A Methodology to Achieve Provable Side-Channel Security in Real-World Implementations

Sonia Belaïd¹, Gaëtan Cassiers³, Camille Mutschler^{4,6}, Matthieu Rivain¹, Thomas Roche⁴, François-Xavier Standaert⁵, and Abdul Rahman Taleb^{1,2}

¹ CryptoExperts, Paris, France
 ² Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
 ³ TU Graz, Graz, Austria
 ⁴ NinjaLab, Montpellier, France
 ⁵ UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
 ⁶ LIRMM, Univ. Montpellier, CNRS, Montpellier, France

Abstract. Physical side-channel attacks exploit a device's emanations to compromise the security of cryptographic implementations. Many countermeasures have been proposed against these attacks, especially the widely-used and efficient masking countermeasure. While theoretical models offer formal security proofs, they often rest on unrealistic assumptions, leading current approaches to prove the security of masked implementations to primarily rely on empirical verification. Consequently, the literature still lacks a well-defined framework for implementing proven secure constructions on physical devices.

In this paper, we present a comprehensive methodology to transform an abstract circuit into a physical implementation secure against sidechannel attacks. We introduce new tools for adapting the ideal noisy leakage model to practical scenarios. We also highlight the design objectives for embedded devices to achieve high levels of security, while acknowledging the limitations and challenges in applying leakage models in practice. Our aim is to demonstrate the possibility of bridging theory and practice, encouraging further research to achieve practical implementations proven secure against side-channel attacks without relying on ideal assumptions about the leakage.

1 Introduction

Cryptographic algorithms' security is usually studied in the *black-box* model, where the adversary is limited to the knowledge of some inputs and outputs. However, as revealed in the late nineties [52], their implementation on physical devices can be vulnerable to *side-channel attacks*. Such attacks exploit the device's physical emanations, such as the execution time [52], device temperature [49], power consumption [53], or electromagnetic radiation [67] during the algorithm execution.

Since the discovery of side-channel attacks, several countermeasures have been studied to protect cryptographic algorithms. Among the different approaches, one of the most widely used is known as masking, simultaneously introduced by Chari et al. [29], and by Goubin and Patarin [46] in 1999. It consists in splitting a sensitive variable x into n random shares, among which any combination of n-1 shares does not reveal any secret information. This can be achieved by generating n-1 shares uniformly at random x_1, \ldots, x_{n-1} and computing the last share x_n so that $x = x_1 * \ldots * x_{n-1} * x_n$ according to some group law *. The motivation is to make it more difficult for an attacker to recover a secret by manipulating the shares instead of the sensitive value. Indeed, the adversary must recombine information from all the shares to learn something about the sensitive value. Assuming that the information gathered through side-channel observations involves some kind of noise, it has been shown that it becomes exponentially harder to recover the secret as the number of shares grows [10,23,48].

Meanwhile, proving or validating such security levels in practice is not trivial. Generally, providing security guarantees against side-channel attacks is tricky, and several works tackle this issue [40,51,47,23]. The approaches currently found in the literature range from purely qualitative solutions such as leakage detection (e.g., ISO17825 [1,75]) or test vector leakage assessment (TVLA) [45], which aim to detect information leakage using statistical analysis, to more quantitative solutions such as mounting known attacks on the implementation and inferring the security level from the best attacks. For instance, common-criteria certification procedures currently follow this empirical approach to validate the security of implementations for smartcards against side-channel attacks [23,24].

Having more formal and quantified security guarantees would be more satisfying but complicated as it relies on physical assumptions and mathematical arguments. The community introduced so-called *leakage models* to theoretically reason on the security of masked implementations. They aim to define the attacker's capabilities to counteract the subsequent side-channel attacks formally. The most famous one is the t-probing model, introduced by Ishai, Sahai, and Wagner in 2003 [50]. In this model, the leakage is modeled as the exact values of t intermediate variables chosen by the attacker for t < n, the number of shares. A circuit is then secure in this model if no such leakage of t variables reveals information about the sensitive variables. Despite its wide use by the community [73,68,34,12,35] thanks to its convenience to build security proofs, the t-probing model sometimes fails to reflect the reality of embedded devices. For instance, it does not capture *horizontal attacks* [9], which exploit the repeated manipulation of variables within an execution.

These issues motivated the formalization of the noisy leakage model [66]. This model better captures the reality of embedded devices by assuming that each intermediate variable leaks a noisy function of its value. However, proving security in the noisy leakage model [11,61] is more complex than in the t-probing model. In 2014, Duc, Dziembowski, and Faust [38,39] proposed a security reduction from the noisy model to the t-probing model, relying on an intermediate model, the random probing model, which benefits from a tighter reduction with the noisy leakage model. In a nutshell, it assumes that every wire in the circuit leaks with some constant leakage probability. This leakage

probability is related to the amount of side-channel noise in practice. The random probing model captures *horizontal attacks*, and has been studied recently in many works [3,5,4,13,15,16,26].

The noisy leakage, random probing, and t-probing models have proven helpful for the community to model side-channel attacks theoretically and provide formal security proofs on masked implementation. Meanwhile, applying these security proofs to real-world implementations is still challenging.

First, the theoretical literature lacks a proper methodology to implement proven secure constructions in the leakage models on a physical device while preserving the proven security levels. Second, these leakage models rely on two assumptions about the physical device for which a systematic investigation is lacking: the leakage of an elementary operation only depends on its inputs (*i.e.* the *data isolation* assumption), and the leakage's noise of an operation is independent of the previous and following noises (*i.e.* the *noise independence* assumption).

The first assumption (data isolation) can be easily broken, for instance, due to physical effects on a device. In particular, transitions occurring on memory buses or CPU registers between a previously processed value \mathbf{x}_{i-1} and the current one \mathbf{x}_i usually leak some information correlated to $\mathbf{x}_{i-1} \oplus \mathbf{x}_i$, which violates the data isolation principle [33,6]. On the hardware level, glitches further make the successive gates' leakages mutually dependent on their respective inputs [56,57,58]. On the software level, CPU synchronization limits, but does not eliminate, the issue of glitches. These issues can be avoided by adding registers and controlling transitions [43,27] in hardware, and by trying to avoid transitions using assembly programming tricks in software [44,20,17]. However, these techniques still rely on abstract models for the leakage, and current techniques in the literature test the data isolation assumption only indirectly, by estimating the statistical security order of an implementation [7,71].

As for the second assumption (noise independence), the noise in the side channel leakage of a device is multivariate, and the noises occurring during successive operations likely include some dependency. This assumption is only currently studied at a high level in a few works [48,31].

Contributions. Our contributions can be summarized as follows.

- First, we present a complete methodology to transform abstract circuits into physical implementations secure against side-channel attacks. For this purpose, we rely on a random probing compiler and discuss the concrete steps to use the reduction from the noisy leakage model to the random probing model on physical implementations. While this reduction is well-studied in theory, our methodology summarizes all hypotheses that must be met in practice and identifies technical challenges that must be overcome to achieve formally secure circuits on physical devices.
- Then, we propose new tools to solve these technical challenges.
 - We explain how to enforce the data isolation assumption and introduce a novel practical test for its validation on a physical implementation.

Our test offers a direct approach, contrasting with existing methods in the literature. We conduct experiments on a real target, an STM32F3 MCU, using NewAE's ChipWhisperer-Lite CW1173 board. While our test does not provide a formal proof for the assumption, it stands as the first literature instance of directly addressing and validating this hypothesis with a practical, dedicated procedure.

- We offer a method to integrate the noise independence assumption into the analysis, making it possible to quantify the loss of security implied by a lack of independence. We specifically discuss a relaxation of the assumption aiming to split the noise occurring during the execution of the algorithm into independent noises on each of the operations. We first show a trivial way of doing the split and then express it as a constrained optimization problem that better scales with the size of the circuit. We propose a direct non-optimal solution to the problem and leave the question of optimally and efficiently solving it as an open problem.
- Finally, we highlight the design goals that this security reduction involves. We also exhibit the remaining limitations and open problems of the practical usability of the leakage models. Our goal is to show that it is possible to bridge theory and practice and to motivate further research on remaining issues to fully close the gap, that is to get practical implementations proven secure against side-channel attacks on a physical device without any ideal assumption about the leakage. For instance, one could quantify the impact of a lack of signal independence on security or find an optimal solution for the noise split relaxation to achieve the best security levels.

The organization of the paper is as follows. In Section 2, we provide background and formalize key assumptions. Section 3 outlines our methodology with some details deferred to later sections. In Section 4, we prove our main result: our methodology outputs practically secure implementations against side-channel attacks. Section 5 details our procedures to relax or enforce the identified assumptions. Finally, we conclude with discussions and future directions in Section 6. We also exhibit an implementation of the different steps of our proposed methodology in Supplementary Material (B), on a real target, an STM32F3 MCU, using NewAE's ChipWhisperer-Lite CW1173 board.

2 Technical Background

2.1 Notations

We denote by \mathcal{V} a finite set called the variable space and by \mathcal{X} the input space for the leakage. We denote by \mathcal{Y} the leakage distribution. We use capital letters to denote random variables over a set or a distribution, *e.g.*, X denotes a random variable over \mathcal{X} , and $Y(\boldsymbol{x})$ denotes a random variable (or equivalently a leakage function) over the distribution \mathcal{Y} , taking as input \boldsymbol{x} , a value over the input space \mathcal{X} . We denote by \boldsymbol{y} a leakage trace, *i.e.* a realization of $Y(\boldsymbol{x})$. Any two probability distributions D_1 and D_2 are said ε -close, denoted $D_1 \approx_{\varepsilon} D_2$, if their statistical distance is upper bounded by ε , that is $\mathsf{SD}(D_1; D_2) := \frac{1}{2} \sum_x |p_{D_1}(x) - p_{D_2}(x)| \le \varepsilon$, where $p_{D_1}(\cdot)$ and $p_{D_1}(\cdot)$ denote the probability mass functions of D_1 and D_2 .

