HAL
open science

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

Qian Guo, Vincent Grosso, François-Xavier Standaert, Olivier Bronchain

To cite this version:

Qian Guo, Vincent Grosso, François-Xavier Standaert, Olivier Bronchain. Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020 (4), 10.13154/tches.v2020.i4.209-238 . hal-02901366

HAL Id: hal-02901366

https://hal.science/hal-02901366v1

Submitted on 17 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Qian Guo[1,2,3], Vincent Grosso[4],
François-Xavier Standaert[3], Olivier Bronchain[3]

[1] Department of Electrical and Information Technology, Lund University, Sweden
[2] Department of Informatics, University of Bergen, Norway
[3] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
[4] CNRS/Laboratoire Hubert Curien, Université de Lyon, France

e-mails: qian.guo@eit.lth.se;grosso.vincent@gmail.com;fstandae@uclouvain.be

Abstract. One important open question in 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 attacks exploiting a divide-and-conquer strategy, the answer is negative: 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 decoded thanks to belief propagation. Previous works have shown the low data complexities that SASCA can reach in practice.

In this paper, 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 bounding the security of concrete implementations against SASCA in a fast and intuitive manner. We use it in order 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 tradeoffs between their amount of randomness and physical noise level. Third, we show that it can considerably speed up the evaluation of other countermeasures such as shuffling.

Keywords: Side-Channel Analysis · Worst-Case Security Evaluations · Horizontal (aka Multi-Target) Attacks · Belief Propagation · Masking · Shuffling

1 Introduction

A recent line of works started the investigation of masking security proofs as an ingredient of concrete side-channel security evaluations [PR13, DDF14, DFS19, GS18]. These papers 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 [MOS11]. Yet, the security bounds (e.g., in [DDF14, DFS19]) suggest that the adversary's success probability grows with the circuit size. It 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 or not.

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 it by proposing a model for the Soft Analytical Side-Channel Attacks (SASCA) introduced in [VGS14] and further developed in [GS15, GS18, GRO18]. These 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. Yet, the systematic exploitation of SASCA for physical security analyzes is limited by their heuristic nature and high time complexity. Our model – that we denote as the Local Random Probing Model (LRPM) – specializes the Random Probing Model used in masking proofs to local decoding rules similar to those used in SASCA. It enables interesting connections with the erasure channel used in coding theory, leading to the following benefits.

First, the LRPM allows speeding up the evaluation of SASCA (which is computationally intensive). For example, we show that we can bound the results of a SASCA against an AES implementation from [VGS14, GS15] 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 rounds in known-plaintext/ciphertext attacks, inner rounds' leakages cannot be neglected in other cases (e.g., in unknown-plaintext/ciphertext attacks). We insist that these computational gains become even more relevant when considering high-end devices. For example, propagating information through a XOR operation on a b-bit bus requires $\mathcal{O}(b \cdot 2^b)$ operations in SASCA, but this propagation has constant cost in the LRPM. So while our model already leads to considerable speed ups in the simple case of 8-bit devices investigated in previous works (due to the code size of the target implementations), these gains significantly increase if targeting a 32-bit or larger architecture. In this respect, we note that it is always possible to trade time for data, by targeting only a part of the bus and considering that the untargeted bits of the architecture produce "algorithmic noise". Yet, this would understate the security level since performing up to 2^{50} operations is achievable for determined adversaries [VGRS12, MOW14]. As a result, a tool that allows evaluating the worst-case data complexity of a SASCA is handy for security analysts.

Second, the LRPM allows revisiting masking security proofs and the evaluation of actual implementations in a flexible manner. More precisely, the RPM was so far mostly used as a technical ingredient in order to connect (abstract) probing security and (concrete) noisy leakage security [ISW03, PR13, DDF14]. 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. By nearly worst-case, we mean that our following tools can be used to bound the complexity of SASCA by estimating the information leakage that can be obtained when decoding the factor graph of an implementation using the Belief Propagation (BP) algorithm. A worst-case attack would apply Bayes on the full (multivariate) leakage distribution of the implementation, which rapidly leads to unrealistic time complexities as the size of protected implementations increase [GS18]. The latter is particularly interesting for two main reasons. On the one hand, it highlights 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 merging of some nodes in the factor graph that are connected by linear operations. On the other hand, recent works have optimized masking schemes in order to minimize their randomness requirements [BBP+16, BBP+17]. But ultimately, side-channel security is a tradeoff between physical noise and mathematical randomness. The current state-of-the-art typically optimizes this tradeoff based on qualitative arguments (i.e., trying to minimize the manipulation of the shares) or thanks to time-consuming heuristics such as SASCA [BCPZ16, GS18], which can hardly be used to analyze the noise vs. randomness tradeoff in a systematic manner. The LRPM is appealing for this purpose: in a

recent work, it has been used as a tool to guide the design of new multiplication algorithms optimized globally and jointly based on these ingredients (i.e., noise and randomness), for example enabling the identification of efficient gadgets with constant noise rate [CS19].

Finally, the LRPM can also be used to speed up the evaluation of other practically-relevant countermeasures such as shuffling [HOM06], for which the worst-case analysis implies computing expensive high-dimensional integrals [VMKS12]. We propose a simple modeling that captures the intuitions of previous analyzes at minimum evaluation cost.

Paper structure. After providing the necessary background in Section 2, our results are structured in three parts. We start by introducing the LRPM and 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 use the LRPM in order 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 protected (masked or shuffled) 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 [CRR02], 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).[1] 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 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 [SMY09]:

$$\mathrm{MI}(Y; L) = \mathrm{H}[Y] + \sum_{y \in \mathcal{Y}} \Pr[y] \sum_{l \in \mathcal{L}} \mathsf{f}[l|y]. \log_2 \Pr[y|l], \tag{1}$$

[1] Alternatively, leakage models can also be built by exploiting linear regression [SLP05].

where $\mathrm{H}[\cdot]$ is the entropy function. As discussed in [DFS19, dCGRP19] such a metric can be used in order to bound the success rate of a divide-and-conquer side-channel attack. Considering the typical context 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 in order to perform a successful key recovery can then be bounded as follows:

$$N \geq \frac{\mathrm{H}[K]}{\mathrm{MI}(K; X, L)} = \frac{\mathrm{H}[K]}{\mathrm{MI}(Y; L)}. \tag{2}$$

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 the discussion about leakage certification in [DSV14, BHM+19]. Our investigations are orthogonal to this modeling issue and just require to be fed with the best possible approximation of the MI metric, as for the 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 implies that only the leakage of the first/last rounds of a block cipher implementation can be exploited in this way [MOW14]. Second, estimating the optimal (mixture) model for a masked implementation becomes prohibitively expensive as the number of shares increases [GS18]. SASCA were first introduced as a solution in order to mitigate the first issue [VGS14, GS15], and were recently considered as an efficient heuristic in order to deal with the second one [BCPZ16, GS18]. Roughly, they work by describing the target implementation and its leakages in a way similar to a Low-Density Parity Check code (LDPC) [Gal62], which the adversary then tries to decode using posterior probabilities obtained thanks to a TA on all the intermediate values of the leaking implementation. More precisely, they work in three main 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 [Mac03].

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 [GS15] (but for more target intermediate variables).

3. *Decoding.* Similar to LDPC codes, the factor graph is decoded using a BP algorithm [Pea82], which iterates the local propagation of the information about the variable nodes of the target implementation. The BP algorithm has guaranteed convergence when the factor graph is a tree and only converges heuristically if it contains cycles.

2.3 The Belief Propagation (BP) algorithm

Our description is inspired by the description of [Mac03, Chapter 26]. For this purpose, we denote by α_i the ith intermediate value and by f_i the ith 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 $q_{v_n \to f_m}$. 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:

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

The scores are then updated for all α_i's according to two rules (one per message type):

$$r_{f_m \to v_n}(\alpha_n) = \sum_{\boldsymbol{\alpha}_m \setminus \alpha_n} \big(f_m(\boldsymbol{\alpha}_m, \alpha_n) \prod_{n' \in \mathcal{N}(f_m) \setminus n} p_{v_{n'} \to f_m}(\alpha_{n'}) \big). \tag{3}$$

$$p_{v_n \to f_m}(\alpha_n) = \prod_{m' \in \mathcal{M}(v_n) \setminus m} r_{f_{m'} \to v_n}(\alpha_n). \tag{4}$$

In Equation 4, $\mathcal{M}(v_n)$ denotes all the neighbors of the variable node v_n. The variable node sends to f_m the product of the messages about α_n received from all its neighbors excluding f_m. In Equation 3, the function node f_m sends to v_n a sum over all the possible values (denoted as $\boldsymbol{\alpha}_m$) of its neighbors (denoted as $\mathcal{N}(v_n)$), excepted for v_n which is fixed to α_n. Each element of the sum is a product over the messages received from all its neighbors (once again excluding v_n) about their value in $\boldsymbol{\alpha}_m$. 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.

