Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint

Qian Guo[1], Vincent Grosso[2], François-Xavier Standaert[3]

[1] Department of Informatics, University of Bergen, Norway
[2] Digital Security Group, Radboud University, Nijmegen, The Netherlands
[3] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium
e-mails: fywzguoqian,grosso.vincent@gmail.com;fstandae@uclouvain.be

Abstract. One important open question in the field of side-channel analysis is to find out whether all the leakage samples in an implementation can be exploited by an adversary, as suggested by masking security proofs. For concrete attacks exploiting a divide-and-conquer strategy, the answer is negative (i.e., only the leakages corresponding to the first/last rounds of a block cipher can be exploited). Soft Analytical Side-Channel Attacks (SASCA) have been introduced as a powerful solution to mitigate this limitation. They represent the target implementation and its leakages as a code (similar to a Low Density Parity Check code) that is then decoded thanks to belief propagation. Previous works have shown the low data complexities that SASCA can reach in practice (at the cost of a higher time complexity). In this work, we revisit these attacks by modeling them with a variation of the Random Probing Model used in masking security proofs, that we denote as the Local Random Probing Model (LRPM). Our study establishes interesting connections between this model and the erasure channel used in coding theory, leading to the following benefits. First, the LRPM allows assessing the security of concrete implementations against SASCA in a fast and intuitive manner. We use it to confirm that the leakage of any operation in a block cipher can be exploited, although the leakages of external operations dominate in known-plaintext/ciphertext attack scenarios. Second, we show that the LRPM is a tool of choice for the (nearly worst-case) analysis of masked implementations in the noisy leakage model, taking advantage of all the operations performed, and leading to new possibilities of tradeoffs between their amount of randomness and physical noise level.

1 Introduction

A recent line of works started the investigation of masking security proofs as an ingredient of concrete side-channel security evaluations [7, 8, 13, 22]. These papers essentially concluded that for all the "accessible" leakage samples of an implementation (i.e., corresponding to a target value that can be guessed by the adversary) these proofs are tight and can be matched by a standard DPA [17]. Yet, the security bounds (e.g., in [7, 8]) actually suggest that the adversary's success probability grows linearly in the circuit size. The latter implies that all the leakage samples of an implementation should be exploitable, independent of whether they can be exploited via a divide-and-conquer strategy.

Contribution. In this paper, we tackle this important question of whether any leakage sample in an implementation can be efficiently exploited by a determined adversary. We answer the question by proposing a model for the Soft Analytical Side-Channel Attacks (SASCA) introduced in [29] and further developed in [12, 13]. The latter attacks are natural candidates for capturing the circuit size parameter of masking security proofs since they aim at exploiting the total amount of information leaked by an implementation. Our model – that we denote as the Local Random Probing Model (LRPM) – specializes the Random Probing Model used in these proofs to local decoding rules similar to those used in SASCA. It enables interesting connections with the well-known erasure channel used in coding theory, leading to the following concrete benefits.

First, the LRPM allows significantly speeding up the evaluation of SASCA (which are computationally intensive). For example, we show that we can reproduce the results of a SASCA against an AES implementation from [12, 29] in seconds (rather than hours) of computations. These results confirm that the leakage of any operation within a block cipher can be exploited. More precisely, and despite the first and last rounds' leakages are more useful than the inner round ones in known-plaintext/ciphertext attacks, inner rounds' leakages cannot be neglected in other cases (e.g., in unknown-plaintext/ciphertext attacks).

Second, the LRPM allows revisiting masking security proofs and the evaluation of actual implementations in a flexible manner. More precisely, the RMP was so far mostly used as a technical ingredient in order to connect (abstract) probing security and (concrete) noisy leakage security [7, 14, 22]. We show that combining it with the factor graph describing an implementation and local decoding rules, we can analyze security in the noisy leakage model using easy-to-estimate approximations of the leakages in hand in a nearly worst-case manner.[1] The latter is particularly interesting for two main reasons. First, it shows that there may be a gap between the concrete security of simple gadgets and the one of complex combinations of gadgets such as S-boxes, cipher rounds, ... For example, we exhibit new attack paths in masked implementations based on the the merging of some nodes in the factor graph that are connected by linear operations. Second, recent works have optimized masking schemes in order to minimize their randomness requirements [3, 4]. Our results recall that masked implementations can be viewed as a tradeoff between physical noise and mathematical randomness, so at some point randomness minimization may become detrimental to security. The latter has been put forward experimentally in [2, 13] by using concrete (horizontal and SASCA) attacks. But in view of the heuristic nature and high time complexity of these attacks, they can hardly serve as a solution to discuss the noise vs. randomness tradeoff in a systematic manner. In this context, the LRPM could be used as a tool to guide the design of new implementations optimized globally and jointly based on these ingredients.

[1] Precisely, we bound the information leakage that can be obtained by decoding the factor graph of an implementation using the belief propagation algorithm. A worst-case attack would apply Bayes on the full (multivariate) leakage distribution of the implementation, which usually leads to unrealistic time complexities [13].

Paper structure. After providing the necessary background in Section 2, our results are structured in three main parts. We start by outlining the general ideas behind our modeling in an intuitive manner in Section 3. Its goal is to describe easy-to-reproduce toy examples of implementations and security evaluations. Next our first main contribution is in Section 4 where we introduce the LRPM and use it to analyze the practically-relevant case study of an AES implementation. As previously mentioned, it allows us to considerably speed up the analysis of SASCA, and to clarify the extent to which all the leakage samples in an implementation can be exploited by an adversary. Finally, we present our second main contribution in Section 5 where we show how the LRPM applied to concrete masked implementations can be used to exhibit new security threats (e.g., due to a so-far unreported lack of refreshing of linear operations), and to analyze their (nearly) worst-case security and noise vs. randomness tradeoff.

2 Background

We first describe the different technical tools needed for our modeling.

2.1 Template attacks and MI metric

Template Attacks (TA) are a standard tool for extracting information from physical side-channels. They model the leakages using a set of distributions corresponding to intermediate computations in the target implementation. In the seminal TA paper [5], Chari et al. use normal distributions for this purpose (but any distribution is eligible). For an adversary targeting an intermediate value y, the leakage Probability Density Function (PDF) is then evaluated as:

$$\mathsf{f}[L = l|Y = y] := \mathsf{f}[l|y] \sim \mathcal{N}(l|\mu_y, \sigma_y^2).$$

This approach requires estimating the sample means and variances for each value y (and mean vectors / covariance matrices in case of multivariate attacks).[2] As a result, concrete attacks usually extract information for target values of 8 to 32 bits (so that it is possible to build the so-called "templates" $\mathsf{f}[l|y]$ for each y). In a divide-and-conquer attack, one additionally requires that the targets only depend on 8 to 32 key bits (so that the models for an enumerable part of the key can be tested exhaustively). Based on such a leakage model, the probability of a candidate y given a leakage l is directly computed thanks to Bayes:

$$\Pr[y|l] = \frac{\mathsf{f}[l|y]}{\sum_{y^* \in \mathcal{Y}} \mathsf{f}[l|y^*]}.$$

The amount of information collected on y is then usually measured in terms of the mutual information between the random variables Y and L [27]:

$$\mathrm{MI}(Y; L) = \mathrm{H}[Y] + \sum_{y \in \mathcal{Y}} \Pr[y] \sum_{l \in \mathcal{L}} \mathsf{f}[l|y]. \log_2 \Pr[y|l].$$

[2] Alternatively, leakage models can also be built by exploiting linear regression [26].

As discussed in [8] such a metric can be used to bound the success rate of a divide-and-conquer side-channel attack. Considering the typical case where the target intermediate value y is the XOR between a (known) plaintext byte x and an (unknown) key byte k, we have $\mathrm{MI}(K; X, L) = \mathrm{MI}(Y; L)$. The number of traces N required to perform a key recovery can then be bounded by:

$$N \geq \frac{\mathrm{H}[K]}{\mathrm{MI}(K; X, L)} = \frac{\mathrm{H}[K]}{\mathrm{MI}(Y; L)}. \quad (1)$$

Note that in practice, the modeling of the leakage PDF is prone to (estimation and assumption) errors, which may reduce the amount of information extracted: see for example the discussion about leakage certification in [9]. Our following investigations are orthogonal to this modeling issue and just require to be fed with the best possible approximation of the MI metric, just as for the worst-case evaluation of any (e.g., divide-and-conquer) side-channel attack.

2.2 Soft Analytical Side-Channel Attacks

While TA aim to optimally extract information from actual leakage samples thanks to an accurate leakage model obtained though profiling, they are still limited in two important directions. First, they can only target the leakage of target intermediate computations that depend on an enumerable part of the key. This typically implies that only the leakage of the first/last rounds of a block cipher implementation can be exploited in this way [18]. Second, estimating the optimal (mixture) model for a masked implementation becomes prohibitively expensive as the number of shares increases [13]. SASCA were first introduced as a solution to mitigate the first issue [12, 29], and were recently considered as an efficient heuristic to deal with the second one [2, 13]. Roughly, they work by describing the target implementation and its leakages in a way similar to a Low-Density Parity Check code (LDPC) [10], which the adversary then tries to decode using posterior probabilities obtained thanks to a TA on all the intermediate values of the implementation. More precisely, they work in three steps:

1. *Construction.* The cipher is represented as a so-called "factor graph" with two types of nodes and bidirectional edges. First, variable nodes represent the intermediate values. Second, function nodes represent the a-priori knowledge about the variables (e.g., the known plaintexts and leakages) and the operations connecting the different variables. Those nodes are connected with bidirectional edges that carry two types of messages (i.e., propagate the information) through the graph: the type q messages are from variables to functions and the type r messages are from functions to variables (see [16] for more details).