2.2 Abstract Circuits

Definition 1 (Abstract Circuit Family). An abstract circuit family is a pair $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ such that

- $-\mathcal{V}$ is the variable space,
- \mathcal{G} called the gate family is a set of functions. For each function $g \in \mathcal{G}$, there exists $\ell, m \in \mathbb{N}$ such that $q: \mathcal{V}^{\ell} \to \mathcal{V}^{m}$.

An abstract circuit C belonging to the family $\mathbb{C} = (\mathcal{V}, \mathcal{G})$, which is written $C \in \mathbb{C}$, is defined as an acyclic directed graph whose edges are wires carrying values over \mathcal{V} , and vertices are gates processing operations over \mathcal{V} . It is further formally composed of input gates of fan-in 0 and fan-out 1 and output gates of fan-in 1 and fan-out 0. Evaluating an ℓ -input *m*-output circuit C consists in writing an input $\mathbf{x} \in \mathcal{V}^{\ell}$ in the ℓ input gates, processing the gates from input gates to output gates, then reading the output $\mathbf{z} \in \mathcal{V}^m$ from the *m* output gates. This is denoted by $\mathbf{z} = C(\mathbf{x})$. During the evaluation process, each wire in the circuit is assigned with a value on \mathcal{V} . We call the tuple of all these wire values a wire assignment of C (on input \mathbf{x}).

The definition of a circuit compiler (CC, Enc, Dec) which turns an abstract circuit into a randomized circuit is recalled in Supplementary material (A).

2.3 Random-Probing Model

Let $p \in [0, 1]$ be some constant leakage probability parameter, usually called *leakage rate*. The random probing leakage can be defined in two ways depending on whether we consider leakage on the wires or the gates of an abstract circuit C from a family $\mathbb{C} = (\mathcal{V}, \mathcal{G})$.

In the *wire leakage* setting, the *p*-random probing model states that during the evaluation of a circuit C, each wire leaks its value with probability p (and leaks nothing otherwise), where all the wire leakage events are mutually independent. To formally define this leakage, we consider two probabilistic algorithms:

- The *leaking-wires sampler* takes as input an abstract circuit C and a probability $p \in [0, 1]$, and outputs a set W, denoted as

$$W \leftarrow \text{LeakingWires}(C, p)$$
,

where W is constructed by including each wire label from the circuit C with probability p to W (where all the probabilities are mutually independent).

- The assign-wires sampler takes as input an abstract circuit C, a set of wire labels W (subset of the wire labels of C), and an input $\boldsymbol{x} \in \mathcal{V}^{\ell}$, and it outputs a |W|-tuple $\boldsymbol{w} \in \mathcal{V}^{|W|}$, denoted as

$$\boldsymbol{w} \leftarrow \mathsf{AssignWires}(C, W, \boldsymbol{x}) \;,$$

where \boldsymbol{w} corresponds to the assignments of the wires of C with label in W for an evaluation on input \boldsymbol{x} .

By convention, we do not consider leakage on the output wires (*i.e.* input wires of the output gate) of a circuit, since when composing several circuits, these wires become input wires to the next circuit.

We can analogously define the *gate leakage* setting with similar probabilistic procedures LeakingGates and AssignGates. For the sake of completeness, a detailed description is given in Supplementary material (A).

Based on these notions, we now formally define the (wire or gate) random probing leakage of a circuit.

Definition 2 (Random Probing Leakage). The *p*-random probing wire leakage of an abstract circuit C with ℓ inputs, on input $\boldsymbol{x} \in \mathcal{V}^{\ell}$ is the distribution $\mathcal{L}_{p}^{wire}(C, \boldsymbol{x})$ obtained by composing the leaking-wires and assign-wires samplers as

$$\mathcal{L}_p^{wire}(C, \boldsymbol{x}) \stackrel{id}{=} \mathsf{AssignWires}(C, \mathsf{LeakingWires}(C, p), \boldsymbol{x})$$
.

For the p-random probing gate leakage, $\mathcal{L}_p^{gate}(C, \boldsymbol{x})$ is obtained as

$$\mathcal{L}_{p}^{gate}(C, \boldsymbol{x}) \stackrel{ia}{=} \mathsf{AssignGates}(C, \mathsf{LeakingGates}(C, p), \boldsymbol{x})$$

We can define the random probing security of an abstract circuit C.

Definition 3 (Random Probing Security). An abstract circuit C with ℓ inputs, from a family of circuits $\mathbb{C} = (\mathcal{V}, \mathcal{G})$, is (p, ε) -random probing secure (RPS) in the wire leakage setting with respect to encoding Enc if there exists a simulator Sim such that for every $\mathbf{x} \in \mathcal{V}^{\ell}$:

$$\operatorname{Sim}(C) \approx_{\varepsilon} \mathcal{L}_{n}^{wire}(C, \operatorname{Enc}(\boldsymbol{x})) .$$
(1)

A circuit compiler (CC, Enc, Dec) is (p, ε) -random probing secure in the wire leakage setting if for every circuit C the compiled circuit $\widehat{C} = CC(C)$ is $(p, |C| \cdot \varepsilon)$ random probing secure in the wire leakage setting where |C| is the size of the original circuit.

We equivalently define (p, ε) -random probing security for a circuit and a circuit compiler in the gate leakage setting, where we use \mathcal{L}_p^{gate} instead of \mathcal{L}_p^{wire} .

We have the following reduction of security, which states that if a circuit is random probing secure in the wire leakage setting, then it is secure in the gate leakage setting. Our proof is given in Supplementary material (A).

Lemma 1. Let C be an abstract circuit with ℓ inputs from a family of circuits $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ such that each gate $g \in \mathcal{G}$ has at most two input wires. If C is (p, ε) -random probing secure with respect to encoding Enc in the wire leakage setting, then C is (p', ε') -random probing secure in the gate leakage setting, with $p' = p^2$ and $\varepsilon' = \varepsilon$.

2.4 Noisy Leakage Model

The noisy leakage model was formalized in [66]. In this model, a leaking computation is modeled by a sequence of elementary operations $(g_i)_i$ accessing a common memory called *internal state*. Each elementary operation reads its input and writes its output on the internal state. When processed on some input \boldsymbol{x} , an elementary operation g_i reveals $f_i(\boldsymbol{x})$ to the adversary for some noisy leakage function f_i . A noisy leakage function takes two arguments: the value \boldsymbol{x} held by the accessed part of the internal state (data isolation assumption) and a random string ρ long enough to model the leakage noise. Each execution leaks the values $(f_i(\boldsymbol{x}_i, \rho_i))_i$ where the \boldsymbol{x}_i 's are the successive intermediate values (from the internal state) in input of the elementary operations g_i 's and ρ_i 's are fresh random strings. We stress that all the ρ_i 's involved in successive executions are uniformly and independently drawn (independent noise assumption).

We note that from a formal point of view, there is an equivalence between the circuit model used by the gate-leakage random probing model and the internal state model used by the noisy leakage model. In both cases the computation is divided into sub-computations (either gates or elementary operations) and the full leakage is composed of the outputs of leakage functions (either random probing functions or noisy functions) applied to all the sub-computations input. The internal state model has the advantage of being cosmetically closer to a real software implementation, moreover it is useful to consider the order of operations while relaxing the data isolation and noise independence assumptions (as discussed later).

For the sake of simplicity, we shall omit the random string parameter, which leads to the notation $f_i(\mathbf{x})$ where \mathbf{x} is the accessed value. Note that $f_i(\mathbf{x})$ can be seen as the output of a probabilistic algorithm. In particular, $f_i(\mathbf{x})$ can take several values with a given probability distribution, and can therefore be considered as a random variable. The noisy property of f is captured by assuming that the bias introduced in the distribution of a uniform random variable X given the leakage f(X) is bounded. This is formalized in the next definition:

Definition 4 (Noisy Function). Let \mathcal{X} be a finite set and let $\delta \in \mathbb{R}$. A δ -noisy leakage function f on \mathcal{X} is a function of domain $\mathcal{X} \times \{0,1\}^{|\rho|}$ for some $|\rho| \in \mathbb{N}$ such that

$$\beta(X|Y) := \sum_{y \in \text{Range}(f)} \Pr(Y = y) \cdot \Delta((X \mid Y = y); X) \le \delta , \qquad (2)$$

where Δ is a statistical distance measure, X is a uniform random variable over \mathcal{X} and where Y = f(X, R) for a uniform random variable R over $\{0, 1\}^{|\rho|}$.

The above definition depends on the notion of statistical distance. In the original definition from [66], the authors use the L_2 norm. The authors of [38] then suggested to use the L_1 norm (normalized by $\frac{1}{2}$). It was later suggested in [65] to use a statistical distance notion based on the *relative error*. Noisy functions based on this distance are referred to as *average relative error* (ARE) noisy

leakage functions in [65] since the relative error is averaged over the distribution of the leakage Y in Equation 2.

As recalled hereafter, the noisy leakage metrics based on the L_1 statistical distance (SD) and the ARE enjoy useful security reductions to the random probing model. We recall the definition of these two metrics based on the *pointwise* mutual information.

Definition 5 (Pointwise Mutual Information). Let X, Y be random variables over \mathcal{X}, \mathcal{Y} respectively. For any $x \in \mathcal{X}, y \in \mathcal{Y}$, the exponential form of the pointwise mutual information (PMI) is defined as:

$$PMI_{X,Y}(x,y) = \frac{P[X = x, Y = y]}{P[X = x] \cdot P[Y = y]} - 1 .$$

Definition 6. Let X, Y be random variables over \mathcal{X}, \mathcal{Y} respectively. We can define the L1 statistical distance (SD) as follows:

$$\operatorname{SD}(X|Y) = \frac{1}{2} \mathbb{E}_{Y=y} \mathbb{E}_{X=x} [|\operatorname{PMI}_{X,Y}(x,y)|].$$

The average relative error (ARE) can also be expressed as:

$$\operatorname{ARE}(X|Y) = \mathbb{E}_{Y=y} \left[\max_{x} |\operatorname{PMI}_{X,Y}(x,y)| \right].$$

From random probing to noisy leakage security. In [38], Duc, Dziembowski, and Faust show the following security reduction: any circuit which is (p, ε) -secure in the random probing model is also (δ, ε) -secure in the noisy leakage model (for the same parameter ε) defined w.r.t. the metric $\beta(X|Y) = \text{SD}(X|Y)$ and for any $\delta \leq p/|\mathcal{X}|$, where \mathcal{X} is the input space of the abstract gates / elementary operations.⁷ This result was later extended to the noisy leakage model defined w.r.t. the metric $\beta(X|Y) = \text{ARE}(X|Y)$ in the work of Prest et al. [65]. Those security reductions directly hold from the following key lemma.