2.4 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. Its (information) *rate*, denoted as R, is $\frac{\kappa}{\eta}$. The redundancy of the code is worth $\eta - \kappa$. We 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 [Gal62])**.** We define a Low-Density Parity-Check (LDPC) code as 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 has been shown to be close to the channel capacity).

2.4.1 Tanner Graphs.

One core feature of LDPC codes is their efficient decoding thanks to the BP algorithm on the *Tanner graph* [Tan81, RU08] representation of the code. A Tanner graph is a bipartite graph representation of the code, where each edge corresponds to a non-zero element in the parity-check matrix $\mathbf{H}$. Sparse $\mathbf{H}$'s are expected to lead to sparse Tanner graphs. In the coding literature, a function node in a Tanner graph is usually called a check node.

2.4.2 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 can be characterized by an erasure probability, or by the mutual information between the bit and the receiver's signal. For a q-ary channel, the leaked information can be bounded by $\log_2(q)$ bits, and this bound can be normalized to 1. After the normalization, the capacity for a q-ary erasure channel with erasure probability p is exactly $1 - p$. As a result, the RPM in [DDF14] can be viewed as consisting of many independent erasure channels with a bound on the erasure probability. Concretely, each channel corresponds to some leakage obtained on an intermediate value computed by a leaking device.

3 A toy example of unprotected implementation

In this section, we introduce our modeling and illustrate it 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 define the LRPM and evaluate the security of our toy example 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 with four plaintext bytes x_1, x_2, x_3 and x_4, four 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. In this setting, a divide-and-conquer adversary targets the four key bytes k_1 to k_4 independently and therefore exploits 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 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 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 capture situations where the information leakage 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 [GS18]). 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.

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:

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

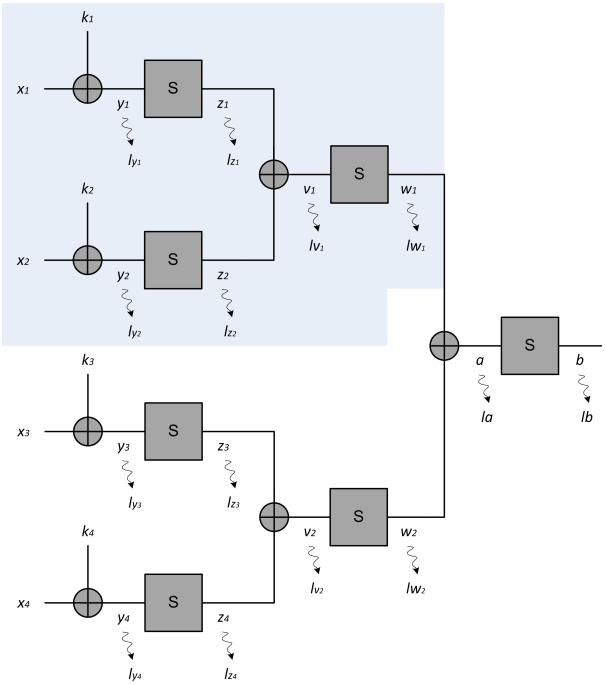


Figure 1: Toy unprotected implementation.

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 2 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. Moreover, considering the normalized mutual information leakage denoted by λ (e.g., here $\lambda = \frac{\epsilon}{8}$), we could re-write the bound as $N_{\mathrm{dc}} \geq \left\lfloor \frac{1}{2\lambda} \right\rfloor$, where the unit for λ becomes a byte.

3.3 A SASCA evaluation with the LRPM

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 (1), its side-channel leakages with the RPM (2) and their exploitation with (local) information propagation rules that can be viewed as a variation of the piling up lemma (3). The LRPM then simply corresponds to the model combing these three ingredients.

Factor graph. The factor graph to use in the LRPM is built in the same was as the factor graph of a SASCA. The factor graph corresponding to the toy example of Figure 1 is illustrated in Figure 2 (for the case where $N = 3$). It 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

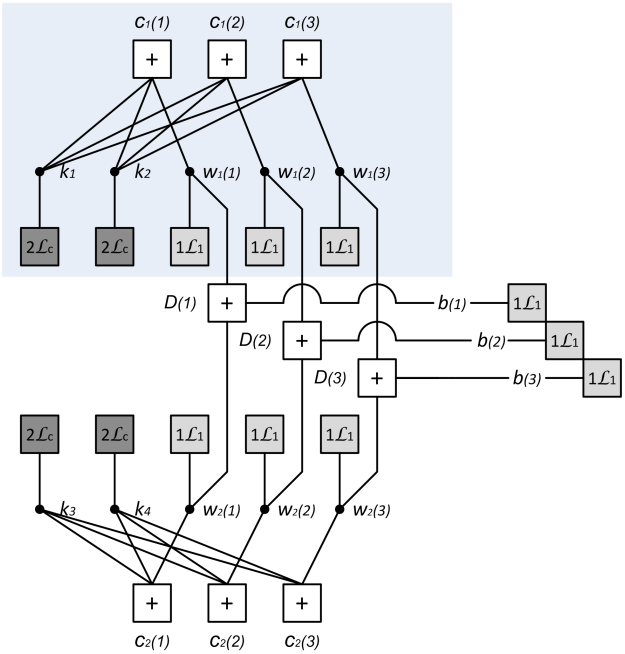


Figure 2: Example of factor graph corresponding to Figure 1 ($N = 3$).

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 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 $1\mathcal{L}_1$ 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. 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", the second ones can be viewed as a "code expansion".[2]

Side-channel leakage modeling. The next LRPM step is to compute the information leakage on each node. For the toy 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 therefore $\lambda = \frac{N\epsilon}{4}$ 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.

[2] 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 identity 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 (since it is non-linear).

Local information propagation rules. 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. In the LRPM, these computations on probability distributions are replaced by estimations of the output MI values on an edge from the input MI values on the other connected edges. We use the normalized mutual information in order to leverage the analogy of Section 2.4.2.

These local rules are illustrated in Figure 3: the left part is for the variable nodes and the right part for the function nodes, assuming a degree-3 node in both cases. Based

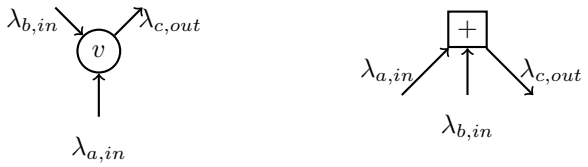


Figure 3: Illustration of the local rules for estimating the normalized MI values.

on this figure, we need to compute the normalized mutual information leakage bound $\lambda_{c,out}$ from two input probability distributions with the normalized MI bounds, $\lambda_{a,in}$ and $\lambda_{b,in}$. More generally, given the edges (with index set $\mathcal{I}$) connected to a node, we need to compute $\lambda_{e,out}$ from $\{\lambda_{e',in} | e' \in \mathcal{I}\}$, for every edge $e \in \mathcal{I}$. Assuming that the probability distribution on each edge is independent, the rule at the variable node is given by:

$$\lambda_{e,out} = \sum_{e' \in \mathcal{I} \setminus \{e\}} \lambda_{e',in}. \tag{5}$$

This rule is similar to the rule of measuring information from independent traces in the divide-and-conquer approach. As for the rule at the function node, we adopt an approximation frequently employed in coding theory [RU08] which is similar to the bounds in the masking security proof papers [DDF14, DFS19]. 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 erasure probability at least $1 - \lambda_{e,in}$. Assuming the independence of the distributions on the input edges, we can observe an erasure channel at the output edge, with erasure probability at least $(1 - \prod_{e' \in \mathcal{I} \setminus \{e\}} \lambda_{e',in})$. The capacity of the channel is at most $\prod_{e' \in \mathcal{I} \setminus \{e\}} \lambda_{e',in}$, and we can therefore derive a normalized MI leakage bound $\lambda_{e,out}$ as:

$$\lambda_{e,out} = \prod_{e' \in \mathcal{I} \setminus \{e\}} \lambda_{e',in}. \tag{6}$$

Illustrations. Based on the previous ingredients, we can now 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, 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 2.3). 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