2. *Information extraction.* The probabilities provided by performing TA on all the intermediate variables of the target implementation are added as function nodes to the factor graph. For this purpose, one can use exactly the same profiling tools as in the divide-and-conquer case [12] (but for more variables).

3. *Decoding.* Similar to LDPC codes, the factor graph is decoded using a Belief Propagation (BP) algorithm [21], which essentially iterates the local propagation of the information about the variable nodes of the target implementation.

Since one goal of our work is to bound the security of an implementation against SASCA, in order to avoid the hassle of actually launching the BP algorithm but also to gain better intuitions about the parameters influencing its success, we defer a description of this algorithm to Appendix A.

2.3 Basics in coding theory

This subsection briefly recalls some basic terminology from coding theory.

Definition 1 (Linear code). *We define an $[\eta, \kappa]_q$ linear code $\mathcal{C}$ as a linear subspace over $\mathbb{F}_q$ of length η and dimension κ, and its (information) rate, denoted as R, is $\frac{\kappa}{\eta}$. The redundancy of the code is then worth $\eta - \kappa$.*

We will simply write $[\eta, \kappa]$ linear code if there is no ambiguity.

Definition 2 (Parity-check matrix). *Let $\mathcal{C} \subseteq \mathbb{F}_q^\eta$ be a linear code of dimension κ. If $\mathcal{C}$ is the kernel of a matrix $\mathbf{H} \in \mathbb{F}_q^{(\eta-\kappa)\times\kappa}$, that is:*

$$\mathcal{C} = \ker(\mathbf{H}) = \left\{ \mathbf{v} \in \mathbb{F}_q^\eta : \mathbf{H}\mathbf{v}^\mathrm{T} = \mathbf{0} \right\},$$

then, we say that $\mathbf{H}$ is a parity-check matrix of the linear code $\mathcal{C}$.

Definition 3 (LDPC codes [10]). *A Low-Density Parity-Check (LDPC) code is a linear code $\mathcal{C}$ with a very sparse parity-check matrix $\mathbf{H}$.*

We note that LDPC codes have been an important topic in the coding community during the past twenty years due to their capacity-approaching feature (i.e., the fact that their decoding performance is close to the channel capacity).

Tanner Graphs. One core feature of LDPC codes is their efficient decoding thanks to the BP algorithm on the *Tanner graph* [23, 28] representation of the code. A Tanner graph is a bipartite graphical representation of the code, where each edge corresponds to a non-zero element in the parity-check matrix $\mathbf{H}$. In general, sparse matrices $\mathbf{H}$ are expected to lead to sparse Tanner graphs.

Erasure Channel / Random Probing Model (RPM). An erasure channel is a communication channel model used in coding theory and information theory. In this model, a transmitter sends a value (usually a bit) and the receiver either receives the bit in full or it receives a message that the bit was not received ("erased" or $\perp$). It exactly corresponds to the definition of RPM in [7].

3 A toy example of unprotected implementation

In this section, we introduce our modeling by evaluating the side-channel security of a toy unprotected implementation. We first show how to analyze its leakage with state-of-the-art tools exploiting a divide-and-conquer strategy. We then evaluate its security against a SASCA exploiting all its target intermediate computations, by exploiting tools from the coding theory literature.

3.1 Target implementation

We consider an implementation setting with four 8-bit plaintext bytes x_1, x_2, x_3 and x_4, four 8-bit key bytes k_1, k_2, k_3 and k_4, and the leaking computations in Figure 1, with S an 8-bit S-box and $\rightsquigarrow l_y$ denoting the generation of a leakage l_y when computing an intermediate result y within the implementation.

Divide-and-conquer targets:

- $y_1 := x_1 \oplus k_1 \rightsquigarrow l_{y_1}$,
- $y_2 := x_2 \oplus k_2 \rightsquigarrow l_{y_2}$,
- $y_3 := x_3 \oplus k_3 \rightsquigarrow l_{y_3}$,
- $y_4 := x_4 \oplus k_4 \rightsquigarrow l_{y_4}$,
- $z_1 := \mathsf{S}(y_1) \rightsquigarrow l_{z_1}$,
- $z_2 := \mathsf{S}(y_2) \rightsquigarrow l_{z_2}$,
- $z_3 := \mathsf{S}(y_3) \rightsquigarrow l_{z_3}$,
- $z_4 := \mathsf{S}(y_4) \rightsquigarrow l_{z_4}$,

1-round SASCA targets:

- $v_1 := z_1 \oplus z_2 \rightsquigarrow l_{v_1}$,
- $v_2 := z_3 \oplus z_4 \rightsquigarrow l_{v_2}$,
- $w_1 := \mathsf{S}(v_1) \rightsquigarrow l_{w_1}$,
- $w_2 := \mathsf{S}(v_2) \rightsquigarrow l_{w_2}$,

2-round SASCA targets:

- $a := w_1 \oplus w_2 \rightsquigarrow l_a$,
- $b := \mathsf{S}(a) \rightsquigarrow l_b$.

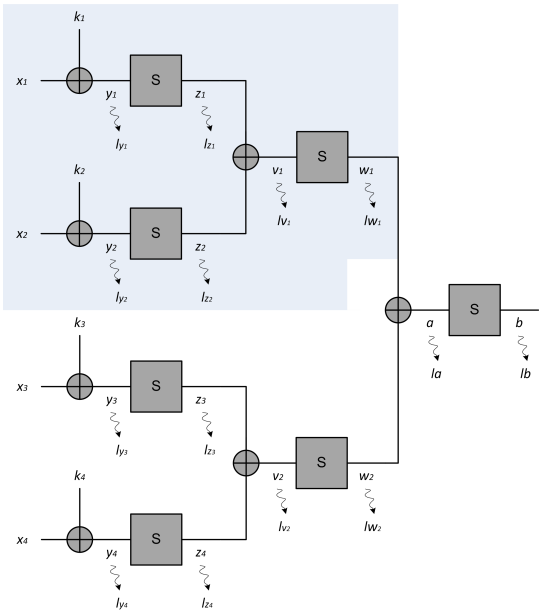

Fig. 1. Toy unprotected implementation.

In this setting, a divide-and-conquer adversary targets the four key bytes k_1 to k_4 independently and therefore exploits all the leakages that only depend on them (e.g., l_{y_1} and l_{z_1} when targeting k_1). The analytical adversary additionally exploits the leakages l_{v_1}, l_{v_2}, l_{w_1}, l_{w_2}, l_a and l_b that depend on several key bytes. More precisely, we will consider a so-called 1-round analytical adversary that (only) exploits l_{v_1} and l_{w_1} (in the grey box of the figure) or l_{v_2} and l_{w_2}, and a 2-round analytical adversary that exploits all the leakages. We will additionally refer to univariate attacks when the adversary only exploits the S-box output leakages and bivariate attacks when exploiting both the S-box input and output leakages. For simplicity, we further assume that the information leakage on each intermediate computation is identical. Measured with the mutual information metric of Section 2.1, it means that $\mathrm{MI}(Y_1; L_{Y_1}) = \mathrm{MI}(Y_2; L_{Y_2}) = \ldots = \epsilon$. Yet, the following evaluations could naturally capture situations where the amount of information leakage (or the noise level) on each intermediate value differs.

3.2 A divide-and-conquer evaluation

A bivariate divide-and-conquer adversary targeting k_1 to k_4 proceeds as follows. For each (known) plaintext byte x_1, he first combines the leakages l_{y_1} and l_{z_1}, leading to a mutual information leakage of 2ϵ bits on k_1 (using the worst-case Independent Operation Leakage assumption from [13]). He then does the same with the plaintext bytes x_n and the leakages l_{y_n} and l_{z_n} with $n = 2, 3, 4$, leading to the same information on each key byte k_n. Assuming independent and uniformly distributed plaintexts, the (data) complexity N_{dc} of a key recovery attack against each key byte can then be bounded thanks to Equation 1 as $N_{\mathrm{dc}} \geq \left\lfloor \frac{8}{2\epsilon} \right\rfloor$. Taking a value of $\epsilon = 0.1$ bits for example, this means that one can guarantee that complete key recovery requires the observation of at least 40 leakages.

3.3 A SASCA evaluation

In order to characterize the increased amount of information that a SASCA can extract from the example of Figure 1, we model an implementation with a factor graph, its side-channel leakages with the RPM and their exploitation with (local) information propagation rules that can be viewed as a variation of the piling up lemma. As an example of the latter, let us assume a channel with field size 256 (i.e., 8-bit target variables) and a check node v with ℓ neighbors. If the mutual information on the edges e with $e \in [1, \ell]$ is λ_j bytes, then the extrinsic mutual information on edge e_0 is $\prod_{e \in [1,\ell], j \neq e_0} \lambda_j$ bytes.[3] We will denote this combination of a factor graph, the RPM and local information propagation rules as the Local Random Probing Model (LRPM), and specify it in Sections 4.1, 4.2, 4.3.

In order to take advantage of this model, one first needs to build the factor graph corresponding to the example of Figure 1. An example of such graph is illustrated in Figure 2 for a simple attack and is based on two principles:

On the one hand, and most importantly, there are two types of leakage (function) nodes in the graph. The first ones are "continuous" leakage nodes and are represented in dark gray (with the $\mathcal{L}_{\mathsf{c}}$ symbol). They correspond to target intermediate variables that can be exploited via a divide-and-conquer attack and for which the leakage is accumulated based on an additive channel (thanks to the uniform plaintext assumption). The second ones are "one-shot" leakage nodes and are represented in light gray (with the $\mathcal{L}_1$ symbol). They correspond to the target intermediate variables that cannot be exploited via a divide-and-conquer attack and for which the leakage is exploited thanks to the BP algorithm.