Lemma 2 ([38,65]). Let $\phi_p : \mathcal{X} \to \mathcal{X} \cup \{\bot\}$ the randomized function defined for every $p \in [0,1]$ as

$$\phi_p(x) = \begin{cases} \bot & \text{with probability } 1 - p \\ x & \text{with probability } p \end{cases}$$
(3)

Let $f : \mathcal{X} \to \mathcal{Y}$ be a δ -noisy leakage function (w.r.t. SD or ARE). There exists a randomized function $f' : \mathcal{X} \cup \{\bot\} \to \mathcal{Y}$ such that for every $x \in \mathcal{X}$ we have

$$f(x) = f'(\phi_p(x)) \quad with \quad \begin{cases} p \le \delta \cdot |\mathcal{X}| & \text{if } \operatorname{SD}(X|f(X)) \le \delta \\ p \le \delta & \text{if } \operatorname{ARE}(X|f(X)) \le \delta \end{cases}$$
(4)

⁷ The input space \mathcal{X} is different than the variable space \mathcal{V} for the variables in a circuit. Typically, when the leakage is defined on the internal state of the gate, the latter can be described by both its input wires, and hence the input space is $\mathcal{X} = \mathcal{V}^2$.

We recall that the ARE is a worst-case metric, contrary to the SD, which is an average-case metric. This explains the tighter reduction (*i.e.* no loss induced by the size of the input space) using the ARE from the noisy model to the random probing model since the latter is also a worst-case model.

Besides this worst-case vs. average-case question, we note that the SD and ARE can be connected to metrics that are used in practice to evaluate the security of a leaking implementation. For example, the SD can be expressed using Mutual Information (MI) thanks to [37] and the Mutual Information can (under some conditions) be expressed using the Signal-to-Noise Ratio (SNR) [54] and the correlation coefficient [19] thanks to [55]. The MI is a standard metric to analyze multivariate leakages while the SNR and correlation coefficient are among the most popular tools for univariate security assessments.

In the following, we shall refer to the reduction from [38] using the SD metric as the *DDF reduction*, and to that from [65] using the ARE metric as the *PGMP reduction*.

2.5 Physical Assumptions

The noisy leakage model has been argued to capture well power and electromagnetic leakages. In all generality, an elementary operation processing a value \boldsymbol{x} gives rise to a leakage trace $Y(\boldsymbol{x})$ which is a multivariate random variable (a.k.a random vector) following a distribution whose parameters depend on \boldsymbol{x} . In most practical contexts, this distribution is well approximated by a multivariate Gaussian $\mathcal{N}(\boldsymbol{m}_x, \boldsymbol{\Sigma})$ for some parameters \boldsymbol{m}_x (mean vector) and $\boldsymbol{\Sigma}$ (covariance matrix), see *e.g.* [30,69]. Such parameters can be inferred in practice through a profiling of the device, from which we obtain the noisy leakage metric δ by evaluating Equation 2.

We still need to stress that, as is, the noisy leakage model relies on two assumptions about the underlying physical device which might not be verified in practice without further care.

Assumption 1 (Data isolation) A leakage function f_i corresponding to the elementary operation $g_i(\mathbf{x}_i)$ only depends on the current state \mathbf{x}_i and not on previously accessed parts of the state: $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \ldots$ The leakage is then assumed to respect some data isolation between successive elementary operations.

However, as mentioned in the introduction, physical effects such as glitches and transitions will likely break this implicit assumption. Hence, one should take special care and enforce data isolation for the model to be valid.

Assumption 2 (Noise independence) The leakage noises from the successive elementary operations are independent from each other. Formally, the random tape ρ_i in each $f_i(\mathbf{x}_i, \rho_i)$ is sampled as a fresh uniform string.

In practice, this assumption does not easily hold: if one cuts a leakage trace into several sub-traces corresponding to successive elementary operations, the noises in the successive sub-traces would likely include some part of dependency. Indeed, a correlation exists between successive leakage points, which makes multivariate statistics particularly useful for side-channel attacks [30].

In the following, we shall refer to the original noisy leakage model, which relies on the two aforementioned assumptions as the *idealized noisy leakage model*. We will explore how to relax or enforce those physical assumptions to reduce the security of a physical implementation to that of an abstract implementation in the idealized noisy leakage model, which subsequently reduces to the random probing security.

3 Methodology

The theoretical community introduced many constructions proven secure in the (random) probing and noisy leakage models with a quantified security level. Meanwhile, it is unclear how to implement such constructions on physical devices while preserving the proven security. Indeed, the existing literature does not explain all the steps nor states all the hypotheses required for preserving proven security claims in practice. In this section, we rigorously exhibit all the steps to turn an abstract circuit into a physical implementation satisfying provable security against side-channel attacks⁸. As illustrated in Figure 1, these steps can be split into two phases:

- (i) a characterization phase which only depends on the device and the sidechannel acquisition tool (*i.e.*, without any knowledge of the abstract circuit). It includes the implementation of specific gates (Step 1), the analysis of Assumption 1 (data isolation) to exhibit a relevant whitening procedure (Step 2), the characterization of the leakage (Step 3), the enforcement and relaxation of Assumption 2 (noise independence) (Step 4), and the estimation of the noisy leakage parameter (Step 5),
- (ii) a compilation phase using the outputs of the characterization phase to turn an abstract circuit $C \in \mathbb{C}$, with $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ an abstract circuit family, into a practically secure implementation for a given security level λ (Step 6). This phase relies on the usage of a secure random probing compiler.

All the intermediate steps are described at a high level in the next sub-sections. In Section 3.7, we briefly describe the experiments that we did to validate our methodology, which are detailed in Supplementary Materiel (B). Finally, some dedicated procedures are further developed in Sections 5.1 and 5.2.

3.1 Step 1: Implementing Abstract Gates

The first step of our methodology consists in implementing abstract gates as software routines. A physical elementary operation abstracted as a gate by the

⁸ We describe our methodology in the context a of software implementation, where elementary calculations align with software routines. We discuss a generalization to the case of hardware implementations in Section 6.



Fig. 1: Illustration of our methodology

noisy leakage model (see Section 2.4) first looks up its operands from memory (the computation state), then executes a sequence of arithmetic instructions (implementing the gate functionality $g \in \mathcal{G}$), and finally writes back the result to memory. This process generates some side-channel leakage depending on the executed instructions and the processed data, which is the leakage of the physical elementary operation abstracted by the noisy leakage model. A developer must first translate this behavior into a software routine on a physical device. We propose to implement such a routine in ARM assembly as follows (with the xor operation as an example):

```
operation_xor:
  ldr r0, [r0]
  ldr r1, [r1]
  eor r0, r1, r0 // For other operations, change instruction
  str r0, [r2]
```

with the following C signature:

We define a routine for each abstract gate $g \in \mathcal{G}$ which, when executed on the target device, behaves as a physical elementary operation abstracted by the noisy model. From these implementations of the abstract gates, any circuit $C \in \mathbb{C}$ can be compiled into a physical implementation on the target device. This implementation takes the form of a sequence of calls to the elementary operations, looking like the following C-syntax example:

```
operation1(a1Ptr, b1Ptr, c1Ptr);
operation2(a2Ptr, b2Ptr, c2Ptr);
...
```

The routines operation1, operation2, ... are all among the implemented gate routines which are mapped from the gates of the circuit. The pointer arguments

(a1Ptr, b1Ptr, c1Ptr), (a2Ptr, b2Ptr, c2Ptr), \ldots are constant addresses triplets which encode the data dependency of the implementation, i.e., the wires in the abstract circuit.⁹

3.2 Step 2: Enforcing / Relaxing Data Isolation

Once the syntax of elementary operations is fixed, the physical assumptions made in the noisy leakage model must be satisfied by the implementations in order to use the security reduction. Our methodology first focuses on the data isolation assumption (*i.e.*, Assumption 1), which requires that the leakage of an elementary operation only depends on its inputs, *i.e.*, is independent of the inputs of the previous and the following operations. This assumption rarely holds in practice, since elementary operations executed successively might leak jointly on their manipulated data. Indeed, after the execution of an elementary operation, the data it has processed might be stored in the physical state of the CPU. The leakage of the following elementary operation will then be a (probabilistic) function of the data it processes and of the physical state of the CPU, hence of the previously processed data. This is a well-known issue in the side-channel literature. In particular, this data non-isolation includes the so-called transition leakage observed and analyzed in many works [62,59,63]. These pitfalls have a direct practical impact, typically leading to losing security orders in the masking scheme [6]. In the provable security setting, this translates to breaking the data isolation assumption: assuming that each elementary operation leaks a (probabilistic) function of the accessed part of the state is incorrect: the leakage also depends on the state's previously accessed part(s). Hence, a developer can not simply implement a circuit as a sequence of the routines introduced in Section 3.1, as the side-channel security can no longer be reduced to the random probing model.

As data isolation plays a crucial role in upholding security proofs and stands as a critical step in our methodology, we introduce a method to enforce it, inspired by prior works (*e.g.*, [20,28]). Additionally, we design a dedicated test to validate data isolation on a target device.

Enforcing data isolation. We use data whitening to enforce the data isolation assumption in our methodology. The principle is to call a routine on constant or random data whose sole purpose is to clean the CPU state from any dependency on the previously processed data. Specifically, after each call to an elementary operation routine, we insert one or more calls for which the arguments point to random or constant data in memory. The intuition is that by relying on a call to a similar elementary operation routine, we expect to clean the data path, namely to write random or constant data in any hardware register containing data-dependent information from the previous call. Nevertheless, although natural, this solution might not suffice to ensure data isolation on some devices.