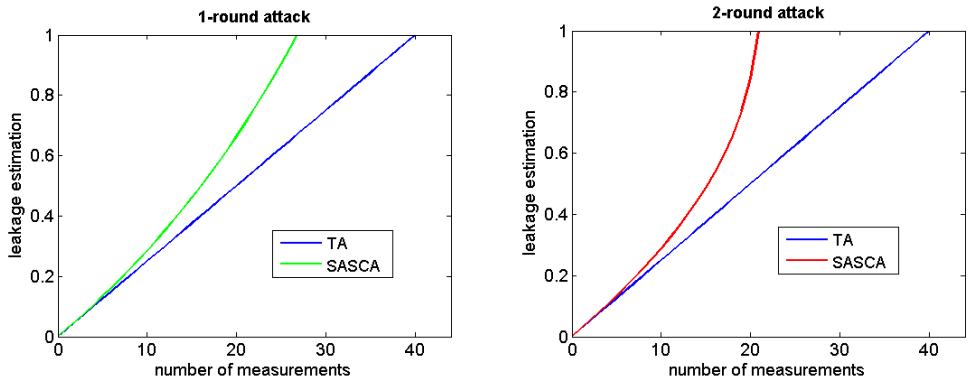


Figure 4: Leakage bounds for the factor graph of Figure 2.

then derive the following set of equations describing the bound:

$$r^{(0)}_{c_m \to k_n} = 0, \tag{7}$$

$$p^{(t)}_{k_n \to c_m} = \frac{N\epsilon}{4} + (N-1) \cdot r^{(t-1)}_{c_m \to k_n}, \tag{8}$$

$$r^{(t)}_{c_m \to k_n} = p^{(t)}_{k_n \to c_m} \cdot \frac{\epsilon}{8}, \tag{9}$$

$$\mathcal{B}^{(t)} = \frac{N\epsilon}{4} + N \cdot r^{(t)}_{c_m \to k_n}. \tag{10}$$

Note that the goal of SASCA is to recover key bytes, so the information bounds on other intermediate variable nodes are excluded in these equations. A pseudo-code for this example is given in Appendix A. 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 4 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 $27 \approx \frac{8}{0.3}$). In this respect, a natural question is whether the same intuition holds when digging further into a cipher's internal operations? We answer it by analyzing the 2-round adversary described by the equations:

$$r^{(0)}_{c_m \to k_n} = 0, \tag{11}$$

$$r^{(0)}_{c_m \to w_n} = 0, \tag{12}$$

$$r^{(0)}_{d_m \to w_n} = 0, \tag{13}$$

$$p^{(t)}_{k_n \to c_m} = \frac{N\epsilon}{4} + (N-1) \cdot r^{(t-1)}_{c_m \to k_n}, \tag{14}$$

$$p^{(t)}_{w_n \to c_m} = \frac{\epsilon}{8} + r^{(t-1)}_{d_m \to w_n}, \tag{15}$$

$$p^{(t)}_{w_n \to d_m} = \frac{\epsilon}{8} + r^{(t-1)}_{c_m \to w_n}, \tag{16}$$

$$r^{(t)}_{c_m \to k_n} = p^{(t)}_{k_n \to c_m} \cdot p^{(t)}_{w_n \to c_m}, \tag{17}$$

$$r^{(t)}_{c_m \to w_n} = p^{(t)}_{k_n \to c_m} \cdot p^{(t)}_{k_n \to c_m}, \tag{18}$$

$$r^{(t)}_{d_m \to w_n} = p^{(t)}_{w_n \to d_m} \cdot \frac{\epsilon}{8}, \tag{19}$$

$$\mathcal{B}^{(t)} = \frac{N\epsilon}{4} + N \cdot r^{(t)}_{c_m \to k_n}. \tag{20}$$

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 4 that the bound becomes larger than one for $N_{\mathrm{a}} \geq 21$. So here as well, the addition of the leakages of w_1 and v in the bounds is very close to the addition of two continuous channels on the key bytes (since $20 = \frac{8}{0.4}$). The latter toy experiments therefore provide a preliminary confirmation of the security proofs in [DDF14, DFS19] from a coding theory viewpoint, and in particular of the fact that the security of an implementation in the LRPM may decrease linearly with its circuit size Γ (reflected by the number of leaking operations), independent of whether these operations are exploitable via divide-and-conquer attacks or not.

3.4 Discussion & definition

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, we briefly discuss the worst-case nature of these predictions. There are two important observations in this respect. First, SASCA applied to complex factor graphs with cycles, such as the ones of a block cipher implementation, may suffer for convergence issues. The LRPM optimistically prevents these issues since it assumes that all the information is accumulated (as described by the local information propagation rules). Second, Equation 6 is an approximation employed in coding theory and is similar to the bounds used in masking security proofs. Yet, the notion of tightness considered in investigations such as [DFS19, GS18] is up to constant factors. A bound with constant factors becomes:

$$\lambda_{e,out} = \alpha \cdot \prod_{e' \in \mathcal{I} \setminus \{e\}} \lambda_{e',in}. \tag{21}$$

The α value proposed in [DFS19] is $\frac{1}{\log(2)}$ (see footnote 3). As will be seen next, this choice is conservative and approximations can become tighter with more heuristic choices of α.

Based on the previous descriptions, we define the LRPM as a model in which an implementation is represented by a factor graph, its leakages are captured by the RPM and the exploitation of its leakages is quantified thanks to the local propagation rules of Equations 5, 6 and 21 (and possibly extension thereof for other operations).

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 start by considering the implementation of four S-boxes with the MixColumns operation in Figure 5.

4.1 Factor graph generation

In the original SASCA description of [VGS14, GS15], 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. 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,

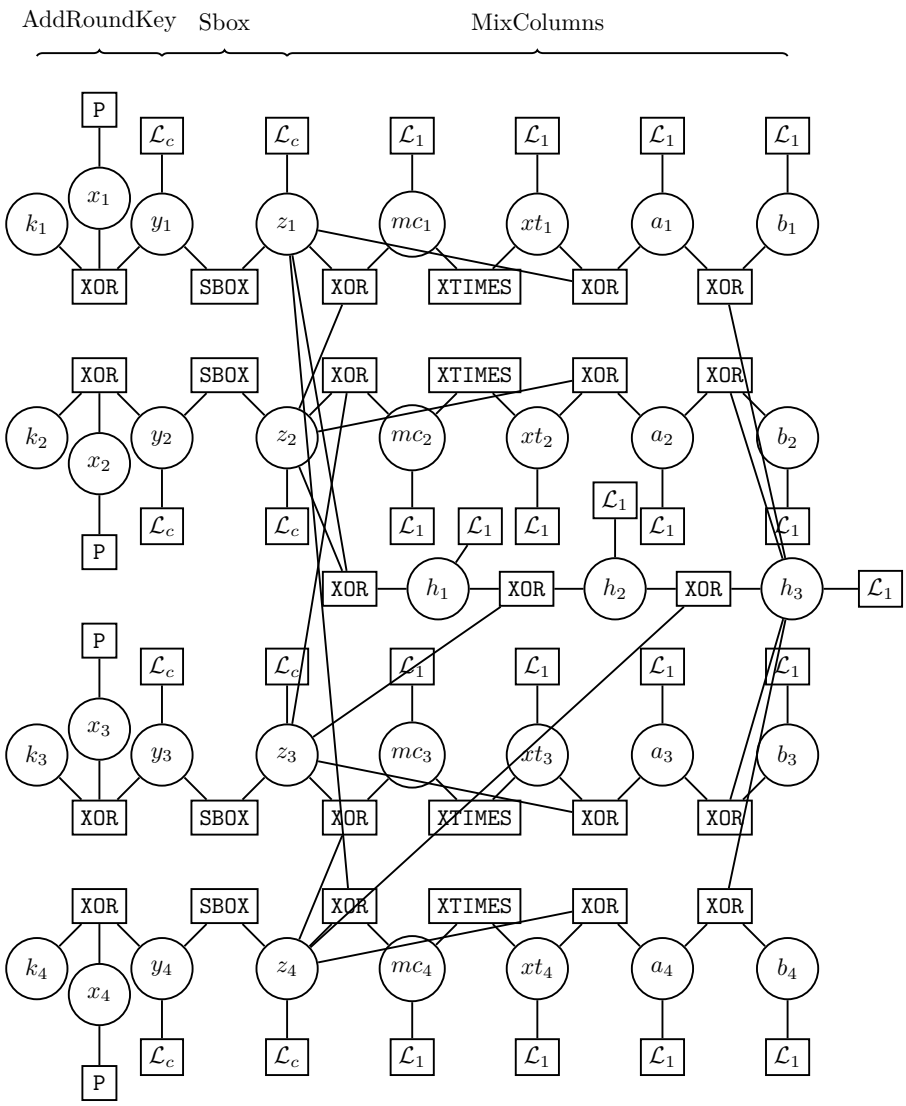


Figure 5: Factor graph for the first round / first column of the AES (adapted from [GS15]).

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 function or bijection. 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 by reducing their noise level.