On the other hand, whenever two intermediate variables X and $\mathsf{F}(X)$ are connected through a bijection F (e.g., the identity function, an S-box or the Xtimes function in the AES), the information on X (i.e., $\mathrm{MI}(X; L)$) can be turned into information on $\mathsf{F}(X)$ (i.e., $\mathrm{MI}(\mathsf{F}(X); L)$) and vice versa. Therefore, we can treat these variables as a single node and accumulate their information based on an additive channel. This combination of targets is reflected by the value before

[3] We use the λ notation rather than the ϵ notation in order to reflect the fact that the leakage is now expressed in bytes - or more generally $\mathbb{F}_q$ elements.

the $\mathcal{L}$ symbols in Figure 2. For example, the $2\mathcal{L}_{\mathsf{c}}$ node attached to k_1 means that there are two continuous channels (i.e., the S-box's input and output) leaking information on this value. Similarly, the $1\mathcal{L}_{\mathbf{1}}$ node attached to $w_n(i)$ and $b(i)$ means that there is a single one-shot leakage attached to each $w_n(i)$ and $b(i)$ intermediate variable (with $i \in [1, N]$ the index of the manipulated plaintext). So this graph corresponds to an attack exploiting bivariate leakages for the divide-and-conquer targets, and univariate leakages for the SASCA targets.

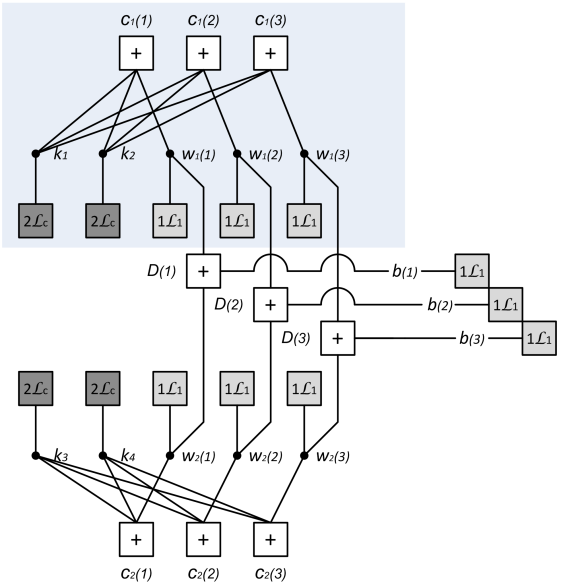

Fig. 2. Example of factor graph corresponding to Figure 1.

From a coding theory viewpoint, these two types of leakages have a simple interpretation. The first (information-accumulating) nodes can be viewed as "noise reduction" while the second ones can be viewed as a "code expansion".

The next step is to compute the information leakage on each node. For the graph of Figure 2, and assuming a mutual information of ϵ bits on every intermediate computation (as in the previous subsection), the leakage on k_n for $n \in [1, 4]$ is bounded by $2N\epsilon$ bits (with the factor 2 coming from the combination of the S-boxes' input and output) and $\lambda = \frac{N\epsilon}{4}$ bytes. Similarly, the one-shot leakages on the $w_n(i)$ and $b(i)$ nodes are bounded by ϵ bits and $\lambda = \frac{\epsilon}{8}$ bytes.

Eventually, we need to bound the information leakage in function of the amount of operations exploited by the adversary. For this purpose, we start by considering the 1-round analytical adversary targeting the key bytes k_1 and k_2 (of which the leakages exploited are in the light gray boxes of Figures 1 and 2). We use the following notations. After (t) iterations of the message passing rules

of the BP algorithm (given in Appendix A), the information (in bytes) from the key variable node k_n to the check node c_m is denoted as $p^{(t)}_{k_n \to c_m}$ (as per the notations of Appendix A). Similarly, the information (in bytes) from the check node c_m to the key variable node k_n is denoted $r^{(t)}_{c_m \to k_n}$. Additionally denoting the information bound (in bytes) on the key bytes k_1 or k_2 with $\mathcal{B}^{(t)}$, we can then derive the following set of equations describing the bound:

$$r^{(0)}_{c_m \to k_n} = 0, \tag{2}$$

$$p^{(t)}_{k_n \to c_m} = \frac{N\epsilon}{4} + (N-1) \cdot r^{(t-1)}_{c_m \to k_n}, \tag{3}$$

$$r^{(t)}_{c_m \to k_n} = p^{(t)}_{k_n \to c_m} \cdot \frac{\epsilon}{8}, \tag{4}$$

$$\mathcal{B}^{(t)} = \frac{N\epsilon}{4} + N \cdot r^{(t)}_{c_m \to k_n}. \tag{5}$$

For illustration, a Matlab pseudo-code for this example is given in Appendix B. Taking a value of $\epsilon = 0.1$ bits as in the previous subsection, and using a similar approximation for the data complexity of an analytical attack $N_{\mathrm{a}} \geq \frac{1}{\mathcal{B}^{(\infty)}}$, we can see in the left part of Figure 10 that the bound becomes larger than one for $N_{\mathrm{a}} \geq 27$. Note that the gain of the analytical strategy over the divide-and-conquer one only appears when the number of measurements becomes sufficient for the decoding to become effective. Note also that for N_a such that $\mathcal{B}^{(t)} = 1$, the additional leakage of w_1 is essentially equivalent to the addition of a third continuous channel on the key bytes in this case (since $27 \approx \frac{8}{0.3}$).

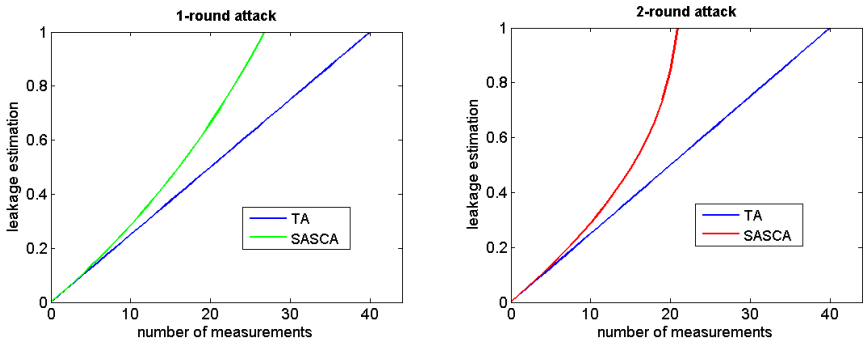

Fig. 3. Leakage bounds for the factor graph of Figure 2.

In this respect, a natural question is whether the same intuition holds when digging further into a cipher's internal operations? We answer this question by analyzing the 2-round adversary described by the following equations:

$$r^{(0)}_{c_m \to k_n} = 0, \tag{6}$$
$$r^{(0)}_{c_m \to w_n} = 0, \tag{7}$$
$$r^{(0)}_{d_m \to w_n} = 0, \tag{8}$$
$$p^{(t)}_{k_n \to c_m} = \frac{N\epsilon}{4} + (N-1) \cdot r^{(t-1)}_{c_m \to k_n}, \tag{9}$$
$$p^{(t)}_{w_n \to c_m} = \frac{\epsilon}{8} + r^{(t-1)}_{d_m \to w_n}, \tag{10}$$
$$p^{(t)}_{w_n \to d_m} = \frac{\epsilon}{8} + r^{(t-1)}_{c_m \to w_n}, \tag{11}$$
$$r^{(t)}_{c_m \to k_n} = p^{(t)}_{k_n \to c_m} \cdot p^{(t)}_{w_n \to c_m}, \tag{12}$$
$$r^{(t)}_{c_m \to w_n} = p^{(t)}_{k_n \to c_m} \cdot p^{(t)}_{k_n \to c_m}, \tag{13}$$
$$r^{(t)}_{d_m \to w_n} = p^{(t)}_{w_n \to d_m} \cdot \frac{\epsilon}{8}, \tag{14}$$
$$\mathcal{B}^{(t)} = \frac{N\epsilon}{4} + N \cdot r^{(t)}_{c_m \to k_n}. \tag{15}$$

Taking again a value of $\epsilon = 0.1$ bits, and using the approximation for the data complexity of an analytical attack $N_{\mathrm{a}} \geq \frac{1}{\mathcal{B}^{(\infty)}}$, we can see in the right part of Figure 10 that the bound becomes larger than one for $N_{\mathrm{a}} \geq 21$. So here as well, we observe that the addition of the leakages of w_1 and v is very close to the addition of two continuous channels on the key bytes (since $20 = \frac{8}{0.4}$).

The latter provides a preliminary confirmation of the security proofs in [7,8] from a coding theory viewpoint, and in particular of the fact that the security of an implementation in the LRPM may indeed decrease linearly with its circuit size Γ (here reflected by the number of leaking operations), independent of whether these operations are exploitable via divide-and-conquer attacks or not.

3.4 Discussion

The next section will provide a more definitive confirmation that our modeling based on the RPM can serve as an excellent (and very fast) predictor of the complexity of SASCA, by studying the practically-relevant case study of an AES implementation. Beforehand, and for completeness, we use our simple examples to briefly discuss the small discrepancies that can occur between (nearly) worst-case analyzes based on the LRPM and concrete SASCA results. The goal of this more heuristic discussion is to serve as background for the interpretation of our next results. For this purpose, we launched experiments against the target implementation of Figure 1 and considered three attacks:

- A divide-and-conquer univariate TA (Template Attack),
- A SASCA additionally exploiting univariate first-round leakages,
- A SASCA exploiting univariate first- and second-round leakages.