⁹ The proposed implementation style is admittedly not very efficient. This paper mainly targets security and simplicity, leaving optimization to future works.

The effectiveness of a whitening routine depends on the microarchitecture of the device's CPU. Therefore, a developer might have to empirically test several approaches before reaching successful and efficient isolation.

Even with an isolation that avoids all transition and glitches effects across operations, it might not be possible to partition the leakage trace in time intervals whose leakage corresponds to only a single operation. Indeed, the leakage is often subject to low-pass filtering inside the target chip or the measurement chain. As a result, the independent intrinsic leakage of many operations will be linearly combined in the measured trace. We propose to relax the noisy leakage model to allow the leakage to be composed of linear combinations of independent noisy leakage functions. In this case, we aim at ensuring that a leakage function f_i corresponding to the elementary operation $g_i(\mathbf{x}_i)$ does not jointly depend on the current state \mathbf{x}_i and previously accessed parts of the state: $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots$ In other words, f_i depends on the current state and has at most linear dependencies on the previous parts $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots$ With this relaxation, we still provide independence between the inputs of the different operations, *i.e.*, data isolation.

Testing data isolation. In Section 5.1, we introduce a novel way to test the effectiveness of a data whitening routine. The idea is to suppose that the leakage distribution can be modeled as a sum of a deterministic function of the first operation's inputs, a deterministic function of the second operation's inputs, and some noise value. In other words, we test that the leakage can be decomposed in additive parts that to not jointly depend on the inputs of both operations.

3.3 Step 3: Characterizing the Leakage

Once data isolation is enforced and tested, one can safely infer the leakage distribution of each physical elementary operation. This is a classical problem in the side-channel literature, and we can rely on a solid theoretical and practical ground for this step. We rely on the common assumption [32,69] that the leakage distribution \mathcal{Y} of an elementary operation with inputs $\boldsymbol{x} \in \mathcal{V}^{\ell}$ takes the form of a deterministic function of \boldsymbol{x} plus an additive Gaussian noise:

$$\mathcal{Y}_{\boldsymbol{x}} = d(\boldsymbol{x}) + \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) , \qquad (5)$$

where the deterministic part of the leakage can be written as a linear combination of a predetermined basis of functions $\mathcal{H} = \{h_1, \ldots, h_m\}$, i.e.:

$$d(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i h_i(\boldsymbol{x}) .$$
 (6)

The choice of the basis of functions \mathcal{H} is determined for each elementary operation routine depending on its internal variables. The basis should at least contain one function for each internal variable bit but might also include monomials of higher degrees due to possible coupling effects [43].

In our methodology, we suggest relying on linear regression in order to estimate the deterministic leakage $d(\cdot)$. It involves acquiring an initial set of ℓ_1 traces, which measure the leakage while executing the operation on ℓ_1 inputs generated uniformly at random. We then use this set to infer the coefficients $\{\alpha_i\}_{i=1,\ldots,m}$. Subsequently, we can compute the covariance matrix using a new set of ℓ_2 traces on uniform random inputs, allowing us to recover (an estimation of) the covariance matrix Σ .

It's worth emphasizing that while we propose linear regression for leakage estimation, our methodology remains adaptable to other estimation methods. Techniques such as template attacks [30], combined with dimensionality reduction [74,25], and the emerging use of machine learning for side-channel analysis [60,64], all offer viable alternatives.

3.4 Step 4: Enforcing / Relaxing Noise Independence

Next, we consider the noise independence assumption (Assumption 2) needed for the reduction from the noisy leakage to the random probing model. Namely, in the idealized noisy leakage model, the noise that occurs during the execution of an elementary operation is drawn independently of the noise that occurs during the execution of the previous ones. Hence, this assumption must be satisfied in practice. Meanwhile, it is hard to enforce and test since no clear separation of the noise occurs during a leakage trace. In our methodology, we propose a novel way to relax this assumption. Namely, we keep the Gaussianity hypothesis, but we allow the leakage of the different operations to overlap. We characterize this relaxation and directly reflect it on the security level by providing a reduction from the noisy leakage model with potential noise dependence to the idealized noisy leakage model.

In our methodology, we propose relaxing the assumption of noise independence by partitioning the noise distribution into multiple distributions, all while minimizing leakage during each operation. In simpler terms, given k consecutive elementary operations of inputs $\{(\boldsymbol{x}_i)\}_{1 \leq i \leq k}$, we can represent the overall leakage distribution as

$$\mathcal{Y} = \sum_{i=1}^{k} d_i(\boldsymbol{x}_i) + \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$$
(7)

where $d_i(\boldsymbol{x}_i)$ are the different deterministic signals of the operations, and the noise drawn from $\mathcal{N}(\mathbf{0}, \Sigma)$ is the global noise. Thanks to the data isolation enforcement and test from Section 3.2, the deterministic signals are mutually data independent. Specifically, while the d_i 's might overlap on some time samples, they independently apply to the inputs \boldsymbol{x}_i . This ensures that the global deterministic leakage can be expressed as a sum in Equation 7.

In order to relax the noise independence, our approach consists in finding a set of covariance matrices $\{\Sigma_i\}_{i\in[k]}$ such that $\sum_i \Sigma_i = \Sigma$. This way, we can split the Gaussian noise distribution $\mathcal{N}(\mathbf{0}, \Sigma)$ into k independent Gaussian distributions $\mathcal{N}(\mathbf{0}, \Sigma_1), \ldots, \mathcal{N}(\mathbf{0}, \Sigma_k)$. This representation enables us to split the leakage distribution into several functions $\mathcal{Y}_i = d_i(\mathbf{x}_i) + \mathcal{N}(\mathbf{0}, \Sigma_i)$ for every $i \in \{1, \ldots, k\}$.

An adversary given a leakage sample of each \mathcal{Y}_i is more powerful than an adversary given a sample of the global leakage \mathcal{Y} because the former can always sum the \mathcal{Y}_i samples to get a \mathcal{Y} sample.

Instantiation. Section 5.2 introduces several methods to define such matrices Σ_i , which sum to the original covariance matrix Σ while minimizing the mutual information between the leakage and the signals.

3.5 Step 5: Estimating the Noisy Leakage Parameter

Recall that once data isolation is enforced and tested as described in Section 3.2, we can empirically characterize the leakage distribution of each elementary operation as described in Section 3.3. Assuming that the leakage takes the form of Equation 5, we compute the coefficients in the deterministic function for each elementary operation (as defined in Equation 6). Then, we can apply the noise relaxation described in Section 3.4 to get the covariance matrix of the Gaussian noise, which we suppose is the same for each elementary operation, following the representation in our optimization problem.

We can then compute the noisy leakage parameter δ related to the δ -noisy leakage model. As explained in Section 2.4, reducing the *idealized* noisy leakage model to the random probing model provides the leakage probability in the latter. More precisely, in order to achieve (δ, ε) -security in the idealized noisy leakage model, an abstract circuit should achieve (p, ε) -security in the random probing model with $p = \gamma \cdot \delta$ for some constant factor γ depending on the noisy leakage metric ($\gamma = |\mathcal{X}|$ for the SD metric, $\gamma = 1$ for the ARE metric). The factor γ depends on the chosen noisy leakage metric. There are different options available. The original security reduction [38] (DDF reduction) relies on the statistical distance between a (uniform) variable X and the same variable conditioned on its leakage Y: $\delta = E_y[SD((X|Y = y); X)]$. When reducing to the random probing model, this value is multiplied by the size of the input space \mathcal{X} (*i.e.*, the definition set of inputs x of an elementary operation), hence losing tightness in the tolerated leakage rate through the reduction. In a more recent work [65] (PGMP reduction), the authors express δ using different noisiness metrics from the pointwise mutual information. The most interesting metric is the average relative error (ARE), a worst-case metric (just like the random probing model), contrary to the statistical distance, which is an average-case metric. When computing $\delta = ARE(X; X|Y)$, the reduction to the random probing model thus yields tighter results with $p = \delta$.

In order to estimate the noisy leakage parameter in our methodology, we compute both ARE and SD metrics using the inferred leakage model to compare both reductions to the random probing model. Using the pointwise mutual information (Definition 6), we have

$$ARE = \mathbb{E}_{Y} \max_{X=\boldsymbol{x}} \left| \frac{P[X=\boldsymbol{x}, Y=\boldsymbol{y}]}{P[X=\boldsymbol{x}] \cdot P[Y=\boldsymbol{y}]} - 1 \right|$$

$$= \mathbb{E}_{Y} \max_{X=\boldsymbol{x}} \left| \frac{P[Y=\boldsymbol{y}|X=\boldsymbol{x}]}{\sum_{X=\boldsymbol{x}'} P[Y=\boldsymbol{y}|X=\boldsymbol{x}']} \cdot \frac{1}{P[X=\boldsymbol{x}]} - 1 \right|$$
(8)

and

$$SD = \mathbb{E}_{Y} \frac{1}{2} \sum_{X=\boldsymbol{x}} \left| \frac{P[Y = \boldsymbol{y} | X = \boldsymbol{x}]}{\sum_{X=\boldsymbol{x}'} P[Y = \boldsymbol{y} | X = \boldsymbol{x}']} \cdot \frac{1}{P[X = \boldsymbol{x}]} - 1 \right|$$
(9)

From the equations above, we need to compute the sum of the conditional probabilities P[Y = y|X = x'] for $x' \in \mathcal{X}$ for ARE and SD estimations. Then, in the case of ARE estimation, we need to find the maximal value of the expression given between $|\cdot|$ in Equation 8 over the values taken by X. While in the case of the SD estimation, we compute a sum over the expression given between $|\cdot|$ in Equation 9. Finally, we must compute the expected value for ARE and SD for Y.