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 explain in the bottom part of Figure 7 the correspondence between the merged nodes in Figure 7 and the original variable nodes in Figure 5. For instance, its first row of the first column means that the node v_1 in Figure 7 is the merger of three random variables in Figure 5: k_1, y_1 and

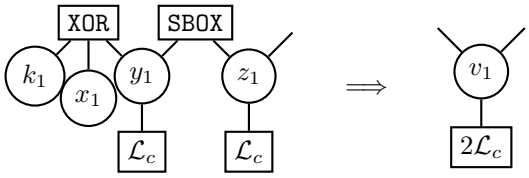


Figure 6: Illustration of the merging trick.

z_1. The leakages are combined accordingly. As in the previous section, all the leakages are single-shot except the (continuous) ones that correspond to variable nodes and 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). The figure also includes the number of leakage channels. We can see that the factor graph used for the BP algorithm is 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 Analysis and results

Following the usual approach in simulated side-channel attacks, we analyzed the case where all the target intermediate variables leak some deterministic function of their value, with some additive Gaussian noise. From this simple setting, we can compute a MI value according to Definition 1 and normalize it. We investigated the following parameters:

1. *Different number of rounds of leakage exploited.* We evaluated how the data complexity of SASCA is reduced by using the leakage of more block cipher rounds.

2. *Different deterministic parts of the leakage function.* We considered the standard Hamming weight leakage function and a random one (the same as used in [DFS19]), with the goal to identify possible dependencies of the model on this function.

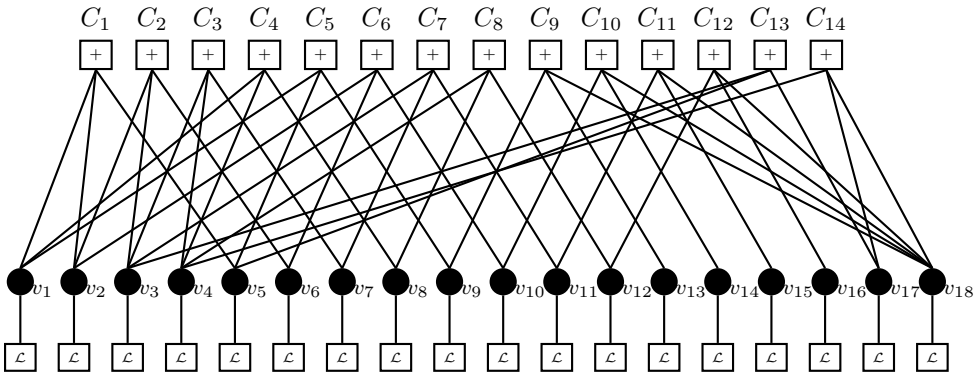
3. *Different noise levels.* We considered lower and higher noise levels, in order to evaluate how the improvements of SASCA over TA depend on this parameter.

4. *Known or unknown plaintext attacks.* We considered both the case of an attack with known plaintexts as in the standard DPA setting and the unknown plaintext context which may be considered in certain applications of side-channel attacks [HTM09].

A pseudo-code corresponding to one AES round is given in Appenxix B. The result of our first set of experiments is given in Figure 8. It corresponds to the simplest case of an attack in the known plaintext context, for a value of $\epsilon = 0.05$. We launched the attacks with both Hamming weight and random leakages. The success rate curves in the right part of the figure confirm that both for TA (which is known [MOS11]) and for SASCA, the attack complexity is independent of the shape of the leakage function. They also confirm the observation in [GRO18] that digging beyond the second round of the block cipher implementation does not lead to significant improvements in the known plaintext context. The model predictions are given on the left part of the figure. They provide the same intuitions and are conservative (as expected due to their worst-case nature).[3]

The result of our second set of experiments is given in Figure 9. It considers the same (known plaintext) attack context with a higher noise level (i.e., $\epsilon = 0.01$). The main

[3] Practical attacks are reported with their success rate while the model is expressed with information bounds. Yet, we note that in the case where the information is bounded in function of the number of measurements, as in the LRPM, the success rate is directly bounded by the MI (see [DFS19], Equation 19).



- $v_1 := y_1,\ z_1,\ k_1$; 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 $:= \mathcal{L}_1$;

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

conclusion of these plots is that while the increased noise increases the complexity of the attacks (as reflected by the X axes), the improvement that SASCA brings over TA is independent of the noise. This was already observed in [VGS14] for practical attacks and is confirmed with the model. This observation is particularly relevant in practice because it implies that less conservative estimates than our worst-case predictions can generally be considered by adding some factor to the data complexity predicted by the model.

Next, the move to an unknown plaintext scenario is reported in Figure 10 (for $\epsilon = 0.5$). This time, the main conclusion is that digging deeper in the block cipher rounds is significantly more useful (confirming the observations in [RSV09]), which matches the intuition that in this case, all the cipher rounds are equally useful to the adversary.

This last set of experiment is also a good one to confirm the significant gains in evaluation time that the model allows. For example, running a single 10-round SASCA on the 32 cores of an Intel Haswell architecture takes 7.5 minutes on average with our prototype code (and the success rate curve which is computed over 40 independent experiments for this number of rounds therefore took 5 hours to be generated). The bounds provided by the model are produced in seconds of computation on the same platform.

We finally checked the impact of the α parameter (from Section 3.4) in these various experiments and concluded that small variations of it have limited impact on our conclusions. For simplicity, we therefore use a value of $\alpha = 1$ in the remaining of the paper.

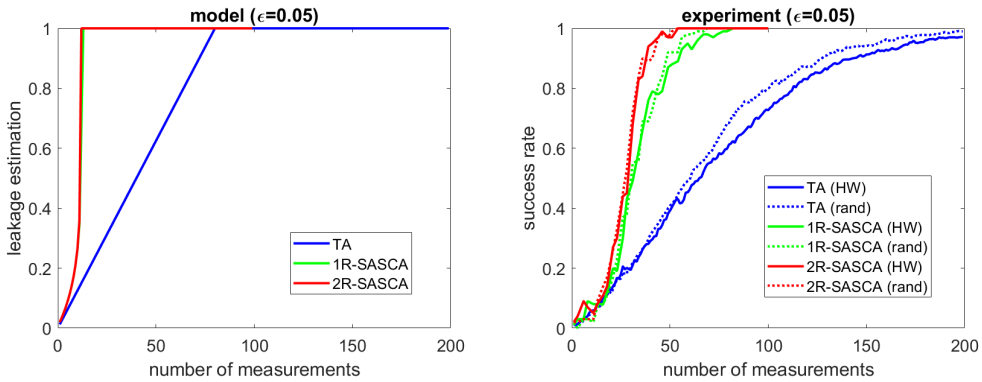


Figure 8: Leakage bounds and success rates of experimental SASCA with up to two rounds against an unprotected AES implementation in a known plaintext scenario (I).

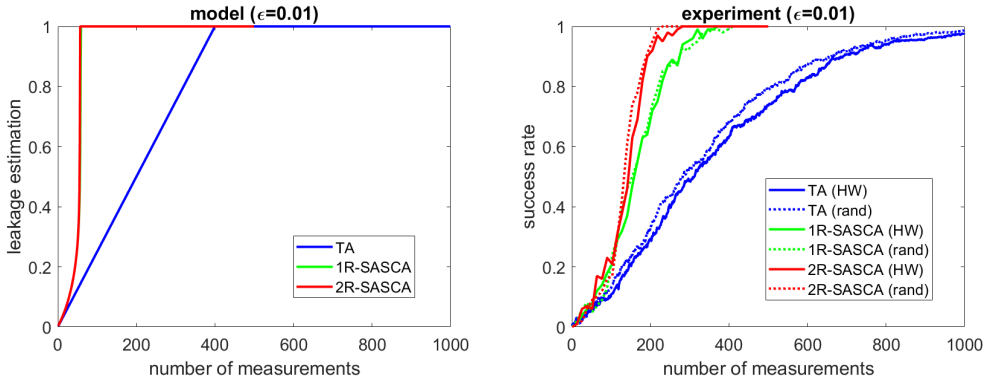


Figure 9: Leakage bounds and success rates of experimental SASCA with up to two rounds against an unprotected AES implementation in a known plaintext scenario (II).