Following the usual approach in simulated side-channel attacks, we first considered Hamming weight leakages: for example $l_{y_1} = \mathsf{HW}(y_1) + \rho$ with ρ a Gaussian-distributed noise, from which we extracted a MI value using the tools in [8]. The success rate curves for these attacks are given in the left part of Figure 4. A success rate of 90% is reached after approximately 148 traces for the TA, approximately 80 traces for the 1-round SASCA and approximately 67 traces for the 2-round SASCA. So one can notice that the complexity improvement due to 1- and 2-round SASCA is slightly reduced compared to the factors 2 and 3 that would be predicted based on the equations in the previous subsection.

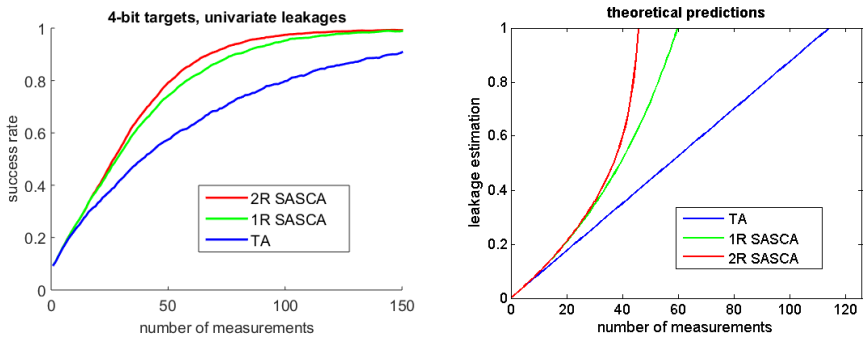

Fig. 4. Experimental results (left) and theoretical predictions with XOR heuristic (right) for TA, 1R and 2R SASCA against the implementation of Figure 1.

The latter is in fact natural since (*i*) the piling-up lemma only provides a bound for the extrinsic mutual information on an edge (see the discussion in [8,13]), and (*ii*) the BP algorithm applied on factor graphs with cycles (such as the ones we consider by mixing multiple plaintexts) is only a heuristic and does not guarantee to extract all the available information. To these general observations, one must add that it is very easy to conceive leakage functions that are informative and would not enable any information propagation through an XOR operation: think for example of 2-bit sensitive values where the leakage of one input is the LSB and the leakage of the other input is the MSB.

The XOR heuristic. Based on these observations, we conclude that the concrete analysis of SASCA (and the evaluation of all the leakages in an implementation) is inevitably slightly heuristic and implementation-dependent. Since our goal is to reduce the cost of analyzing these heuristic and implementation-dependent aspects of SASCA as much as possible, we suggest a simple solution in order to mitigate the impact of these dependencies, which is to investigate the "XOR loss" factor that reduces the extrinsic information leakage propagated through an XOR operation in the factor graph. In the 1-round attack, this typically happens for Equation 4. So a simple approximation is to compare the ideal predictions obtained with this equation and the result of a concrete 1-round at-

tack, and to extract a value $\alpha \leq 1$ (called the XOR loss) corresponding to the ratio between the (ideal) theoretical expectations and the practical results. We can then use this value α to modify the equation 4 as follows:

$$r^{(t)}_{c_m \to k_n} = p^{(t)}_{k_n \to c_m} \cdot \frac{\epsilon}{8} \cdot \alpha, \tag{16}$$

and include exactly the same multiplicative loss factor in Equations 12, 13 and 14 for the 2-round attack. The results obtained with such a heuristic are given in the right part of Figure 4. We reach 90% of mutual information leakage after 103 traces for the TA, 56 traces for the 1-round SASCA and 45 traces for the 2-round SASCA. So the gains observed for the 1-round SASCA ($\frac{148}{80} \approx 1.85$ in practice, $\frac{103}{56} \approx 1.83$ in our predictions) and for the 2-round SASCA ($\frac{148}{67} \approx 2.20$ in practice, $\frac{103}{45} \approx 2.28$ in our predictions) are remarkably close. The absolute values are slightly less accurate (since the bounds based on the mutual information do not capture the "saturation" of the success rate curves), yet provide a reasonable and very fast assessment of the security level against SASCA.[4]

In the following, we will sometimes use a value of $\alpha = 0.9$ (corresponding to the Hamming weight leakages we will use in our experiments), in order to evaluate the impact of this XOR loss in other concrete scenarios.

Additional observations. To conclude this discussion, we launched similar experiments to evaluate other contexts and attack parameters, namely:

Graph extension vs. graph combination. The SASCA of Figure 4 are based on an extended factor graph connecting all the manipulated plaintexts. A simpler variant is to extract information on the key bytes for each plaintext separately (with a smaller factor graph) and to combine the information afterwards. The latter in general leads to lower information extraction, and in the particular case of our toy implementation, achieves a success rate similar to the one of a TA.

Change of leakage function. We repeated the same experiments with various leakage functions (e.g., identify, random) and obtained identical success rates given an adaptation of the noise level in order to keep the same MI.

Impact of the S-box. It is interesting to note that the S-box in Figure 1 is important for the effectiveness of the SASCA. For example, replacing the S-box by an identify function, the first-round intermediate value w_1 is worth $x_1 \oplus k_1 \oplus x_2 \oplus k_2$. As a result, a SASCA guessing only one byte of key will be unable to exploit this information, since w_1 depends on $k_1 \oplus k_2$ rather than on k_1 and k_2. Adding the S-box allows $w_1 = \mathsf{S}(x_1 \oplus k_1) \oplus \mathsf{S}(x_2 \oplus k_2)$ to depend on k_1 and k_2 jointly.

We finally mention that the impact of the noise in a SASCA against an unprotected (e.g., AES) implementation was previously analyzed in [29]. In particular, it was shown in this reference that the gain of a SASCA over a TA in this context is independent of the noise level (given that this noise is sufficient: for too low noise levels, algebraic properties of the leakages can play a role).

[4] An approximation is given by $N_{\mathrm{dc}} \geq \frac{8}{\epsilon}$, $N_{1\mathrm{R}} \geq \frac{8}{\epsilon \cdot (1+\alpha)}$ for the 1-round attack and $N_{2\mathrm{R}} \geq \frac{8}{\epsilon \cdot (1+\alpha+\alpha^3)}$ for the 2-round attack, where the $(.)^3$ comes from the three XOR losses that the leakage on b undergoes in its connection with the key bytes.

4 The AES case study

We now move to our first main contribution and show how to formalize a realistic case study based on an unprotected AES implementation. For this purpose, we consider the implementation of four S-boxes and MixColumns in Figure 5.

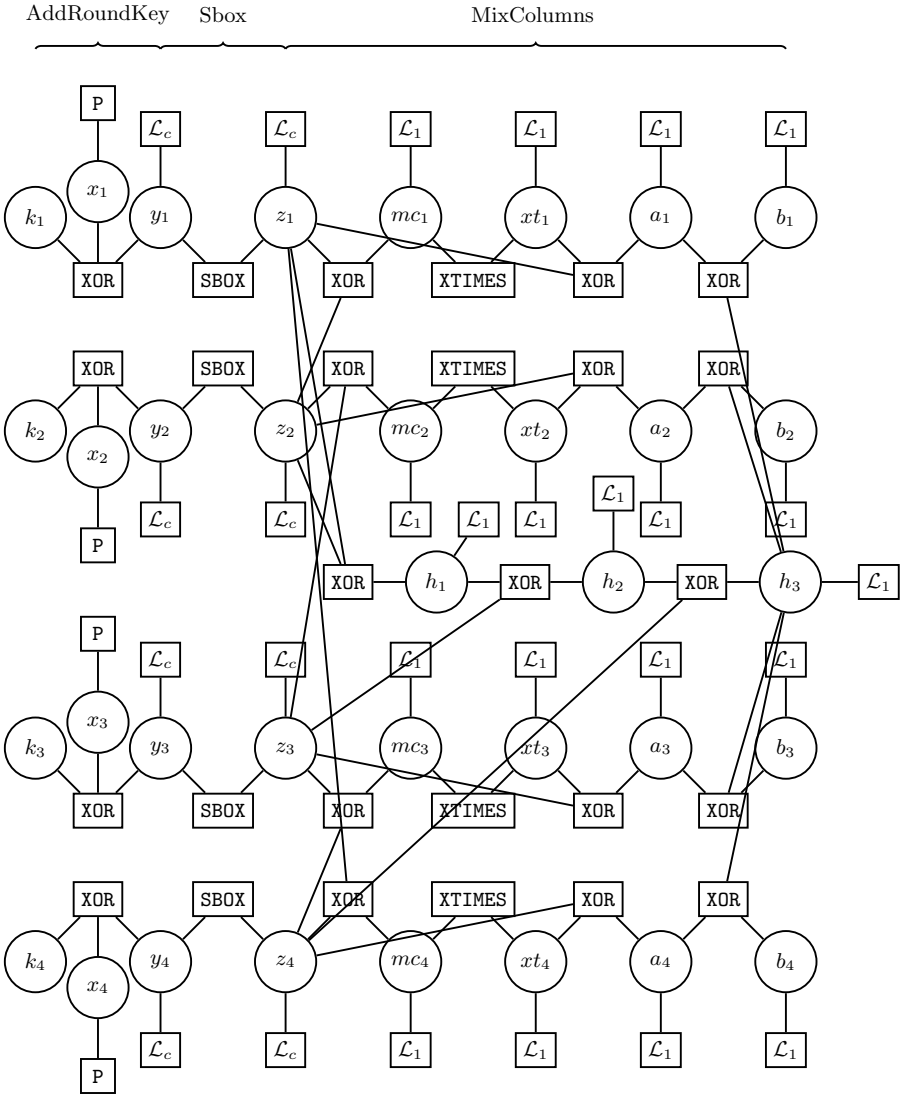

Fig. 5. Factor graph for the first round / first column of the AES (adapted from [12]).