Computing the conditional distribution. In order to estimate the conditional distribution P[Y = y|X = x] given a leakage trace and x, we can use the leakage characterization computed earlier, since the conditional distribution is known to be expressed as

$$P[Y = \boldsymbol{y}|X = \boldsymbol{x}] = \frac{1}{\sqrt{(2\pi)^n \left|\boldsymbol{\Sigma}\right|}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - d(\boldsymbol{x}))^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{y} - d(\boldsymbol{x}))\right) \quad (10)$$

where n is the number of samples in y [30].

Estimating the expected value. Each sample point in the leakage distribution \mathcal{Y} is a continuous random variable over \mathbb{R} . Hence, the expected value \mathbb{E}_Y is computed as an integral. Instead of computing the integral, we use a Monte Carlo integration method to estimate the expected value. Namely, we draw several random leakage values to estimate the expected value. The ARE is then computed as

$$ARE = \frac{1}{k} \sum_{Y=y} \max_{X=x} \left| \frac{P[Y=y|X=x]}{\sum_{X=x'} P[Y=y|X=x']} \cdot \frac{1}{P[X=x]} - 1 \right|, \quad (11)$$

and the SD is computed as

$$SD = \frac{1}{2 \cdot k} \sum_{Y=y} \sum_{X=x} \left| \frac{P[Y=y|X=x]}{\sum_{X=x'} P[Y=y|X=x']} \cdot \frac{1}{P[X=x]} - 1 \right|.$$
(12)

In both equations, k denotes the number of leakage vectors \boldsymbol{y} drawn from the leakage distribution \mathcal{Y} . Each leakage vector \boldsymbol{y} is generated as $\boldsymbol{y} = d(\boldsymbol{x}) + \boldsymbol{\varphi}$, where \boldsymbol{x} is an input generated uniformly at random, $d(\cdot)$ is the deterministic function for an elementary operation, and $\boldsymbol{\varphi}$ is generated from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \Sigma)$ following the noise covariance Σ .

Thanks to the law of large numbers, we know that as k approaches infinity, this estimation converges to the expected value \mathbb{E}_Y . We show in Supplementary Material (B) in our experimental results that the convergence curve reaches a plateau after some number of samples, which corresponds to the value of \mathbb{E}_Y .

3.6 Step 6: Compiling the Cryptographic Implementation

At this stage, we have estimated the leakage parameter of each isolated noiseindependent elementary operation. As defined in Section 3.5, this parameter can be computed using the SD or ARE metric. We obtain an equivalent leakage probability p in the random probing model in both cases by applying the reduction. The reduction is tighter in the case of ARE, where the same leakage parameter is the leakage probability in the random probing model.

Different vs. maximum leakage probabilities. As we might obtain different noisy leakage parameters δ_i for the different elementary operations, leading to different leakage probabilities in the random probing model, we consider that all the gates leak with the maximum probability corresponding to maximum noisy metric $\delta = \max_i \delta_i$. A random probing secure circuit with maximum leakage probability is straightforwardly random probing secure with different leakage probabilities.

Gate vs. wire leakage model. In our characterization, we rely on the leakage of the elementary operations abstracted as gates in a circuit. Meanwhile, most random probing secure constructions suppose leakage on wires instead. Our methodology applies an additional transition from the gate to the wire leakage model to circumvent this issue. As proved in Lemma 1, we can reduce the security of a circuit in the gate random probing model with leakage probability p, to the wire random probing model with leakage probability \sqrt{p} , assuming that each gate has at most two inputs.

Compiling a cryptographic implementation. The final step consists of a two-stage compilation process applied to the input abstract circuit $C \in \mathbb{C}$ representing the target cryptographic implementation.

- One first applies the random probing secure compiler which, for the obtained leakage probability p and the target security level $\varepsilon = 2^{-\lambda}$, transforms C into a functionally equivalent randomized circuit \widehat{C} achieving (p, ε) -random probing security.
- One then serializes \widehat{C} into a physical implementation, making a sequence of calls to the (whitened) elementary operation routines on the target device. Each elementary operation in the sequence corresponds to a gate in \widehat{C} whose output is written in a fresh memory cell. The circuit wiring is hardcoded in the pointer arguments passed to the successive calls to the elementary operation routines.

The obtained physical implementation achieves λ bits of side-channel security for the target side-channel acquisition tool and physical device, and under relaxed or empirically verified physical assumptions. The overall process and provable security guaranty are wrapped up in Section 4.

Experimental Validation 3.7

In order to validate our methodology, we conducted tests on an STM32F3 MCU using NewAE's ChipWhisperer-Lite CW1173 board, which is based on an ARM Cortex-M4 processor. Detailed experimental validation is provided in Supplementary Material (B). This includes a description of how we implemented elementary operations, our whitening approach, and the application of our data isolation test, later discussed in Section 5.1, along with practical results. Furthermore, we conducted a leakage characterization for the operations, computed the device's noise covariance, and estimated the noisy leakage parameters. Our findings revealed low noise levels on the device, rendering it unsuitable for masking purposes, thus highlighting the need for secure hardware offering sufficient noise levels to attain robust provable security. Finally, we showcase an application of our methodology on a masked AES implementation.

$\mathbf{4}$ Security Proof

After describing our methodology, we aim to prove our main result.

Theorem 1. Let C be an abstract circuit in the abstract circuit family \mathbb{C} with $\mathbb{C} = (\mathcal{V}, \mathcal{G})$. Let \mathcal{D} be a target device, \mathcal{A} a target side-channel acquisition tool, and λ a security level in bits. At the end of the six-step methodology described in Section 3, the resulting physical implementation achieves $\varepsilon = 2^{-\lambda}$ security against side-channel attacks under the following assumptions:

- The data isolation effectively ensures that the deterministic signal can be expressed as a sum ∑_{i=1}^k d_i(a_i, b_i);
 The leakage characterization yields the exact leakage distribution (i.e., the
- exact deterministic functions $\{d_i\}$ and covariance matrix Σ),

While those ideal assumptions might not be perfectly met in practice, they can be naturally relaxed. We thus assume that an adversary cannot effectively exploit the approximation error between this ideal world and the actual leakage to increase the advantage beyond $\varepsilon = 2^{-\lambda}$.

Proof. Figure 2 gives a specific view of our characterization phase with the leakage assumptions. We assume that each elementary operation has at most two inputs and denote such inputs as (a_i, b_i) . After we implement the abstract gates (Step 1) and enforce and test data isolation (Step 2), we can characterize the leakage for each elementary operation (Step 3). We thus get a global leakage model \mathcal{Y} for any sequence of elementary operations. Next, we can apply the noise-splitting strategy to obtain separated leakages \mathcal{Y}_i with independent noises for the different elementary operations (Step 4). We can then estimate the noisy leakage metrics δ_i of the different elementary operations (Step 5) and apply the DDF/PGMP reduction which yields a leakage probability $p_{\mathsf{GL}} = \gamma \cdot \max_i \delta_i$ in the gate-leakage random probing model. Finally, we get a leakage probability $p_{\mathsf{WL}} = \sqrt{p_{\mathsf{GL}}}$ in the wire-leakage random probing model.



Fig. 2: Illustration of the characterization phase with the leakage assumptions

We can go up the methodology path from Figure 2 to formally prove the security of the compiled physical implementation. Consider \widehat{C} the randomized circuit output by the RPS compiler and which achieves $(p_{\text{WL}}, \varepsilon)$ -security in the wire-leakage random probing model. From the argument given above, we have that \widehat{C} achieves $(p_{\text{GL}}, \varepsilon)$ -security in the gate-leakage random probing model. Then, let us consider the *abstract physical implementation* (API) corresponding to \widehat{C} in the idealized noisy leakage model with operation leakage $\{\mathcal{Y}_i\}$ and corresponding noisy metrics $\{\delta_i\}$. By application of the DDF/PGMP reduction, we get that this API achieves $(\{\delta_i\}, \varepsilon)$ -security in this idealized noisy leakage model. This further translates to the security of the API with global leakage \mathcal{Y} thanks to the noise splitting reduction. The physical implementation, which is an instantiation of the API on the target device, thus achieves $\varepsilon = 2^{-\lambda}$ security against side-channel attacks under the two assumptions expressed in the theorem statement.

5 Dedicated Procedures for Assumptions 1 and 2

Our methodology relies on tests and procedures to enforce or relax Assumptions 1 and 2. In this section, we introduce a test of data isolation (Step 2) given a whitening procedure, and showcase various noise splitting implementations (Step 4). Although some proposals are basic proof-of-concept versions that require further refinement, when considered alongside our contributions in Section 3, they collectively enable the successful completion of our overall methodology.

5.1 Step 2: Testing Data Isolation

Masking security proofs require independence between the leakage of all operations. However, enforcing and testing this independence assumption is challenging, leading to another approach in practice based on the test vector leakage assessment (TVLA) [72]. This approach verifies the statistical security order (*i.e.* the smallest statistical moment that leaks) of a masked implementation by detecting secret-dependencies in the statistical moments of the leakage [41]. While dependence in the moment corresponding to the security order is expected due to dependence in inputs of the leakage functions (e.g., all the shares of avalue), lower-order moments in a threshold-probing secure implementation with independent leakage functions are independent of the secret. This test can indeed detect typical leakage independence violations due to physical defaults like glitches [57,58] or transitions [33,6] when they lead to a security order reduction. Due to the difficulty in enforcing strict independence in the implementation and to verify it, a commonly accepted relaxation is to ensure that if there are detectable lower-order leakages, they are of significantly lower amplitude than those at the target security order [41].

While this heuristic works reasonably well in practice, it has two significant limitations. First, by verifying only a security order, it cannot detect leakage dependence issues that would result in other kinds of weaknesses than security order reductions, *e.g.*, easing horizontal attacks. Second, while this approach is always applicable in theory, it requires testing all the mixed statistical moments corresponding to all the tuples of leakage points in the traces of a masked implementation [8], and it is therefore computationally impractical (it scales exponentially with the length of the trace) at large security orders (even the second order can be challenging).