4.3 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 matrix $\mathbf{H}_{14\times 18}$ shown in Figure 11, 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 $\mathbf{H}_{14\times 18}$ explains why BP performs well. Note that this parity-check matrix becomes larger if we have more traces, but its sparsity remains. Supposing N traces are observed, from one column of the first AES round we could generate a parity-check matrix of row size $14N$ and column size $(4 + 14N)$, leading to a $[4 + 14N, 4]_q$ LDPC code. Note also that this transformation does not mean that the description of AES is no longer non-linear. It only means that this transform could hide the non-linearity of AES by the merge trick that locally combines the leakages connected by a non-linear operation (e.g., an S-box). Therefore, the (in)vulnerability of AES against SASCA can be characterized by the decoding performances of one particular LDPC code derived from the inherent structure of the cipher. When N increases, the dimension of the derived code is invariant, i.e., equivalent to the key size. Its length,

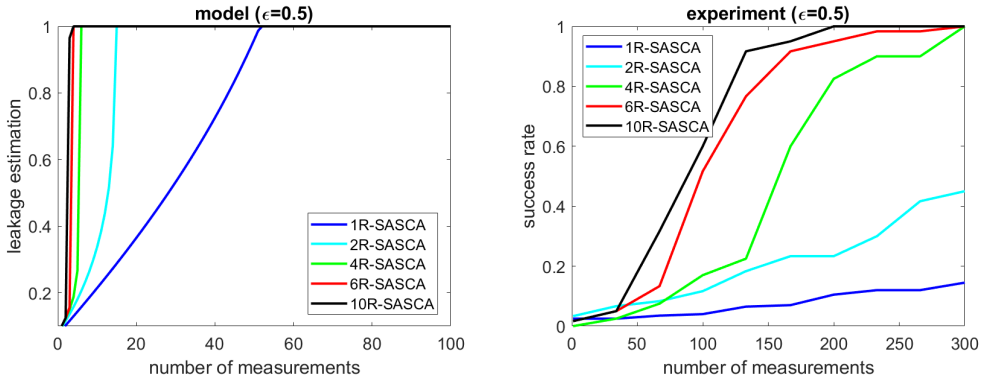


Figure 10: Leakage bounds and success rates of experimental SASCA with up to ten rounds against an unprotected AES implementation in a unknown plaintext scenario.

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. 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 codes have provably poor decoding performances).

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

Figure 11: Parity-check matrix of the LDPC code corresponding to Figure 7.

5 Protected implementations

We next move to our second main contribution and extend our analysis and modeling of SASCA to the more challenging case of protected implementations, for which higher security levels can be expected and therefore shortcut approaches to simplify the security evaluations are increasingly needed. We start with masking and show that our concrete

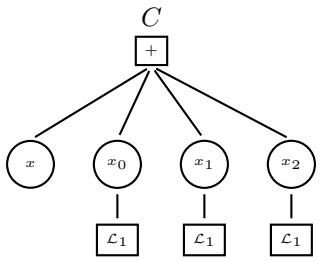


Figure 12: Masked encoding with $d = 3$ shares.

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 and optimize the tradeoff between the amount of physical noise and mathematical randomness in masked gadgets. We follow with shuffling and demonstrate similar gains.

5.1 Masked encoding

The Tanner graph of a masked encoding with three shares is given in Figure 12. In general, it corresponds to a secret value x 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 byte) of λ, leading to a set of equations:

$$p_{x_i \to c} = \lambda \tag{22}$$

$$r_{c \to x} = p^d_{x_i \to c} \cdot \alpha^{(d-1)}, \tag{23}$$

$$\mathcal{B} = N \cdot r_{c \to x}, \tag{24}$$

where N is the number of measurements used in the attack. The resulting approximation is similar to the bounds given in masking proofs (excepted that our modeling considers the leakages per byte rather than the mutual information). Note that in this simple context, the application of the BP algorithm is equivalent (from the information theoretic viewpoint) to the application of a multivariate TA if the noise covariance matrix is diagonal.

5.2 Masked multiplication

Given some encoded values, masked multiplications usually apply some processing to the shares, based on secure addition and multiplication algorithms [ISW03]. For illustration, we first consider a factor graph for a secure multiplication algorithm similar to the one investigated in [GS18] and depicted in Figure 13 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 $\mathcal{L}_\mathrm{s}$).[4] We can consider two leakage assumptions for this purpose:

- *First assumption:* single 1-shot leakages (i.e., $\mathcal{L}_\mathrm{s} = \lambda$).
- *Second assumption:* multiple 1-shot leakages (i.e., $\mathcal{L}_\mathrm{s} = d \cdot \lambda$) due to the need to manipulate each share d times in the secure multiplication algorithm of [ISW03].

The g function in Figure 13 corresponds to the (bijective) square operations of an AES S-box implementation (see Section 5.3). Hence, 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:

[4] As discussed in [GS18], the latter are necessary to initialize the shares' leakage.

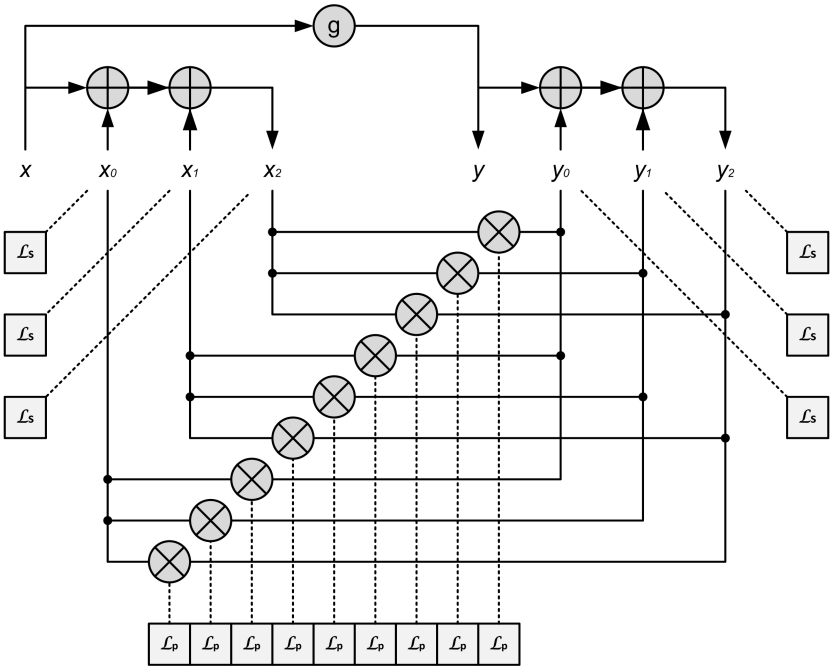


Figure 13: Factor graph for a secure multiplication with three shares.

$$r^{(0)}_{s\to x_i} = 0 \quad (25)$$
$$r^{(0)}_{s\to x} = 0 \quad (26)$$
$$r^{(0)}_{p\to x_i} = 0 \quad (27)$$

$$p^{(t)}_{x_i\to s} = \mathcal{L}_s + d\cdot r^{(t-1)}_{p\to x_i} \quad (28)$$
$$p^{(t)}_{x_i\to p} = \mathcal{L}_s + (d-1)\cdot r^{(t-1)}_{p\to x_i} + r^{(t-1)}_{s\to x_i} \quad (29)$$
$$p^{(t)}_{x\to s} = (2N-1)\cdot r^{(t-1)}_{s\to x} \quad (30)$$
$$r^{(t)}_{s\to x} = \left(p^{(t)}_{x_i\to s}\right)^{d}\cdot \alpha^{(d-1)} \quad (31)$$
$$r^{(t)}_{s\to x_i} = \left(p^{(t)}_{x_i\to s}\right)^{(d-1)}\cdot p^{(t)}_{x\to s}\cdot \alpha^{(d-1)} \quad (32)$$
$$r^{(t)}_{p\to x_i} = p^{(t)}_{x_i\to p}\cdot \mathcal{L}_p\cdot \beta \quad (33)$$
$$\mathcal{B}^{(t)} = 2N\cdot r^{(t)}_{s\to x} \quad (34)$$

Thanks to the symmetry of Figure 13 , we only need to write equations for the x variables. The β factor plays the same role as the α for the multiplications (we set both to one for simplicity in this experiment – small variations do not affect our conclusions). We also mention the factor $(2N-1)$ in Equation 30 for which N edges come from the y shares and $(N-1)$ edges come from the x shares. Similarly in Equation 34, the factor 2 comes from the combination of the leakages on x and y thanks to the bijective g.

One possibly suprising element of these equations is the multiplicative local estimation rule of Equation 33. One could indeed expect a higher extrinsic information than predicted by this rule due to the well-known "zero problem" of multiplicative masking schemes [GT02]. However, a reasoning based on the random probing model shows that the situation is actually different here. Suppose that we have three random variables X, Y and Z over

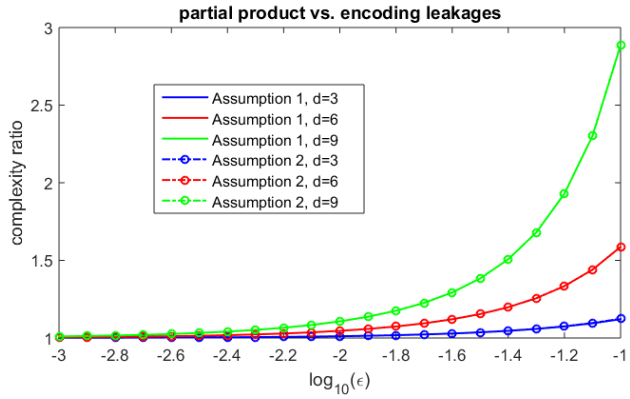


Figure 14: Information gain of masked multiplications.