4.1 Factor graph generation

In the original SASCA description of [12,29], the function nodes can be classified in three categories: leakage nodes, 1-input nodes (e.g., corresponding to the S-box and Xtimes operations), and nodes with two or more inputs (e.g., corresponding to the XOR operations). We have seen in Section 3 that if we locally combine the information leakages of variable nodes connected via 1-input function nodes before running the BP algorithm, then the decoding problem can be reduced to a simpler bipartite factor graph, called Tanner graph in coding theory.

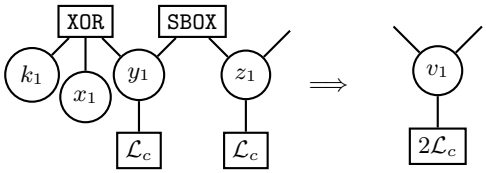

Fig. 6. Illustration of the merging trick.

Figure 6 illustrates this "merge trick" with a simple example. In the left part of the plot, showing a sub-graph of Figure 5, we have that $z_1 = \mathsf{S}(y_1)$ for the two variables z_1 and y_1 with distinct continuous leakages, and $y_1 = \mathsf{XOR}(x_1, k_1)$. Since the plaintext x_1 is known, we also have a bijection between k_1 and y_1. Therefore, we can merge these variables according to the two bijections, and create a new variable v_1 to represent them, with a single bivariate leakage function node corresponding to two continuous leakage channels.

This merge trick actually generalizes the "average trick" that combines the information on identical random variables obtained from many leakage traces in standard DPA. That is, say two leaking intermediate variables are connected by a function F: averaging is possible if F is the identity function. But in general, the merging trick straightforwardly applies to any bijection.[5] As will be discussed in the next section, this generalization turns out to be very useful in order to improve attacks against masked implementations of the AES S-box.

Figure 7 shows the simplified Tanner graph from the factor graph in Figure 5 corresponding to the first column of the first AES round. We then explain in Figure 8 the correspondence between the merged nodes in Figure 7 and the original variable nodes in Figure 5. For instance, the first row of the first column in Figure 8 means that the node v_1 in Figure 7 is the merge of three random variables in Figure 5: k_1, y_1 and z_1. The leakages are combined accordingly. As in the previous section, all the leakages are single-shot excepted the (continuous)

[5] It is in fact not even necessary that F is a bijection. For various F functions, it is possible to extract information on a random variable X from $\mathsf{F}(X)$.

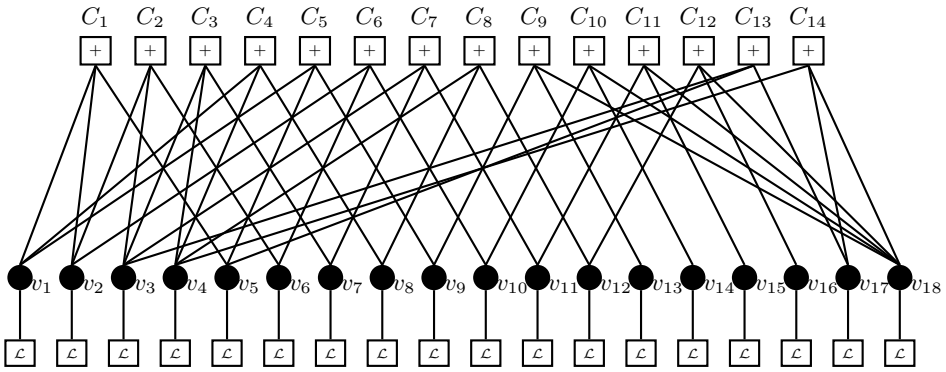

Fig. 7. Tanner graph associated with one column of the first AES round.

ones that correspond to variable nodes that can be targeted by a divide-and-conquer attack (i.e., the nodes v_n for $n \in \{1, 2, 3, 4\}$ that represent the key bytes), and the figure also includes the number of leakage channels.

- v_1:= y_1, z_1, k_1; leakage := $2\mathcal{L}_c$;
- v_2:= y_2, z_2, k_2; leakage := $2\mathcal{L}_c$;
- v_3:= y_3, z_3, k_3; leakage := $2\mathcal{L}_c$;
- v_4:= y_4, z_4, k_4; leakage := $2\mathcal{L}_c$;
- v_5:= xt_1, mc_1, h_1; leakage := $3\mathcal{L}_1$;
- v_6:= xt_2, mc_2; *leakage* := $2\mathcal{L}_1$;
- v_7:= xt_3, mc_3; leakage := $2\mathcal{L}_1$;
- v_8:= xt_4, mc_4; leakage := $2\mathcal{L}_1$;
- v_9:= a_1; leakage := $\mathcal{L}_1$;
- v_{10}:= a_2; leakage := $\mathcal{L}_1$;
- v_{11}:= a_3; leakage := $\mathcal{L}_1$;
- v_{12}:= a_4; leakage := $\mathcal{L}_1$;
- v_{13}:= b_1; leakage := $\mathcal{L}_1$;
- v_{14}:= b_2; leakage := $\mathcal{L}_1$;
- v_{15}:= b_3; leakage := $\mathcal{L}_1$;
- v_{16}:= b_4; leakage := $\mathcal{L}_1$;
- v_{17}:= h_2; leakage := $\mathcal{L}_1$;
- v_{18}:= h_3. leakage := $\mathcal{L}_1$;

Fig. 8. The correspondence between the nodes in Figure 5 and 7.

We can directly see that the factor graph used for the BP algorithm is greatly simplified. In contrast with the factor graph representation in Figure 5, which includes 35 variable nodes, the new one in Figure 7 consists of only 18 variable nodes. Moreover, if one has N traces, the number of variable nodes in the extended graph corresponding to multiple traces drops from $4 + 31N$ to $4 + 14N$, leading to reduced time and memory complexities for the decoding.

4.2 Local estimation rules

Based on the Tanner graph from the previous subsection, the BP algorithm works locally. That is, a (variable or function) node computes the output distribution on an edge according to the input probability distributions coming from

its neighbors. Thus, given a factor graph, the main ingredients to analyze the decoding performance are the local rules for the estimation of the output MI values on an edge from the input MI values on the other connected edges.

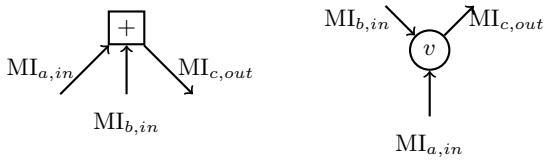

Fig. 9. Illustration of the local rules for estimating MI values.

These local rules are illustrated in Figure 9. The left part of the figure is for the function nodes and the right part for the variable nodes, assuming a degree-3 node in both cases. Based on this figure, we need to compute $\mathrm{MI}_{c,out}$ from two input probability distributions with MI values, $\mathrm{MI}_{a,in}$ and $\mathrm{MI}_{b,in}$, respectively. More generally, given the edges (with index set $\mathcal{I}$) connected to a node, we need to compute $\mathrm{MI}_{e,out}$ from $\{\mathrm{MI}_{e',in} | e' \in \mathcal{I}\}$, for every edge $e \in \mathcal{I}$.

Assuming that the probability distribution on each edge is independent, the rule at the variable node is simple, and given by:

$$\mathrm{MI}_{e,out} = \sum_{e' \in \mathcal{I} \setminus \{e\}} \mathrm{MI}_{e',in}, \tag{17}$$

As for the rule at the function node, we adopt an approximation frequently employed in coding theory [23] and also in the previous masking proof papers [7, 8]. That is, we approximate the probability distribution on each edge $e \in \mathcal{I}$ by a distribution from a q-ary erasure channel (which, as previously mentioned, corresponds to the random-probing model) with capacity equal to $1 - \mathrm{MI}_{e,in}$.[6] Thus the erasure probability is $\mathrm{MI}_{e,in}$, and we can compute $\mathrm{MI}_{e,out}$ as:

$$\mathrm{MI}_{e,out} = \prod_{e' \in \mathcal{I} \setminus \{e\}} \mathrm{MI}_{e',in}. \tag{18}$$

We call this approximation "piling-up lemma" due to their similar forms (despite the piling up lemma actually applies to binary random variables).

Eventually, and as discussed in Section 3.4, we can include a XOR loss factor α in Equation 18 in order to reflect information losses in the BP algorithm:

$$\mathrm{MI}_{e,out} = \prod_{e' \in \mathcal{I} \setminus \{e\}} \alpha \cdot \mathrm{MI}_{e',in}, \tag{19}$$

The latter allow more realistic (and less pessimistic) security estimations.

[6] The value $\mathrm{MI}_{e,in}$ is then computed using logarithm with base q.

Cautionary note. As already mentioned at the end of Section 3.4, Equation 17 can be inaccurate when the noise becomes too small, leading to highly correlated distributions and MI values quite dependent of the algebraic form of the leakages. The latter case is however of limited practical relevance, since such low-noise implementations generally correspond to insecure ones anyway.

4.3 The Local Random Probing Model (LRPM)

Based on the previous descriptions, we can now define the LRPM as a model in which an implementation is represented by a factor graph (as in Section 4.1), its leakages are captured by the RPM, and the exploitation of its leakages is quantified thanks to the local propagation rules of Section 4.2. As mentioned in Footnote 1, the latter is not exactly worst-case since it restricts the way the information is exploited. We call it "nearly worst-case" since such a restriction seems necessary in order for the attacks to remain computationally tractable [13]. We also note that due to the reduction in [7], we could equally call this model the "local noisy leakage model". Concretely, it reflects the actual security of an implementation captured with an information theoretic metric [8, 27].