Therefore, we propose another approach with much improved practical efficiency, which can detect leakage dependencies that do not reduce the security order. Our approach is based on testing the independence between the leakage of two consecutive operations. Then, we use an argument based on physics to extend the result of this test to long sequences of operations. We recall that we provide an implementation of this test in Supplementary material (Section B) on a STM32F3 MCU.

Leakage independence for adjacent operations. Let us consider two operations op_1 and op_2 with their respective inputs x_1 and x_2 . We assume that these two operations are executed sequentially, giving rise to a leakage trace Y.

We say that the operations have independent leakage if

$$Y(x_1, x_2) = d_1(x_1) + d_2(x_2) + N$$
(13)

where d_1 and d_2 are the deterministic functions (like in [70]) and N follows a Gaussian noise distribution \mathcal{N} . This definition indeed ensures independence, as it is possible to decompose N into two independent Gaussian noises N_1 and N_2 , giving $Y = (d_1(\boldsymbol{x}_1) + N_1) + (d_2(\boldsymbol{x}_2) + N_2)$. Despite the sequential execution context, we cannot assume that the leakage is a sequential combination of the leakage of the operations $(e.g., Y = (d_1(\boldsymbol{x}_1) + N_1, d_2(\boldsymbol{x}_2) + N_2))$ due to data dependency effect between successive operations (e.g. transitions in CPU buses/registers) and low-pass filtering in the measured circuit or acquisition chain.

We use the following statistical test to verify that Equation 13 holds:

- 1. Generate uniformly at random two pairs α and β of inputs $(\boldsymbol{x}_1^{\alpha}, \boldsymbol{x}_2^{\alpha})$ and $(\boldsymbol{x}_1^{\beta}, \boldsymbol{x}_2^{\beta})$ (similarly to *fixed-vs-fixed* leakage assessment).
- 2. Acquire a set $T_{(\alpha,0)}$ of ℓ traces corresponding to executing the operations with the inputs set to $(\boldsymbol{x}_1^{\alpha}, \boldsymbol{0})$ and compute the average trace $\overline{T}_{(\alpha, 0)}$.
- 3. Do the same thing for $T_{(\beta,0)}$ with inputs $(\boldsymbol{x}_1^{\beta}, \boldsymbol{0}), T_{(0,\alpha)}$ with inputs $(\boldsymbol{0}, \boldsymbol{x}_2^{\alpha}),$ $T_{(0,\beta)}$ with inputs $(0, \boldsymbol{x}_2^{\beta}), T_{(\alpha,\alpha)}$ with inputs $(\boldsymbol{x}_1^{\alpha}, \boldsymbol{x}_2^{\alpha})$ and $T_{(\beta,\beta)}$ with inputs $(\boldsymbol{x}_1^{\beta}, \boldsymbol{x}_2^{\beta})$. We hence acquire in total $6 \cdot \ell$ traces.
- 4. Compute the following

$$T'_{(\alpha,\alpha)} = T_{(\alpha,\alpha)} - \overline{T}_{(\alpha,0)} - \overline{T}_{(0,\alpha)} \qquad T'_{(\beta,\beta)} = T_{(\beta,\beta)} - \overline{T}_{(\beta,0)} - \overline{T}_{(0,\beta)}$$

(*i.e.*, from each trace in $T_{(\alpha,\alpha)}$ (resp. $T_{(\beta,\beta)}$), we subtract $\overline{T}_{(\alpha,0)} + \overline{T}_{(0,\alpha)}$ (resp. $\overline{T}_{(\beta,0)} + \overline{T}_{(0,\beta)})$).

5. Compute the statistical mean equality test on the sets $T'_{(\alpha,\alpha)}$ and $T'_{(\beta,\beta)}$ as

$$t = \frac{\overline{T'}_{(\alpha,\alpha)} - \overline{T'}_{(\beta,\beta)}}{\sqrt{\frac{s_{(\alpha,\alpha)}^2 + s_{(0,\alpha)}^2 + s_{(\alpha,0)}^2 + s_{(\beta,\beta)}^2 + s_{(\beta,0)}^2 + s_{(0,\beta)}^2}}{\ell}}$$
(14)

where $s_{(i,j)}^2$ is the unbiased estimator for the population variance of $T_{(i,j)}$. If no significant difference pops up $(e.g., |t| < 4.5^{10})$, conclude that Equation 13 holds (at least, we could not contradict it).

The motivation for this test is that under the null hypothesis (*i.e.*, Equation 13 holds), $\overline{T}_{(\alpha,0)}$ converges to $d_1(\boldsymbol{x}_1^{\alpha}) + d_2(\mathbf{0})$ and $\overline{T}_{(0,\alpha)}$ to $d_1(\mathbf{0}) + d_2(\boldsymbol{x}_2^{\alpha})$. Therefore, the distribution of $T'_{(\alpha,\alpha)}$ converges to the distribution of $N+d_1(\mathbf{0},\mathbf{0})+d_2(\mathbf{x}_2^{\alpha})$. $d_2(\mathbf{0},\mathbf{0})$, and likewise for $T'_{(\beta,\beta)}$. We finally compute t such that, under the null hypothesis, it follows a standard normal distribution.

Since the noise comes from physical, electronic phenomena, its Gaussian distribution and independence on the data is a reasonable assumption. However, in case of doubt, further statistical tests can be performed. For instance, a test of Gaussianity of $Y(\boldsymbol{x}_1, \boldsymbol{x}_2)$ can be performed for fixed $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, as well as an equality test for the (co)variance of $Y(\boldsymbol{x}_1, \boldsymbol{x}_2)$ across different $(\boldsymbol{x}_1, \boldsymbol{x}_2)$.

Finally, it is worth clarifying that a statistical test can demonstrate the inability to detect dependencies with a specific number of measurements but does not prove independence. Nonetheless, if the test fails to identify a dependency for ℓ traces, it reasonably suggests that this dependency is unlikely to be exploited for an attack with significantly fewer than ℓ traces. In contrast to traditional higher-order TVLA, our test offers greater statistical power: being a first-order test, it exhibits lower sensitivity to noise, regardless of the masking order under consideration.

 $^{^{10}}$ The 4.5 threshold is given for simplicity, a better approach would be to adapt the t-score threshold with respect to the length of the traces (e.g. as proposed in [36]).

Independence in longer operation sequences. We argue that, based on knowledge of the structure of the evaluated processor and under some reasonable physical assumptions, the absence of dependency for adjacent operations guarantees that non-adjacent operations have independent leakage. Concrete justifications are provided in Supplementary Material (Section B.1).

5.2 Step 4: Implementing Noise Splitting

A necessary physical assumption in the noisy leakage model is noise independence as described in Assumption 2. Since enforcing this assumption is hard to achieve, we propose a way to relax it instead, as discussed in Section 3.4.

Consider a sequence of k operations and the corresponding leakage trace Y. Assuming that the noise is additive, we have the decomposition Y = S + N where S is the signal (typically a deterministic function of the input data as presented in the previous sections), and N is the data-independent noise. Further, thanks to the data isolation test of the previous section, we know that the signal can be rewritten as $S = \sum_{i=1}^{k} S_i$, where S_i is the leakage caused by operation *i*. We aim to decompose the noise into a sum of k + 1 independent contributions (one for each operation and a "leftover" one) $N = \sum_{i=0}^{k} N_i$. Assuming that the noise contributions N_i are Gaussian, we only have to ensure that they are not correlated to ensure independence. This gives us a decomposition of the leakage signal $Y = \sum_{i=0}^{k} Y_i$ where $Y_0 = N_0$ and $Y_i = S_i + N_i$ for $i \neq 0$, which ensures signal and noise independence between the components.

We argue that an implementation secure against an adversary with access to $\{Y_i\}_{i \in \{0,...,k\}}$ is also secure against an adversary with access to the global leakage trace Y. The former is stronger than the latter adversary since each sample in Y can be obtained by the sum of the corresponding samples in $\{Y_i\}_{i \in \{0,...,k\}}$. In our methodology, we suggest computing the noisy leakage parameter based on the split leakages Y_i , which is weaker than the noisy leakage parameter computed from Y since we split the noise N into k + 1 smaller ones, which necessarily induces more information leakage on each independent operation. Hence, an abstract circuit secure against an adversary with access to the split leakage is also secure against an adversary with access to the global leakage Y.

We will present two solutions to the above decomposition problem in the following. We first discuss a trivial solution, which has the advantage of being easily applicable but induces a loss in the security level as the size of the implementation grows. Then, we express the decomposition as an optimization problem that better scales with the size of the circuit but is more challenging to solve. We propose a direct solution to the optimization and leave the question of optimally solving it as an open problem.

Trivial Solution. We can perform a trivial split of the noise described above. Namely, for a sequence of k operations, we can split the Gaussian noise $N = \mathcal{N}(\mathbf{0}, \Sigma)$ such that $N_0 = \mathbf{0}$ and $N_i = \mathcal{N}(\mathbf{0}, (1/k) \cdot \Sigma)$ for $i \in \{1, \ldots, k\}$. This decomposition ensures that the leakage Y can be expressed as a sum of Y_i with $Y_0 = \mathbf{0}$ and $Y_i = S_i + \mathcal{N}(\mathbf{0}, (1/k) \cdot \Sigma)$, with noise and signal independence. Meanwhile, the above decomposition scales poorly with the size of the circuit. As the number of operations grows, the noise occurring on each operation decreases, leading to lower security in the noisy leakage model when applying the reduction after the relaxation (*i.e.* increasing the ARE leakage).

Hence, the noise decomposition, in addition to ensuring the independence of the components, should minimize this leakage. As explained in Section 3.6, the chosen ARE (or SD) metric for the noisy leakage model is the maximum among all operations executed. Hence, the chosen decomposition should balance the noise on all operations and scale with the number of operations executed.