$\mathrm{GF}(q)$, 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 $1-(1-e_1)(1-e_2)$ which is reported as $1-\mathrm{MI}_1 \cdot \mathrm{MI}_2$ in the LRPM. So the capacity of the new channel is at most $\mathrm{MI}_1 \cdot \mathrm{MI}_2$, therefore justifying Equation 33.

Based on these equations, we first show in Figure 14 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 assumptions stated earlier in this section. By contrast, for too low noise levels these additional leakages become significant, especially for large d values. A similar view can be extracted from Figure 15 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 14). Overall, these experiments illustrate that the LRPM allows us to revisit the key intuitions of masking security proofs [PR13, DDF14, DFS19, GS18], 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 a leaking target implementation.

A note on the BP algorithm for masked implementations. In the case of unprotected implementations like reported in Section 4.2, running the BP algorithm on a factor graph connecting all the manipulated plaintexts (i.e., so-called graph extension) usually leads to better results than running it on several smaller factor graphs that are then re-combined (i.e., so-called graph combination). Interestingly, we observed that the latter fact does not hold for higher-order masked implementation with more than two shares. The key reason of this 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 can be effective for unprotected implementations

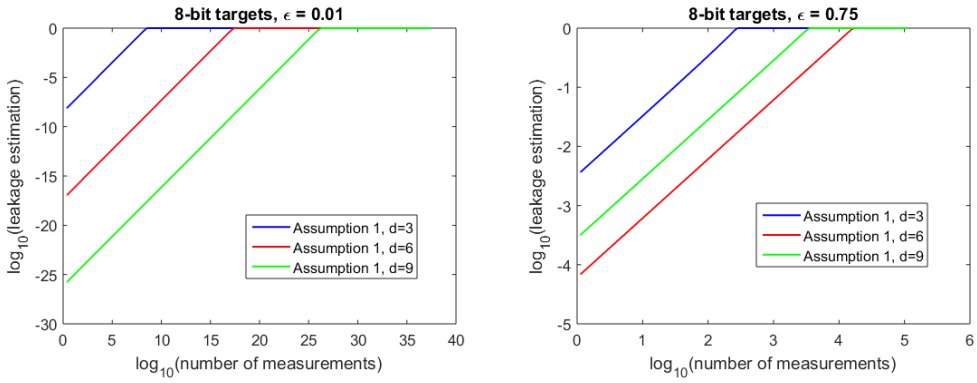


Figure 15: Leakage bounds for the factor graph of Figure 13 (leakage assumption 1).

and first-order masked implementations, but becomes useless for higher orders.

We confirmed this theoretical explanation by running again the experimental SASCA against a (3-share) masked multiplication carried out in [GS18], 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 [GS18], 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 presumably makes the presence of cycles more detrimental.

5.3 Masked S-box implementations

We complement this discussion on masking by illustrating 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 16 (originally proposed in [RP10] but here tweaked with the refreshing gadgets of [ISW03] to avoid the attacks put forward in [CPRR13]). 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 [BCPZ16]. 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 $(2 \cdot \lambda)^d$. Additionally considering the fact that the shares of y_1 and y_2 are 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, one important message is that besides being useful for composability issues [BBD+16], the addition of refreshing gadgets may also be needed to mitigate the noise reduction in masked implementations that SASCA enables. That is, adding refreshing gadgets (which

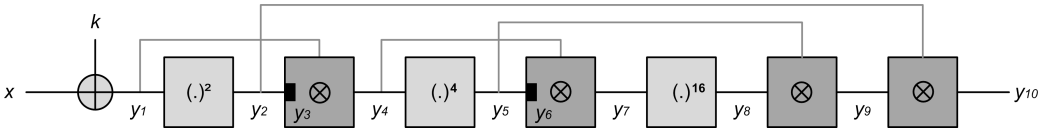


Figure 16: AES S-box from [RP10] (black rectangles are refreshing gadgets from [ISW03]).

are represented by the black rectangles in Figure 16) is necessary whenever multiplying dependent values. The requirements for such (composable) refreshing gadgets are typically high in randomness. But adding a simpler refreshing (e.g., a sharing of zero) after each operation in Figure 16 would also prevent the combination of certain leakages. That is, the merge of y_1 and y_2 would 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 further combinations would be prevented.

5.4 Other (follow up) results

Improving the security of multiplication algorithms against horizontal attacks and optimizing the tradeoff between physical noise and mathematical randomness is one of the main research challenges in masking and leakage-resilience. First progresses in this direction have been made by Batistello et al. [BCPZ16]. Their analysis is based on the qualitative criteria that minimizing the number times each share is manipulated limits the aforementioned noise reductions, and it was validated using concrete (SASCA-like) attacks on a few representative examples (due to the high complexity of mounting these attacks).

In a recent work from TCHES 2019, it was shown that the LRPM can be used to systematize this analysis and making it quantitative, and as a result to identify efficient multiplication gadgets with constant noise rate [CS19] – hence confirming its relevance for reasoning formally about the concrete security of masked implementations.

5.5 Shuffling

We conclude the paper by analyzing the security of another popular countermeasure against side-channel attacks, namely the shuffling introduced in [HOM06]. Its goal is to randomize the execution of the operations: we next study the standard case of a shuffling over $N_s = 16$ S-box executions. For this purpose, we assume that a permutation P is picked up at the beginning of each cipher execution. Then, the first plaintext and subkey manipulated are $x_{\mathsf{P}(1)}$ and $k_{\mathsf{P}(1)}$, followed by $x_{\mathsf{P}(2)}$ and $k_{\mathsf{P}(2)}$, and so on. Two types of permutations are frequently considered in the literature: either the permutation is picked up from the set of all permutations, or a Random Start Index (RSI) is used, in which case the operations are performed in fixed order, but for each execution the first operation is different.

As analyzed in [VMKS12], the worst-case evaluation of the shuffling countermeasure requires computing intensive (N_s-dimension) integrals. We next detail how a much simplified analysis can be based on the LRPM. For this purpose, we first observe that in a shuffled implementation, the key is not used at a fixed time. Hence, the attacker cannot directly accumulate information on it, and has to exploit two kinds of leakages. First, so-called permutation leakages $\mathcal{L}_{\mathsf{P}_i} \leftsquigarrow \mathsf{P}(i)$ that give hints about the execution time. We consider that the attacker can observe ϵ_{P} bits of information on each $\mathsf{P}(i)$ value. Next, the S-boxes computation leakages: using our previous notations (e.g., from Figure 1), it leads to $\mathcal{L}_{t_i} \leftsquigarrow z_{\mathsf{P}(i)}$. The use of the permutation P is here to highlight the fact that at position t_i in the trace we have leakage about the $\mathsf{P}(i)$-th variable and not the (i)-th variable. We consider that the attacker can observe ϵ bits of information on each S-box output. Based

on these notations, we can bound the information exploited by different attacks against a shuffled implementation. We study the next cases from [VMKS12]:

1. A TA against an unprotected device (UNP-TA) for reference. In this case, the operations are performed in a fix order and $\mathcal{B} = \frac{N \cdot \epsilon}{8}$ as previously.

2. A TA against a shuffled implementation without information on the permutation (UNI-TA). In this case, $\epsilon_{\mathsf{P}} = 0$ and the attacker assumes that all permutations are equally likely, which boils down to divide the information on the S-box by the permutation size N_s, leading to an information leakage bound $\mathcal{B} = \frac{N \cdot \epsilon}{8 \cdot N_s}$.

3. A TA against a shuffled implementation with an adversary exploiting the ϵ_{P} bits of "direct permutation leakage" (DPLEAK-TA). In the case, the factor N_s in the previous bound is replaced by $2^{\log_2(N_s)-\epsilon_{\mathsf{P}}}$, which corresponds to the uncertainty remaining on the execution times after the observation of the permutation leakages, leading to another information leakage bound $\mathcal{B} = \frac{N \cdot \epsilon}{8 \cdot 2^{\log_2(N_s)-\epsilon_{\mathsf{P}}}}$.

4. A similar TA against an implementation shuffled with a RSI (RSIENUM-TA). In this case, due to the structure of the RSI, there is only $\log_2(N_s)$ bits of secret for all the N_s S-box executions and all the permutation leakages can be combined to learn it. This essentially provides N_s times more information on the execution time, leading to a last information leakage bound $\mathcal{B} = \frac{N \cdot \epsilon}{8 \cdot 2^{\log_2(N_s)-N_s \cdot \epsilon_{\mathsf{P}}}}$.