4.4 Analysis and results

Based on the Tanner graph of Figure 7 and the local rules of Equations 17 and 18, we can derive formulas to estimate the performance of a SASCA targeting the first round of AES. For readability, we omit the full set of formulas for this case and present the results obtained in Figure 10, where the left plot corresponds to a leakage of $\epsilon = 0.01$ and the right plot corresponds to a leakage of $\epsilon = 0.001$. Similar results for two AES rounds are given in Appendix C, Figure 18.

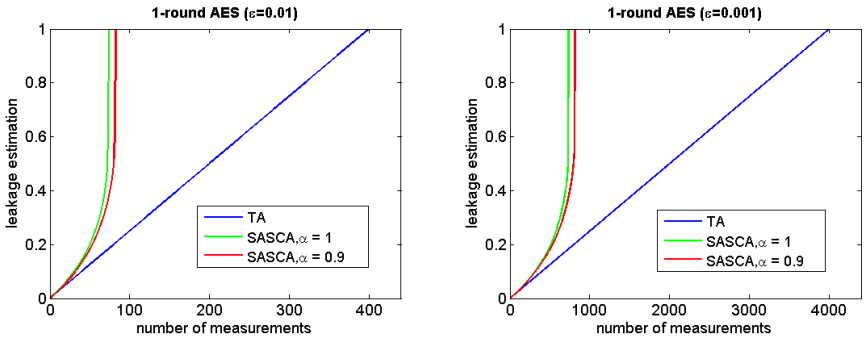

Fig. 10. Leakage bounds for one AES round (known plaintext attack).

We observe that these predictions nicely match the experimental results in [12, 29]. More precisely, the gain obtained thanks to the SASCA over the (bivariate) TA is independent of the noise (as witnessed by the similarity between the left and right parts of the figure); the approximate factor 5 that we

observe between the two attacks quite accurately corresponds to the one in these references; and the addition of a second round of leakages only reduces the attack complexity to a lower extent (i.e., by less than a factor 2 – see Figure 18 Appendix C).[7] Importantly, the fact that the the leakages of the external rounds dominate in known-plaintext/ciphertext attacks does not imply that the inner rounds cannot be exploited. As a typical example, in an unknown-plaintext/ciphertext attack, all rounds leak a similar amount of information (that roughly corresponds to the one of a 2-share masked implementation [19]). As illustrated in Figure 11, the addition of a second round of leakages is clearly beneficial in this case.[8] More technically, we also observe that the impact of the loss factor is quite limited in our experiments, due to the fact that several intermediate variables are re-used multiple times in the implementation of MixColumns. Yet, it increases when considering the inner round leakages (which have to go through more XOR operations) and in the unknown-plaintext scenario.

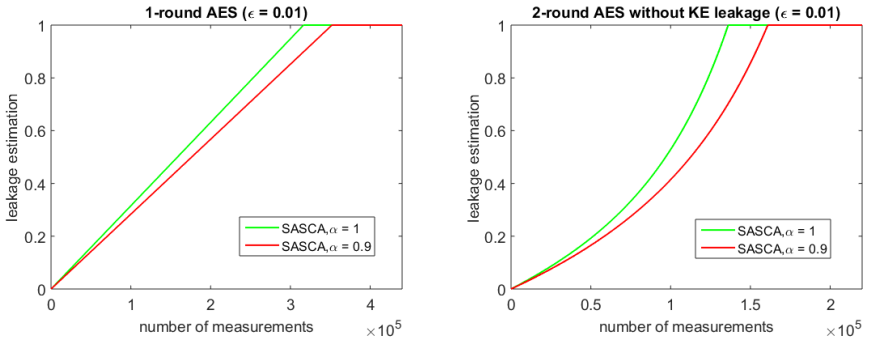

Fig. 11. Leakage bounds for one/two AES rounds (unknown plaintext attack).

Overall, the main interest of the LRPM is that our approximations are obtained instantaneously (given an estimation of ϵ), while they correspond to hours of computations when based on SASCA exploiting the BP algorithm. They are also easily reproducible and deterministic, which makes them a tool of choice for the systematic analysis and comparison of concrete implementations.

4.5 Connections to coding theory

We conclude this section by highlighting interesting connections between SASCA and coding theory. In particular, the Tanner graph derived from one column of the first AES round (in Figure 7) defines an LDPC code with a parity-check

[7] The 2-round attack in Figure 18 exploits key scheduling leakages. The version without key scheduling leakages does not bring any improvement at all.

[8] The 2-round attack in Figure 11 does not exploit key scheduling leakages. The addition of key scheduling leakages would remove the aforementioned "masking effect" and dominate the round leakages, for the 1-round and 2-round cases.

matrix $\mathbf{H}_{18\times 14}$ shown in Figure 12, whose row weight is a constant 3, and the column weight vector (i.e., the degree vector of variable nodes) equals:

$$(\ 3\ 3\ 4\ 4\ 3\ 2\ 2\ 2\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 2\ 5\ )$$

The sparsity of this parity-check matrix $\mathbf{H}_{18\times 14}$ explains why BP performs well.

$$\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

Fig. 12. Parity-check matrix of the SC-LDPC code corresponding to Figure 7.

We can further formalize this type of LDPC codes used in SASCA a Side-Channel Low-Density Parity-Check (SC-LDPC) codes, defined as follows:

Definition 4 (SC-LDPC codes). *A Side-Channel Low-Density Parity-Check (SC-LDPC) code $\mathcal{C}_{N,\mathcal{E}}$ is an LDPC code induced by the factor graph corresponding to a fixed number N of encryption procedures $\mathcal{E}$ of a certain cipher.*

When N increases, the dimension of the derived code is invariant, i.e., equivalent to the key size. Its length, however, increases and the additive channels associated to some positions (i.e., connected to so-called continuous channels) become less noisy. The latter explains why more traces lead to better attacks (and in particular SASCA) from a coding theory viewpoint, since both low-rate codes and low-noise channels generally imply better decoding performances. Note that the coding theory literature contains tools (e.g., EXIT charts) developed for designing codes with near-optimal decoding performances (called capacity-approaching codes). Since the goal of a cryptographic designer is to guarantee that high number of measurements N are needed for decoding, it is an interesting open problem to investigate whether the analogies in this section may lead to better solutions to counteract side-channel attacks (e.g., implementations of which the corresponding codes have provably poor decoding performances).

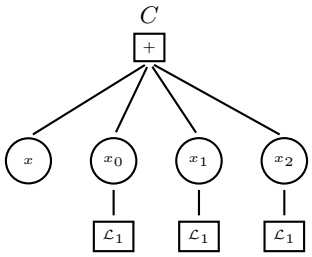

Fig. 13. Masked encoding with $d = 3$ shares.

5 Masked implementations

We next move to our second main contribution and extend our analysis and modeling of SASCA to the more challenging case of masked implementations, for which higher security levels can be expected and therefore shortcut approaches to simplify the security evaluations are increasingly needed. We show that our concrete use of the RPM can lead to new insights (e.g., unreported attack paths against a masked AES S-box due to a lack of refreshing of its linear parts) and serve as a tool to discuss the tradeoff between the amount of noise and randomness used in masked implementations. We first analyze small (encoding and multiplication) gadgets and follow with the analysis of an AES S-box.

5.1 Masked encoding

The Tanner graph of a masked encoding with three shares is given in Figure 13. In general, it corresponds to a secret value x is represented as an XOR of d shares $x = x_0 \oplus x_1 \oplus \ldots \oplus x_{d-1}$. The adversary can access d leakages corresponding to all the shares. For simplicity, we assume that each share's leakage has a similar mutual information (per bit) of λ, leading to a set of equations:

$$p_{x_i \to c} = \lambda \tag{20}$$

$$r_{c \to x} = p^d_{x_i \to c} \cdot \alpha^{(d-1)}, \tag{21}$$

$$\mathcal{B} = N \cdot r_{c \to x}, \tag{22}$$

where N is the number of measurements used in the attack. The resulting approximation is similar to the bounds given in masking proofs.[9] Note that in this simple context, the application of the BP algorithm is equivalent to the application of a multivariate TA if the noise covariance matrix is diagonal.

[9] Excepted that our modeling considers the leakages per bit rather than the mutual information, which is less conservative, yet typically sufficient for simple leakage functions such as the Hamming weight one we consider in our simulations.

5.2 Masked multiplication

Given some encoded values, masked multiplications usually apply some processing to the shares, based on secure addition and multiplication algorithms [14]. For illustration, we first consider a factor graph for a secure multiplication algorithm similar to the one investigated in [13] and depicted in Figure 14 for three shares. The adversary can observe the leakages of the d^2 partial products (denoted as $\mathcal{L}_{\mathrm{p}}$), together with the shares of the leakages (denoted as $\mathcal{L}_{\mathrm{s}}$).[10] We can consider two leakage assumptions for this purpose:

- *First assumption:* single 1-shot leakages ($\mathcal{L}_{\mathrm{s}} = \lambda$).
- *Second assumption:* multiple 1-shot leakages ($\mathcal{L}_{\mathrm{s}} = d \cdot \lambda$) due to the need to manipulate each share d times in the multiplication algorithm of [14].

