green, Arnab Roy, and Elisabeth Oswald. A systematic study of the impact of graphical models on inference-based attacks on AES. In *CARDIS*, volume 11389 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2018.

[GS15] Vincent Grosso and François-Xavier Standaert. Asca, SASCA and DPA with enumeration: Which one beats the other and when? In *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 291–312. Springer, 2015.

[Han19] Helena Handschuh. Riscv and security: how, when and why? (invited talk), CHES 2019. page , 2019.

[HGM+11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptographic Engineering*, 1(4):293–302, 2011.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.

[HZ12] Annelie Heuser and Michael Zohner. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2012.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[JS17] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643. Springer, 2017.

[LMBM13] Liran Lerman, Stephane Fernandes Medeiros, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2013.

[LPB+15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *COSADE*, volume 9064 of *Lecture Notes in Computer Science*, pages 20–33. Springer, 2015.

[LR10] Kerstin Lemke-Rust. On security evaluation testing. *https://www.lorentzcenter.nl/lc/web/2010/383/presentations/LemkeRust.pdf*, 2010.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[MBKP11] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx virtex-ii fpgas. In *ACM Conference on Computer and Communications Security*, pages 111–124. ACM, 2011.

[MMSS19] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited or why proofs in the robust probing model are needed. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):256–292, 2019.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[PSK+18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In *SPACE*, volume 11348 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2018.

[RSV09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: why time also matters in DPA. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.

[RSWO17] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. Iot goes nuclear: Creating a zigbee chain reaction. In *IEEE Symposium on Security and Privacy*, pages 195–212. IEEE Computer Society, 2017.

[SM16] Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptographic Engineering*, 6(2):85–99, 2016.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[Sta19] François-Xavier Standaert. Towards and open approach to secure cryptographic implementations (invited talk), EUROCRYPT 2019. pages xv, https://www.youtube.com/watch?v=KdhrsuJT1sE, 2019.

[SVO+10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.

[TWO13] Michael Tunstall, Carolyn Whitnall, and Elisabeth Oswald. Masking tables - an underestimated security risk. In *FSE*, volume 8424 of *Lecture Notes in Computer Science*, pages 425–444. Springer, 2013.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[ZS19] Yuanyuan Zhou and François-Xavier Standaert. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized resnet model for side-channel attacks. *Journal of Cryptographic Engineering*, Apr 2019.

[ZYSQ13] Yuanyuan Zhou, Yu Yu, François-Xavier Standaert, and Jean-Jacques Quisquater. On the need of physical security for small embedded devices: A case study with COMP128-1 implementations in SIM cards. In *Financial Cryptography*, volume 7859 of *Lecture Notes in Computer Science*, pages 230–238. Springer, 2013.