Figure 17 illustrates our analyzes with this model. For simplicity, we set the S-box leakages to $\epsilon = 0.1$ and vary the permutation leakages. We observe that the conclusions of [VMKS12] are reached in a very efficient manner. In high information (i.e., low-noise) contexts, the security level is low, the RSI shuffling is as weak as an unprotected implementation and the shuffling with uniform permutation does not deliver the expected gains (its security is closer from the UNP-TA than from the UNI-TA). In low information contexts, the opposite observation holds and all shufflings lead to a similar amplification of the noise by a factor N_s. In view of these results and the one in the previous subsection, designing gadgets combining masking and shuffling in order to improve security against horizontal attacks, and evaluating them with the LRPM, is an interesting research direction.

6 Summarizing remarks

In this work, we proposed to model SASCA from a coding theory viewpoint, leveraging the analogy between the random probing model used in masking security proofs and an erasure channel. The model leads to a collection of new results and simpler interpretations of known facts. Its main concrete interest is to provide a considerably faster approach to analyze the security of practical implementations against (horizontal, multi-target) SASCA. By enabling the exploration of a larger design space, it can also lead to better optimizations of the tradeoff between the noise level and the randomness complexity of such implementations. Other contributions include the better interpretation of masking security bounds (e.g., how much the security of an implementation decreases with its number of operations, in the known and unknown plaintext scenarios), and new evaluations of the shuffling countermeasure. We hope these results can serve as a seed for the systematic investigation of combined countermeasures against (nearly worst-case) SASCA.

Acknowledgments. The main part of Qian Guo's work was done when he was a post-doctoral researcher at the ICTEAM/ELEN/Crypto Group, UCL, Louvain-la-Neuve, Belgium. François-Xavier Standaert is a senior associate researcher of the Belgian Fund for

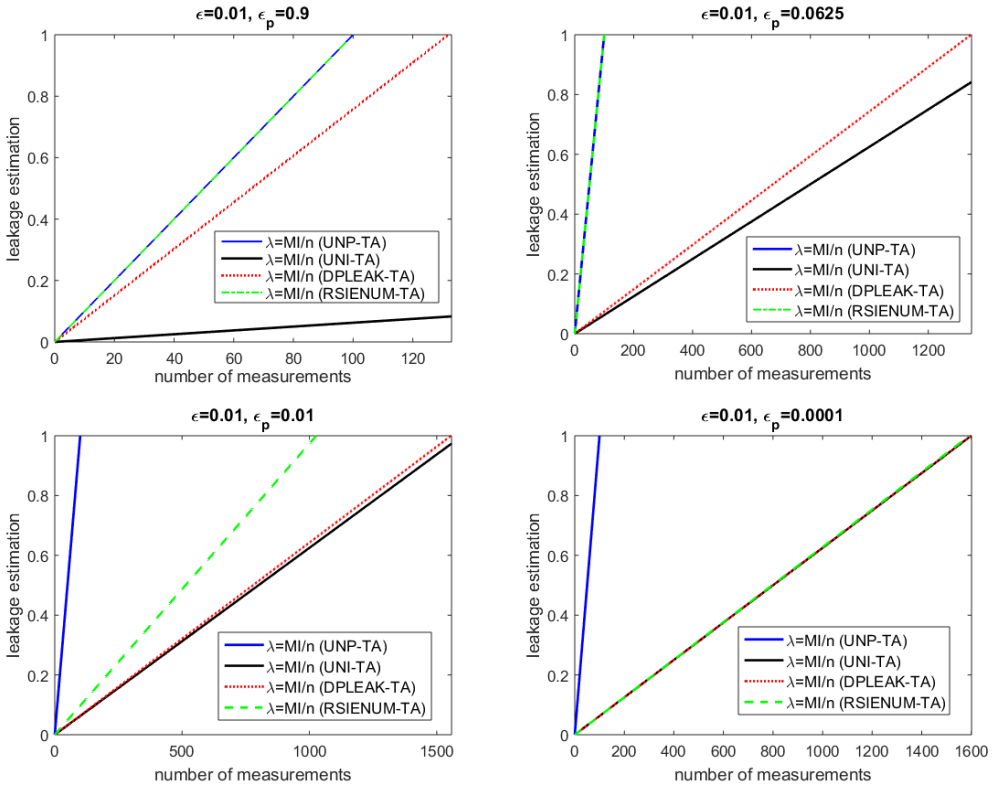


Figure 17: Leakage bounds for shuffled implementations.

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. Qian Guo was also partially supported by the Norwegian Research Council (Grant No. 247742/070), and by the Wallenberg Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[BBD+16] 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.

[BBP+16] 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 *LNCS*, pages 616–648. Springer, 2016.

[BBP+17] 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 *LNCS*, pages 397–426. Springer, 2017.

[BCPZ16] 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 *LNCS*, pages 23–39. Springer, 2016.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019.

[CPRR13] 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 *LNCS*, pages 410–424. Springer, 2013.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Jr. et al. [JKP03], pages 13–28.

[CS19] Gaetan Cassiers and François-Xavier Standaert. Towards globally optimized masking: From low randomness to low noise rate or probe isolating multiplications with reduced randomness and security against horizontal attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):162–198, 2019.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Nguyen and Oswald [NO14], pages 423–440.

[DFS19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *J. Cryptology*, 32(4):1263–1297, 2019.

[DSV14] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Nguyen and Oswald [NO14], pages 459–476.

[Gal62] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Information Theory*, 8(1):21–28, 1962.

[GRO18] Joey Green, Arnab Roy, and Elisabeth Oswald. A systematic study of the impact of graphical models on inference-based attacks on AES. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *LNCS*, 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 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 *LNCS*, pages 291–312. Springer, 2015.

[GS18] 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 *LNCS*, pages 385–412. Springer, 2018.

[GT02] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In Jr. et al. [JKP03], pages 198–212.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *LNCS*, pages 239–252, 2006.

[HTM09] Neil Hanley, Michael Tunstall, and William P. Marnane. Unknown plaintext template attacks. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications, 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*, volume 5932 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2009.

[ISW03] 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 *LNCS*, pages 463–481. Springer, 2003.

[JKP03] Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *LNCS*. Springer, 2003.

[Mac03] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[MOS11] 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.

[MOW14] Luke Mather, Elisabeth Oswald, and Carolyn Whitnall. Multi-target DPA attacks: Pushing DPA beyond the limits of a desktop computer. In Sarkar and Iwata [SI14], pages 243–261.

[NO14] 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 *LNCS*. Springer, 2014.

[Pea82] 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.

[PR13] 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 *LNCS*, pages 142–159. Springer, 2013.

[RP10] 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 *LNCS*, pages 413–427. Springer, 2010.

[RSV09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: why time also matters in DPA. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.

[RU08] Thomas J. Richardson and Rüdiger L. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[SI14] 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 *LNCS*. Springer, 2014.

[SLP05] 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 *LNCS*, pages 30–46. Springer, 2005.

[SMY09] 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 *LNCS*, pages 443–461. Springer, 2009.

[Tan81] Robert Michael Tanner. A recursive approach to low complexity codes. *IEEE Trans. Information Theory*, 27(5):533–547, 1981.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Sarkar and Iwata [SI14], pages 282–296.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *LNCS*, pages 740–757. Springer, 2012.

A 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:**
$\mathrm{MI_{dc}}(1) = 0;$
for $i = 1 : Nmax$
  $\mathrm{MI_{dc}}(i+1) = \mathrm{MI_{dc}}(i) + (Nd * e/n);$
end

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

B Pseudo-code for the first AES round

$n = 8;$ `- n-bit targets`
$e = 0.1;$ `- MI per target operation`
$Nmax = n/(Nd * e);$ `- max. data complexity for the plots`

$Nmax = ceil(Nmax * (1.1))$;
$e = e/n$; - MI per target operation measured by bytes

– **Analytical approach:**
$Nit = 100$; - # of iterations (a safe choice.)
$\mathrm{MI_a}(1) = 0$;
$e_1 = 2 * e$; - Leakages from Variable v_1
$e_2 = 2 * e$;
$e_3 = 2 * e$;
$e_4 = 2 * e$;
$e_5 = 3 * e$;
$e_6 = 2 * e$;
$e_7 = 2 * e$;
$e_8 = 2 * e$;
$e_9 = e$;
$e_{10} = e$;
$e_{11} = e$;
$e_{12} = e$;
$e_{13} = e$;
$e_{14} = e$;
$e_{15} = e$;
$e_{16} = e$;
$e_{17} = e$;
$e_{18} = e$;
for $i = 1 : Nmax$
  $R_{c1v1} = 0$; - MI from Check C_1 to Variable v_1
  $R_{c1v2} = 0$;
  $R_{c1v5} = 0$;
  $R_{c2v2} = 0$;
  $R_{c2v6} = 0$;
  $R_{c2v3} = 0$;
  $R_{c3v3} = 0$;
  $R_{c3v4} = 0$;
  $R_{c3v7} = 0$;
  $R_{c4v1} = 0$;
  $R_{c4v4} = 0$;
  $R_{c4v8} = 0$;
  $R_{c5v1} = 0$;
  $R_{c5v5} = 0$;
  $R_{c5v9} = 0$;
  $R_{c6v2} = 0$;
  $R_{c6v6} = 0$;
  $R_{c6v10} = 0$;
  $R_{c7v3} = 0$;
  $R_{c7v7} = 0$;
  $R_{c7v11} = 0$;
  $R_{c8v4} = 0$;
  $R_{c8v8} = 0$;
  $R_{c8v12} = 0$;
  $R_{c9v9} = 0$;
  $R_{c9v18} = 0$;
  $R_{c10v10} = 0$;