Better Noise Splitting. We propose a better way to split the noise taking advantage of a relaxed noise independence assumption. For a sequence of k elementary operations, we can split the leakage trace into k sub-traces, all of the same size and including the time samples of one elementary operation each. We call distance d between two sub-traces the number of operations (or sub-traces) between them during the computation's sequence (for example, two consecutive sub-traces have distance d = 1). For the sake of simplicity, we assume that all the sub-traces have identical noise distributions and that the dependence (*i.e.* the covariance matrix) between the noises of two sub-traces solely depend on their distance d.¹¹ This means that each operation's noise covariance matrix is the same denoted Σ'_0 , and that the covariance matrix between two sub-traces with distance d is the same along the computation denoted Σ'_d . We then formulate the following relaxed noise-independence assumption.

Assumption 3 (Relaxed noise independence assumption) There exists $d_{max} \in [0, k)$ such that the sub-traces with a distance $d > d_{max}$ have null covariance: $\Sigma'_d = 0 \ \forall d > d_{max}$.

Intuitively, the above assumption captures the expectation that, after some delay, the noise in an operation sub-trace is fully independent of the noise in an earlier operation sub-trace. While we introduce it as a "relaxed assumption", we stress that it is without loss of generality since there always exists such a d_{max} . In particular, the case $d_{max} = k - 1$ captures that the independence between the noise of two operations is never reached.

Under this relaxed noise independence assumption, the global covariance matrix for k operations has the following structure (assuming $d_{max} = 1$):

$$\Sigma = \begin{pmatrix} \Sigma_0' & \Sigma_1' & & \\ \Sigma_1' & \Sigma_0' & \Sigma_1' & \\ & \Sigma_1' & \Sigma_0' & \Sigma_1' & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$
(15)

We introduce another parameter, which we call the *data-dependency depth*, ℓ_{max} . This is the number of sub-traces over which the data dependency of an

¹¹ This assumption is not strictly necessary to the application of our method but makes the presentation much simpler.

elementary operation spans. Specifically, the deterministic part of the leakage d_i of the *i*-th operation is non-zero for samples spanning on sub-traces $i, i + 1, ..., i + \ell_{max}$. This is represented in Figure 3 for $\ell_{max} = 1$.



Fig. 3: Data dependency spanning.

We now explain how a better splitting of the noise can be achieved, first by assuming $d_{max} = \ell_{max} = 1$ (and generalize later). Consider a split of the leakage in three as:

$$\begin{cases} L_1 := (S_1 + S_4 + \ldots) + N_1 \\ L_2 := (S_2 + S_5 + \ldots) + N_2 \\ L_3 := (S_3 + S_6 + \ldots) + N_3 \end{cases}$$

where $S_i = d_i(\boldsymbol{x}_i)$ denotes the signal of the *i*-th operation, which spans over time samples as represented on Figure 3, and with $N_i \sim \mathcal{N}(0, (1/3)\Sigma)$ so that $N_1 + N_2 + N_3 = N \sim \mathcal{N}(0, \Sigma)$. We have that $(L_1 + L_2 + L_3) \sim Y$, the global leakage. Let us now consider $(1/3)\Sigma = AA^T$ the Cholesky decomposition of the global covariance matrix (scaled by 1/3), so that the N_i noises follow a distribution $N_i \sim A \cdot X_i$ with $X_i \sim \mathcal{N}(0, I)$, for I the identity matrix. We have that A^{-1} has the same zero matrix blocks as Σ (see Equation 15). Namely, it can be written as:

$$A^{-1} = \begin{pmatrix} B_0 & B_1^T & & \\ B_1 & B_0 & B_1^T & \\ & B_1 & B_0 & B_1^T \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$
(16)

for some matrices B_0 , B_1 (with B_0 being symmetric). Then we get

$$\begin{cases} A^{-1} \cdot L_1 := (S'_1 + S'_4 + \ldots) + X_1 \\ A^{-1} \cdot L_2 := (S'_2 + S'_5 + \ldots) + X_2 \\ A^{-1} \cdot L_3 := (S'_3 + S'_6 + \ldots) + X_3 \end{cases}$$
(17)

with $S'_i = A^{-1} \cdot S_i$. One can then check that for each of the three leakages, L_1 , L_2 , and L_3 , the successive signals S'_i, S'_{i+3}, \ldots are strictly disjoint (meaning that

they are non-zero over disjoint time samples). This is due to the structure of A^{-1} (see Equation 16) and the fact that each S_i spans over two sub-traces. Then the S'_i span over three sub-traces so that S'_i and S'_{i+3} are disjoint. Moreover, the normalized noises $A^{-1} \cdot N_i = X_i \sim \mathcal{N}(0, I)$ can be trivially separated as

$$\begin{cases} X_1 = X'_1 + X'_4 + \dots \\ X_2 = X'_2 + X'_5 + \dots \\ X_3 = X'_3 + X'_6 + \dots \end{cases}$$
(18)

such that the X'_i span the same time samples as the S'_i . We finally split the leakage in variables $Y_i = A \cdot (S'_i + X'_i)$ which satisfy $Y = \sum_{i=1}^k Y_i$. In this splitted leakage, the amount of noise is scaled by a factor 1/3 compared to the factor 1/k of the trivial solution.

The same reasoning applies to higher values of d_{max} and ℓ_{max} . But instead of dividing the noise in 3 and scaling the covariance by factor 1/3, one has to divide it in $d_{max}+\ell_{max}+1$ and hence scale the covariance by a factor $1/(d_{max}+\ell_{max}+1)$. Depending on the noise dependency depth and data dependency depth, this might still be way better than a factor 1/k.

Towards Optimal Noise Splitting. While better than the trivial solution, the above method is non-optimal since it roughly splits the noise in $d_{max} + \ell_{max} + 1$ regardless of the signals S_i . While the signal S_i may span over the (i + 1)-th sub-trace, it might be much weaker than on the *i*-th sub-trace and should receive a smaller amount of noise than the signal S_{i+1} on these time samples.

Once again, we state our optimization problem for $d_{max} = \ell_{max} = 1$ but stress that it can be generalized to higher depths. Recall that we want to split the global covariance matrix into k+1 covariance matrices $\Sigma_0, \ldots, \Sigma_k$ such that

$$\Sigma = \Sigma_0 + \Sigma_1 + \dots + \Sigma_k \tag{19}$$

to split the leakage into n leakages: $Y_i := S_i + N_i$ with $N_i \sim \mathcal{N}(0, \Sigma_i)$.

Given the data-dependency spanning (*c.f.* Figure 3), Σ_i is only required to span the same leakage samples as d_i . Then the (lowered) global covariance matrix Σ has the following structure:



From this structure, we observe that for an operation, say the *i*-th one, we need to split the covariance matrix Σ'_0 between Y_i and Y_{i-1} (since d_{i-1} spans over time samples of the *i*-th operation). On the other hand, Σ'_1 does not need to be

split. Namely, defining Σ_i , for every $i \in \{1, \ldots, n\}$, as the symmetric positive semi-definite matrix

$$\Sigma_i := \begin{pmatrix} \bar{\Sigma}_0^{(0)} & \bar{\Sigma}_1 \\ \bar{\Sigma}_1 & \bar{\Sigma}_0^{(1)} \end{pmatrix}$$
(20)

with

$$\bar{\Sigma}_0^{(0)} + \bar{\Sigma}_0^{(1)} \le \Sigma_0' \quad \text{and} \quad \bar{\Sigma}_1 \le \Sigma_1' , \qquad (21)$$

we ensure Equation 19, where by $\Sigma' \leq \Sigma$ we mean that there exists a positive semi-definite matrix Σ'' (*i.e.* Σ'' is a covariance matrix) such that $\Sigma' + \Sigma'' = \Sigma$.

Let δ_i be a leakage metric corresponding to Y_i . Our optimization goal is

$$\min_{\bar{\Sigma}_0^{(0)}, \bar{\Sigma}_0^{(1)}, \bar{\Sigma}_1} \max \{\delta_i\}_i$$

under the constraints of Equation 21, and for Σ_i symmetric and positive semidefinite.

To sum up, under the assumptions stated above, we infer the leakage parameters which are the functions d_i , and the covariance matrices Σ'_0 and Σ'_1 and we look for a matrix Σ_i as defined in Equation 20 (in particular, a split of the Σ'_0 matrix into $\bar{\Sigma}_0^{(0)} + \bar{\Sigma}_0^{(1)}$) for which the maximal δ_i is minimized.

Choosing δ_i . Ideally, we would find the decomposition as the one that minimizes the SD or ARE leakage metric. Meanwhile, choosing metrics simpler to express can lead to optimization problems with simpler constraints, theoretically and efficiently solvable with current tools. For instance, we can choose our metric to be the multivariate SNR denoted SNR_i for the leakage Y_i , defined as the maximal eigenvalue of the matrix $\Sigma_{d_i} \tilde{\Sigma}_i^{-1}$, where Σ_{d_i} is the covariance matrix of $d_i(X)$, for X uniform over \mathcal{X} . Then, our optimization goal becomes

$$\min_{\bar{\Sigma}_0^{(0)}, \bar{\Sigma}_0^{(1)}, \bar{\Sigma}_1} \max \{ \mathrm{SNR}_i \}_i$$

under the same constraints as earlier, which leads to a convex optimization problem. Minimizing the SNR ultimately leads to low the SD or ARE and therefore appears as a natural first step before solving the more general case. It is in line with our goal to exhibit a first complete connection between the theory and practice of the masking countermeasure, leaving the question of an optimized methodology relying on the best combination of metrics and proofs as an interesting direction for further research.

6 Discussions and Perspectives

This paper proposes the first complete methodology to connect the theory and practice of provably secure masked implementations. The main goal of this approach is to obtain higher confidence security guarantees than with the current heuristic. Our methodology combines models and metrics from the literature in a principled manner to transfer formal security claims into concrete security levels that rely on hypotheses that can be validated experimentally. The main technical novelty is the relaxation of the ideal assumptions of the noisy leakage model (data and noise independence) to more realistic requirements, which still implies the ideal hypotheses without a large tightness gap. We also propose and demonstrate an experimental methodology to validate the relaxed hypotheses.