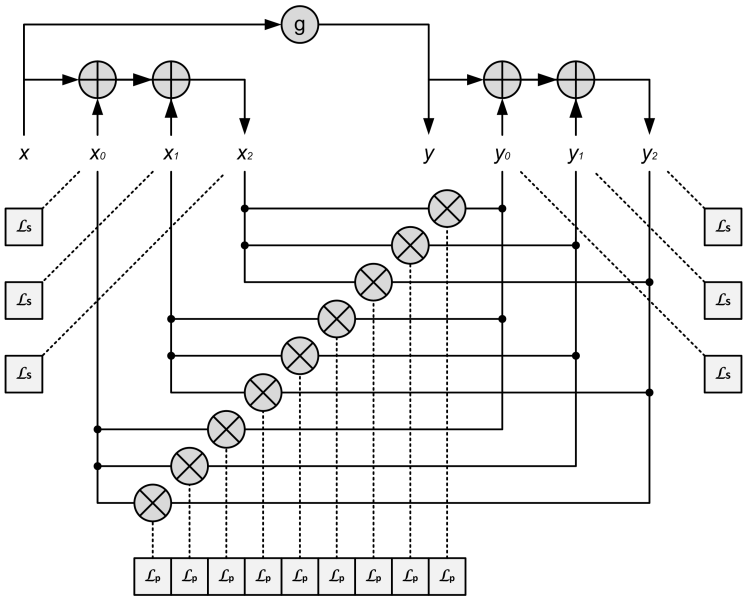

Fig. 14. Factor graph for a secure multiplication with three shares.

Since g is a bijection, we employ the merge trick to have the following equations, where the check nodes of the XOR operations used for the encoding of x and y are denoted s, and the ones of the multiplication functions are denoted p:

$$r^{(0)}_{s \to x_i} = 0 \tag{23}$$

$$r^{(0)}_{s \to x} = 0 \tag{24}$$

$$r^{(0)}_{p \to x_i} = 0 \tag{25}$$

[10] As discussed in [13], the latter are necessary to initialize the shares' leakage.

$$p^{(t)}_{x_i \to s} = \mathcal{L}_s + d \cdot r^{(t-1)}_{p \to x_i} \tag{26}$$

$$p^{(t)}_{x_i \to p} = \mathcal{L}_s + (d-1) \cdot r^{(t-1)}_{p \to x_i} + r^{(t-1)}_{s \to x_i} \tag{27}$$

$$p^{(t)}_{x \to s} = (2N-1) \cdot r^{(t-1)}_{s \to x} \tag{28}$$

$$r^{(t)}_{s \to x} = \left(p^{(t)}_{x_i \to s}\right)^d \cdot \alpha^{(d-1)} \tag{29}$$

$$r^{(t)}_{s \to x_i} = \left(p^{(t)}_{x_i \to s}\right)^{(d-1)} \cdot p^{(t)}_{x \to s} \cdot \alpha^{(d-1)} \tag{30}$$

$$r^{(t)}_{p \to x_i} = p^{(t)}_{x_i \to p} \cdot \mathcal{L}_p \cdot \beta \tag{31}$$

$$\mathcal{B}^{(t)} = 2N \cdot r^{(t)}_{s \to x} \tag{32}$$

Due to the symmetry, we only need to write equations for the x variables. Similar to the XOR loss factor α, we define a multiplicative loss factor β (we set both to one for simplicity – small variations do not affect our conclusions). We also mention the factor $(2N-1)$ in Equation 28 for which N edges come from the y shares and $(N-1)$ edges come from the x shares. Similarly in Equation 32, the factor 2 comes from the combination of the leakages on x and y thanks to g.

Maybe the only surprising element of these equations is the multiplicative local estimation rule of Equation 31. One could indeed expect a higher extrinsic information than predicted by this rule due to the well-known "zero problem" of multiplicative masking schemes [11]. However, a reasoning based on the random probing model shows that the situation is actually different here. Roughly, suppose that we have three random variables X, Y and Z over $\mathrm{GF}(q)$, where $Z = X \cdot Y$ and we have two independent leakages $\mathcal{L}(Z)$ and $\mathcal{L}(X)$ from two erasure channels with erasure probabilities e_1 and e_2 (i.e., capacity $\mathrm{MI}_1 = 1 - e_1$ and $\mathrm{MI}_2 = 1 - e_2$). The question is: how much information can we gain about Y? Since we know the value y of Y if and only if $\mathcal{L}(X) \in \mathrm{GF}(q) \setminus \{0\}$ and $\mathcal{L}(Z) \neq \perp$, the two observations $\mathcal{L}(Z)$ and $\mathcal{L}(Y)$ form a new erasure channel of X with erasure probability at least $1 - (1-e_1)(1-e_2) = 1 - \mathrm{MI}_1 \cdot \mathrm{MI}_2$. So the capacity of the new channel is at most $\mathrm{MI}_1 \cdot \mathrm{MI}_2$, justifying Equation 31.

Based on these equations, we first show in Figure 15 the reduction of the data complexity when exploiting the leakages of the partial products $\mathcal{L}_\mathrm{p}$, expressed as the ratio with the data complexity of the attack exploiting only the shares' leakages $\mathcal{L}_\mathrm{s}$. It clearly illustrates that as the noise level increases (i.e., the leakage per share ϵ decreases), the impact of these partial product vanishes, independent of the leakage assumption.[11] By contrast, for too low noise levels these additional leakages become significant, especially for large d values. So the latter confirms the requirement that the level of noise needed to "hide" the shares' leakages increases linearly in d. A similar view can be extracted from Figure 16 where the information leakages of different masked multiplications are given, for two noise levels. The latter additionally shows the positive impact of increasing d when the noise level is large enough, and its detrimental effect when the noise is too low (due to the higher complexity ratio illustrated in Figure 15).

[11] Yet, it remains that the actual data complexities depend on the leakage assumption and are a factor d^d lower in the case of the second leakage assumption [13].

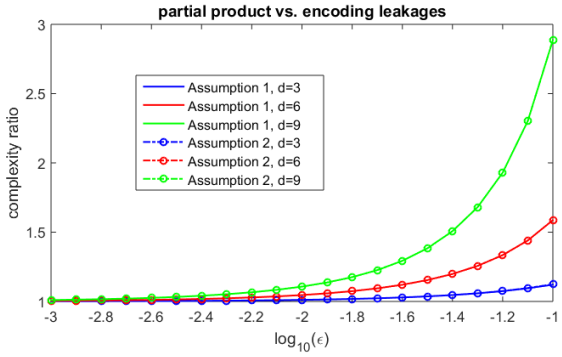

Fig. 15. Information gain of masked multiplications.

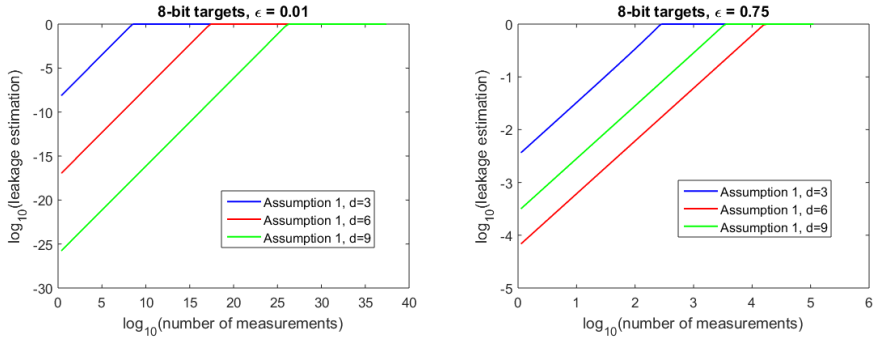

Fig. 16. Leakage bounds for the factor graph of Figure 14 (leakage assumption 1).

Overall, these experiments illustrate that the LRPM allows us to revisit the key intuitions of masking security proofs [7,8,13,22], but in a much more flexible manner: we are indeed able to immediately gauge the impact of the repeated manipulation of a share, or a variation of information leakage for a complete factor graph accurately describing the target implementation.

A note on the BP algorithm for masked implementations. The experimental investigations in Section 3.4 suggested that in the case of an unprotected implementation, running the BP algorithm on a factor graph connecting all the manipulated plaintexts (i.e., graph extension) leads to better results than running it on several smaller factor graphs that are then re-combined (i.e., graph combination). Interestingly, we can show that the latter fact does not hold for higher-order masked implementation with more than two shares.

The key observation is that in a masked implementation, we can only extend the graph thanks to the unshared values which can potentially accumulate leakages continuously (all other intermediate variables are ephemeral). These unshared variables are then injected in each sub-graph by passing through the

sharing node (e.g., a d-input XOR in the case of Boolean masking). This means that the information that the unshared values are able to send to each share essentially corresponds to the information of a $(d-1)$-share encoding. As a result, graph extension is effective for unprotected implementation and first-order masked implementation, but rapidly becomes uselss for higher orders.

We confirmed this theoretical explanation by running again the experimental SASCA against a (3-share) masked multiplication carried out in [13], Section 5.4. The latter was performed with graph combination. We evaluated the impact of graph extension and did not observe any improvement, as predicted.

This limited positive impact of graph extension is a plausible reason for the slightly worse performances of the BP algorithm in a masked context (and the fact the gain of a SASCA over a TA decreases with the noise level in [13], contrary to what is observed for unprotected implementations). In an unprotected implementation, the accumulation of continuous leakages presumably allows optimal information extraction by the BP algorithm, despite the presence of cycles in the factor graph that do not guarantee convergence. In a masked implementation, the lack of continuous leakages make the presence of cycles more detrimental.