$R_{c10v18} = 0;$
$R_{c11v11} = 0;$
$R_{c11v18} = 0;$
$R_{c12v12} = 0;$
$R_{c12v18} = 0;$
$R_{c13v3} = 0;$
$R_{c13v5} = 0;$
$R_{c13v17} = 0;$
$R_{c14v4} = 0;$
$R_{c14v17} = 0;$
$R_{c14v18} = 0;$
for $j = 1 : Nit$
$P_{v1c1} = i * e_1 + (i-1) * R_{c1v1} + i * R_{c4v1} + i * R_{c5v1};$
`- MI from Variable` v_1 `to Check` C_1
$P_{v1c4} = i * e_1 + (i-1) * R_{c4v1} + i * R_{c1v1} + i * R_{c5v1};$
$P_{v1c5} = i * e_1 + (i-1) * R_{c5v1} + i * R_{c1v1} + i * R_{c4v1};$
$P_{v2c1} = i * e_2 + (i-1) * R_{c1v2} + i * R_{c2v2} + i * R_{c6v2};$
$P_{v2c2} = i * e_2 + (i-1) * R_{c2v2} + i * R_{c1v2} + i * R_{c6v2};$
$P_{v2c6} = i * e_2 + (i-1) * R_{c6v2} + i * R_{c2v2} + i * R_{c1v2};$
$P_{v3c2} = i * e_3 + (i-1) * R_{c2v3} + i * R_{c3v3} + i * R_{c7v3} + i * R_{c13v3};$
$P_{v3c3} = i * e_3 + (i-1) * R_{c3v3} + i * R_{c2v3} + i * R_{c7v3} + i * R_{c13v3};$
$P_{v3c7} = i * e_3 + (i-1) * R_{c7v3} + i * R_{c3v3} + i * R_{c2v3} + i * R_{c13v3};$
$P_{v3c13} = i * e_3 + (i-1) * R_{c13v3} + i * R_{c3v3} + i * R_{c7v3} + i * R_{c2v3};$
$P_{v4c3} = i * e_4 + (i-1) * R_{c3v4} + i * R_{c4v4} + i * R_{c8v4} + i * R_{c14v4};$
$P_{v4c4} = i * e_4 + (i-1) * R_{c4v4} + i * R_{c3v4} + i * R_{c8v4} + i * R_{c14v4};$
$P_{v4c8} = i * e_4 + (i-1) * R_{c8v4} + i * R_{c4v4} + i * R_{c3v4} + i * R_{c14v4};$
$P_{v4c14} = i * e_4 + (i-1) * R_{c14v4} + i * R_{c4v4} + i * R_{c8v4} + i * R_{c3v4};$
$P_{v5c1} = e_5 + R_{c5v5} + R_{c13v5};$
$P_{v5c5} = e_5 + R_{c1v5} + R_{c13v5};$
$P_{v5c13} = e_5 + R_{c5v5} + R_{c1v5};$
$P_{v6c2} = e_6 + R_{c6v6};$
$P_{v6c6} = e_6 + R_{c2v6};$
$P_{v7c3} = e_7 + R_{c7v7};$
$P_{v7c7} = e_7 + R_{c3v7};$
$P_{v8c4} = e_8 + R_{c8v8};$
$P_{v8c8} = e_8 + R_{c4v8};$
$P_{v9c5} = e_9 + R_{c9v9};$
$P_{v9c9} = e_9 + R_{c5v9};$
$P_{v10c6} = e_{10} + R_{c10v10};$
$P_{v10c10} = e_{10} + R_{c6v10};$
$P_{v11c7} = e_{11} + R_{c11v11};$
$P_{v11c11} = e_{11} + R_{c7v11};$
$P_{v12c8} = e_{12} + R_{c12v12};$
$P_{v12c12} = e_{12} + R_{c8v12};$
$P_{v17c13} = e_{17} + R_{c14v17};$
$P_{v17c14} = e_{17} + R_{c13v17};$
$P_{v18c9} = e_{18} + R_{c10v18} + R_{c11v18} + R_{c12v18} + R_{c14v18};$
$P_{v18c10} = e_{18} + R_{c9v18} + R_{c11v18} + R_{c12v18} + R_{c14v18};$
$P_{v18c11} = e_{18} + R_{c10v18} + R_{c9v18} + R_{c12v18} + R_{c14v18};$
$P_{v18c12} = e_{18} + R_{c10v18} + R_{c11v18} + R_{c9v18} + R_{c14v18};$
$P_{v18c14} = e_{18} + R_{c10v18} + R_{c11v18} + R_{c12v18} + R_{c9v18};$
$R_{c1v1} = P_{v2c1} * P_{v5c1} * \alpha;$

$R_{c1v2} = P_{v1c1} * P_{v5c1} * \alpha;$
$R_{c1v5} = P_{v2c1} * P_{v1c1} * \alpha;$
$R_{c2v2} = P_{v3c2} * P_{v6c2} * \alpha;$
$R_{c2v6} = P_{v3c2} * P_{v2c2} * \alpha;$
$R_{c2v3} = P_{v2c2} * P_{v6c2} * \alpha;$
$R_{c3v3} = P_{v4c3} * P_{v7c3} * \alpha;$
$R_{c3v4} = P_{v3c3} * P_{v7c3} * \alpha;$
$R_{c3v7} = P_{v4c3} * P_{v3c3} * \alpha;$
$R_{c4v1} = P_{v4c4} * P_{v8c4} * \alpha;$
$R_{c4v4} = P_{v1c4} * P_{v8c4} * \alpha;$
$R_{c4v8} = P_{v4c4} * P_{v1c4} * \alpha;$
$R_{c5v1} = P_{v5c5} * P_{v9c5} * \alpha;$
$R_{c5v5} = P_{v1c5} * P_{v9c5} * \alpha;$
$R_{c5v9} = P_{v5c5} * P_{v1c5} * \alpha;$
$R_{c6v2} = P_{v6c6} * P_{v10c6} * \alpha;$
$R_{c6v6} = P_{v2c6} * P_{v10c6} * \alpha;$
$R_{c6v10} = P_{v6c6} * P_{v2c6} * \alpha;$
$R_{c7v3} = P_{v7c7} * P_{v11c7} * \alpha;$
$R_{c7v7} = P_{v3c7} * P_{v11c7} * \alpha;$
$R_{c7v11} = P_{v7c7} * P_{v3c7} * \alpha;$
$R_{c8v4} = P_{v8c8} * P_{v12c8} * \alpha;$
$R_{c8v8} = P_{v4c8} * P_{v12c8} * \alpha;$
$R_{c8v12} = P_{v8c8} * P_{v4c8} * \alpha;$
$R_{c9v9} = e_{13} * P_{v18c9} * \alpha;$
$R_{c9v18} = e_{13} * P_{v9c9} * \alpha;$
$R_{c10v10} = e_{14} * P_{v18c10} * \alpha;$
$R_{c10v18} = e_{14} * P_{v10c10} * \alpha;$
$R_{c11v11} = e_{15} * P_{v18c11} * \alpha;$
$R_{c11v18} = e_{15} * P_{v11c11} * \alpha;$
$R_{c12v12} = e_{16} * P_{v18c12} * \alpha;$
$R_{c12v18} = e_{16} * P_{v12c12} * \alpha;$
$R_{c13v3} = P_{v5c13} * P_{v17c13} * \alpha;$
$R_{c13v5} = P_{v3c13} * P_{v17c13} * \alpha;$
$R_{c13v17} = P_{v5c13} * P_{v3c13} * \alpha;$
$R_{c14v4} = P_{v18c14} * P_{v17c14} * \alpha;$
$R_{c14v17} = P_{v18c14} * P_{v4c14} * \alpha;$
$R_{c14v18} = P_{v4c14} * P_{v17c14} * \alpha;$
$\mathcal{B}(j+1) = i * e_1 + i * R_{c1v1} + i * R_{c4v1} + i * R_{c5v1};$
end
$\mathrm{MI_a}(i+1) = max(\mathcal{B});$
end