5.3 Masked S-box implementations

To conclude this paper, we finally illustrate that our modeling does not only allow confirming existing analyzes and connecting the RPM with coding theory tools and SASCA, but can also put forward attacks that were not directly captured by theoretical analysis so far. For this purpose, we consider the case study of an AES S-box as represented in Figure 17 (originally proposed in [24] but here tweaked with the refreshing gadgets of [14] to avoid the attacks put forward in [6]). Considering asymptotic analysis for readability, our main observation is that the merge trick discussed in Section 4.1 applied to the case of a masked implementation can bring a powerful generalization of the horizontal attacks in [2]. For this purpose, first consider an attack targeting the shares of y_1. Given a leakage λ on each share (and ignoring the loss factors for simplicity), we obtain a leakage bound of λ^d on y_1. But by merging the node of y_1 with the one of y_2 (which is feasible since these linear functions are operated share by share), one reduces this bound to $(2 \cdot \lambda)^d$. Then additionally considering the fact that the shares of y_1 and y_2 will be loaded d times when being multiplied, it is further reduced to $(2{\cdot}\lambda{+}2d{\cdot}\lambda)^d$, meaning a security reduction by a factor $(2{+}2d)^d$, which grows exponentially in d. We leave the application of this attack to concrete implementations as an interesting direction for further investigations.

The latter observation naturally becomes even more critical if all the S-box (and possibly MixColumns) operations are combined in a similar manner. In this respect, the important message of this final analysis is that besides being useful for composability issues [1], the addition of refreshing gadgets may also be useful to prevent such powerful SASCA. For example, adding a simple refreshing (e.g., a sharing of zero) after each (even linear) operation in Figure 17 would prevent the combination of certain leakages. That is, the merge of y_1 and y_2 would

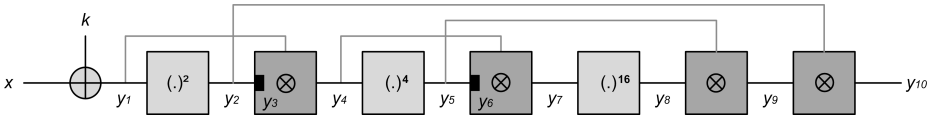

Fig. 17. AES S-box from [24] (with refreshing from [14]).

remain unavoidable (leading to a security reduction of 2^d), just as the combination of the d manipulations of each share during the multiplication (leading to a security reduction of d^d), but their further combination would be prevented by the refreshing. Improving the security of the multiplication could also be achieved, by using the algorithm proposed in [2] (which minimizes the number of times each share is manipulated). So our results recall that the masking problem can be stated as a global optimization trading the physical randomness (aka noise) and the pseudo-randomness used for refreshing the shares. The design of efficient gadgets (and complete implementations) that optimize the tradeoff between these two ingredients is one of the main research challenges in masking and leakage-resilience. In this respect, and once again, we observe that a systematic analysis of multiplication gadgets (or complete implementations) with heuristic tools such as SASCA or the ones in [2] would hardly be practical. By contrast, the LRPM provides a principled path for this purpose, which we hope will enable improved analyzes and optimizations in the future.

Acknowledgments. François-Xavier Standaert is a senior associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the ERC project 724725 (acronym SWORD), the EU project REASSURE and the CHIST-ERA project SECODE.

References

1. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 116–129. ACM, 2016.
2. Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.
3. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-*

12, 2016, Proceedings, Part II, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, 2016.

4. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Private multiplication over finite fields. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 397–426. Springer, 2017.
5. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Jr. et al. [15], pages 13–28.
6. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.
7. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Nguyen and Oswald [20], pages 423–440.
8. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.
9. François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Nguyen and Oswald [20], pages 459–476.
10. Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Information Theory*, 8(1):21–28, 1962.
11. Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In Jr. et al. [15], pages 198–212.
12. Vincent Grosso and François-Xavier Standaert. Asca, SASCA and DPA with enumeration: Which one beats the other and when? In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 291–312. Springer, 2015.
13. Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 385–412. Springer, 2018.
14. Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
15. Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Red-*

wood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*. Springer, 2003.
16. David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.
17. Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.
18. Luke Mather, Elisabeth Oswald, and Carolyn Whitnall. Multi-target DPA attacks: Pushing DPA beyond the limits of a desktop computer. In Sarkar and Iwata [25], pages 243–261.
19. Marcel Medwed, François-Xavier Standaert, Ventzislav Nikov, and Martin Feldhofer. Unknown-input attacks in the parallel setting: Improving the security of the CHES 2012 leakage-resilient PRF. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 602–623, 2016.
20. Phong Q. Nguyen and Elisabeth Oswald, editors. *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*. Springer, 2014.
21. Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In David L. Waltz, editor, *Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, August 18-20, 1982.*, pages 133–136. AAAI Press, 1982.
22. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.
23. Thomas J. Richardson and Rüdiger L. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.
24. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
25. Palash Sarkar and Tetsu Iwata, editors. *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*. Springer, 2014.
26. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.
27. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances*

in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
28. Robert Michael Tanner. A recursive approach to low complexity codes. *IEEE Trans. Information Theory*, 27(5):533–547, 1981.
29. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Sarkar and Iwata [25], pages 282–296.

A The Belief Propagation (BP) algorithm

Our description is inspired by the description of [16, Chapter 26]. For this purpose, we denote by α_i the i^{th} intermediate value and by f_i the i^{th} function node. The nodes will be connected by edges that carry two types of messages. The first ones go from a variable node to a function node, and are denoted as $p_{v_n \to f_m}$. The second ones go from a function node to a variable node, and are denoted as $r_{f_n \to v_m}$. In both cases, n is the index of the sending node and m the index of the recipient node. The messages carried correspond to the scores for the different values of the variable nodes. At the beginning of the algorithm execution, the messages from variable nodes to function nodes are initialized with no information on the variable. That is, for all n, m and for all α_n we have:

$$p_{v_n \to f_m}(\alpha_n) = 1.$$

The scores are then updated according to two rules (one per type of messages):

$$r_{f_m \to v_n}(\alpha_n) = \sum_{\alpha_{n'}, n' \neq n} \big(f_m(\alpha_{n'}, x_n) \prod_{n'} p_{v_{n'} \to f_m}(\alpha_{n'}) \big). \tag{33}$$

$$p_{v_n \to f_m}(\alpha_n) = \prod_{m' \neq m} r_{f_{m'} \to v_n}(\alpha_n). \tag{34}$$

In Equation 34, the variable node v_n sends the product of the messages about α_n received from the others function nodes ($m' \neq m$) to the function node f_m, for each value of α_n. And in Equation 33, the function node f_m sends a sum over all the possible input values of f_m of the value of f_m evaluated on the vector of $(\alpha_{n'}, n' \neq n)$'s, multiplied by the product of the messages received by f_m for the considered values of $\alpha_{n'}$. The BP algorithm essentially works by iteratively applying these rules on all nodes. If the factor graph is a tree (i.e., if it has no loop), a convergence should occur after a number of iterations at most equal to the diameter of the graph. In case the graph includes loops (e.g., as in the case of a SASCA against an AES implementation), convergence is not guaranteed, but usually occurs after a number of iterations slightly larger than the graph diameter. The main parameters influencing the time and memory complexity of the BP algorithm are the number of possible values for each variable (e.g., 2^8 for 8-bit target intermediate computations) and the number of edges in the factor graph. The time complexity additionally depends on the number of inputs of the function nodes representing the (e.g., block cipher) operations, since the first rule sums over all the input combinations of these operations.

B Matlab pseudo-code of a toy example

$n = 8$; -- n-bit targets
$e = 0.1$; -- MI per target operation
$Nd = 2$; -- 1 for univariate, 2 for bivariate
$Nmax = n/(Nd * e)$; -- max. data complexity for the plots)
$Nmax = ceil(Nmax * (1.1))$;

– **Divide-and-conquer approach:**
$\mathrm{MI}_{\mathrm{dc}}(1) = 0$;
for $i = 1 : Nmax$
  $\mathrm{MI}_{\mathrm{dc}}(i+1) = \mathrm{MI}_{\mathrm{dc}}(i) + (Nd * e)$;
end

– **Analytical approach:**
$Nit = 10$; -- # of iterations (2 x graph diam.)
$\mathrm{MI}_{\mathrm{a}}(1) = 0$;
for $i = 1 : Nmax$
  $Rck(1) = 0$;
  for $j = 1 : Nit$
    $Pkc(j+1) = i * (Nd * e/n) + (i-1) * Rck(j)$;
    $Rck(j+1) = Pkc(j+1) * (Nd * e/n)$;
    $\mathcal{B}(j+1) = i * (Nd * e/n) + i * Rck(j+1)$;
  end
  $\mathrm{MI}_{\mathrm{a}}(i+1) = max(\mathcal{B})$;
end

C Additional figures

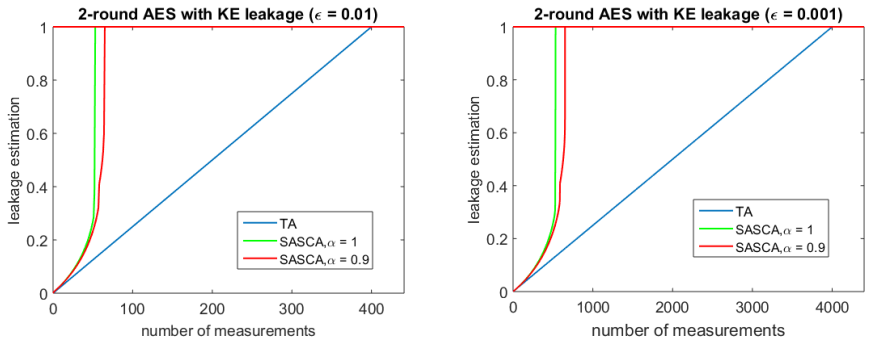

Fig. 18. Leakage bounds for two AES rounds (known plaintext attack).