When applying our methodology "end-to-end" to an AES implementation on a commercial off-the-shelf (COTS) Cortex-M4 microcontroller, we identify two main issues: the lack of noise on this device and the non-tightness of the overall masking security proof.

The lack of noise of COTS microcontrollers is already a security issue in the practical software masking literature [25,22,21] and is therefore not one specific to our methodology. It is hard to find simple COTS microcontrollers with high noise levels. More complex MCUs are generally noisier, but studying and ensuring isolation on these is more challenging (at least when treating them as black boxes). The noise level is then a difficulty for research but not a fundamental one since it is possible to manufacture simple microcontrollers with higher noise levels (*e.g.* by adding noise engines). For more complex microcontrollers, working in a more open setting helps ensure isolation with high confidence and reasonable effort, for example, using instruction-set power leakage contracts [17]. Let us note that our methodology also applies to hardware implementations that are typically noisier and perform operations in parallel, although the approach for ensuring isolation (*i.e.* avoiding glitches, transitions, and couplings) will be different [43,27].

Regarding the tightness of the security proof, we have high noise requirements and need many shares for a given security level (Table 4). These requirements may seem excessive to practitioners, and the given security levels seem far from what the state-of-the-art attacks can achieve, or even from the security level recently proven for a single sharing [11]. Let us discuss some of the sources of the non-tightness in our security bounds and directions for improving them.

First, in our experiments, we consider the masking scheme based on the expansion technique [13]. While this scheme has the state-of-the-art minimum noise level requirement, the scaling of the security level with the leakage parameter p is sub-optimal: an optimal masking scheme with n shares would scale as $\mathcal{O}(p^n)$, while ours has a lower exponent [15,26]. This issue can be solved by using tighter random probing security proofs such as the ones based on probe distribution tables [26], but this approach requires more work to scale with large circuits.

The next step in the proof is the reduction of noisy leakage to random probing. Using the ARE metric over the SD metric is already a significant gain as it avoids a field-size loss in the proof. On our test device, the ARE worst-case metric is much larger than the SD, canceling part of the gain, but it may be due to the low noise level, as adding noise reduces the ARE vs. SD gap. If the use of worst-case leakage metric remains an issue on noisy devices, a possibility is to use the reduction to the average random probing model [42], which would relax the noise requirements at the cost of a stronger security model (hence more complex masked circuit).

Finally, reducing gate-probing to wire-probing involves a square loss in the random probing parameter. It appears as a proof artifact that could be improved.

On the practical side, our methodology relies on well-defined and realistic physical assumptions. We propose concrete tests for some of these assumptions, while others are assumed to hold using physics-based arguments. While our methodology makes a significant step towards experimentally validated hypotheses, there is still a place for improvement to replace the remaining assumptions by experimental tests, even though these assumptions may appear sound in the first place (*e.g.* noise Gaussianity, isolation of consecutive operations implying isolation of non-consecutive ones). Further, due to the nature of statistical tests, quantitative assumptions (*e.g.* independence, isolation) can never be proven, only invalidated. A fully-proven approach would instead rely on quantitative variants of these assumptions, which can be proven with high confidence using statistical tests (*e.g.* using statistical power and effect size [76]). Finally, there is room for efficiency improvements, such as optimizing the noise splitting or reducing the data isolation performance overhead.

In conclusion, we have shown how to achieve provable side-channel security in practice under relaxed leakage assumptions, although the current state of the art gives rise to constructions that are inefficient for the noise levels currently available on COTS devices. Promising directions to fully close the gap are the design of chips embedding noise engines, achieving much higher noise levels, and improving masking schemes and their security proofs. In particular, we have identified several concrete directions to improve the tightness of security proofs from both the theoretical and practical sides.

References

- 1. Information technology Security techniques Testing methods for the mitigation of non-invasive attack classes against cryptographic modules. Standard, International Organization for Standardization, Geneva, CH (Jan 2016)
- Adomnicai, A., Peyrin, T.: Fixslicing AES-like ciphers. IACR Transactions on Cryptographic Hardware and Embedded Systems 2021(1), 402-425 (2021). https://doi.org/10.46586/tches.v2021.i1.402-425, https://tches. iacr.org/index.php/TCHES/article/view/8739
- Ajtai, M.: Secure computation with information leaking to an adversary. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing. pp. 715–724. ACM Press, San Jose, CA, USA (Jun 6–8, 2011). https://doi.org/10.1145/1993636.1993731
- Ananth, P., Ishai, Y., Sahai, A.: Private circuits: A modular approach. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 427–455. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/ 978-3-319-96878-0_15
- 5. Andrychowicz, M., Dziembowski, S., Faust, S.: Circuit compilers with $O(1/\log(n))$ leakage rate. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology EURO-

CRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 586–615. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49896-5_21

- Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.: On the cost of lazy engineering for masked software implementations. In: Joye, M., Moradi, A. (eds.) Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8968, pp. 64–81. Springer (2014). https://doi.org/10.1007/978-3-319-16763-3_5, https://doi.org/10. 1007/978-3-319-16763-3_5
- Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.X., Strub, P.Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EU-ROCRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 535–566. Springer, Heidelberg, Germany, Paris, France (Apr 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56620-7_19
- Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F., Strub, P.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 535–566 (2017). https://doi.org/10.1007/978-3-319-56620-7_19, https://doi.org/10.1007/ 978-3-319-56620-7_19
- Battistello, A., Coron, J.S., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2016. Lecture Notes in Computer Science, vol. 9813, pp. 23–39. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–19, 2016). https://doi.org/10.1007/ 978-3-662-53140-2_2
- Béguinot, J., Cheng, W., Guilley, S., Liu, Y., Masure, L., Rioul, O., Standaert, F.: Removing the field size loss from duc et al.'s conjectured bound for masked encodings. In: COSADE. Lecture Notes in Computer Science, vol. 13979, pp. 86– 104. Springer (2023)
- Béguinot, J., Cheng, W., Guilley, S., Liu, Y., Masure, L., Rioul, O., Standaert, F.X.: Removing the field size loss from duc et al.'s conjectured bound for masked encodings. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 86–104. Springer (2023)
- Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 616–648. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/ 978-3-662-49896-5_22
- Belaïd, S., Coron, J.S., Prouff, E., Rivain, M., Taleb, A.R.: Random probing security: Verification, composition, expansion and new constructions. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology CRYPTO 2020, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 339–368. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56784-2_12

- Belaïd, S., Mercadier, D., Rivain, M., Taleb, A.R.: Ironmask: Versatile verification of masking security. In: 43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022. pp. 142–160. IEEE (2022). https://doi.org/10.1109/SP46214.2022.9833600, https://doi.org/10.1109/ SP46214.2022.9833600
- Belaïd, S., Rivain, M., Taleb, A.R.: On the power of expansion: More efficient constructions in the random probing model. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021, Part II. Lecture Notes in Computer Science, vol. 12697, pp. 313–343. Springer, Heidelberg, Germany, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77886-6_11
- Belaïd, S., Rivain, M., Taleb, A.R., Vergnaud, D.: Dynamic random probing expansion with quasi linear asymptotic complexity. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021, Part II. Lecture Notes in Computer Science, vol. 13091, pp. 157–188. Springer, Heidelberg, Germany, Singapore (Dec 6–10, 2021). https://doi.org/10.1007/978-3-030-92075-3_6
- Bloem, R., Gigerl, B., Gourjon, M., Hadzic, V., Mangard, S., Primas, R.: Power contracts: Provably complete power leakage models for processors. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022: 29th Conference on Computer and Communications Security. pp. 381–395. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). https://doi.org/10.1145/3548606.3560600
- Boyar, J., Matthews, P., Peralta, R.: Logic minimization techniques with applications to cryptology. Journal of Cryptology 26(2), 280-312 (Apr 2013). https: //doi.org/10.1007/s00145-012-9124-7
- Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: CHES. Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer (2004)
- Bronchain, O., Cassiers, G.: Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based KEMs. IACR Transactions on Cryptographic Hardware and Embedded Systems 2022(4), 553–588 (2022). https://doi.org/10.46586/tches.v2022.i4.553-588
- Bronchain, O., Cassiers, G., Standaert, F.X.: Give me 5 minutes: Attacking AS-CAD with a single side-channel trace. Cryptology ePrint Archive, Report 2021/817 (2021), https://eprint.iacr.org/2021/817
- 22. Bronchain, O., Standaert, F.X.: Breaking masked implementations with many shares on 32-bit software platforms. IACR Transactions on Cryptographic Hardware and Embedded Systems 2021(3), 202-234 (2021). https://doi. org/10.46586/tches.v2021.i3.202-234, https://tches.iacr.org/index.php/ TCHES/article/view/8973
- Bronchain, O., Standaert, F.X.: Breaking masked implementations with many shares on 32-bit software platforms: Or when the security order does not matter. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 202–234 (2021)
- Bruneau, N., Guilley, S., Najm, Z., Teglia, Y.: Multi-variate high-order attacks of shuffled tables recomputation. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2015. Lecture Notes in Computer Science, vol. 9293, pp. 475–494. Springer, Heidelberg, Germany, Saint-Malo, France (Sep 13–16, 2015). https://doi.org/10.1007/978-3-662-48324-4_24
- Cassiers, G., Devillez, H., Standaert, F.X., Udvarhelyi, B.: Efficient regressionbased linear discriminant analysis for side-channel security evaluations: Towards analytical attacks against 32-bit implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 270–293 (2023)

- Cassiers, G., Faust, S., Orlt, M., Standaert, F.X.: Towards tight random probing security. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 185–214. Springer, Heidelberg, Germany, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/ 978-3-030-84252-9_7
- 27. Cassiers, G., Standaert, F.X.: Provably secure hardware masking in the transitionand glitch-robust probing model: Better safe than sorry. IACR Transactions on Cryptographic Hardware and Embedded Systems 2021(2), 136-158 (2021). https://doi.org/10.46586/tches.v2021.i2.136-158, https://tches. iacr.org/index.php/TCHES/article/view/8790
- Cassiers, G., Standaert, F.: Provably secure hardware masking in the transitionand glitch-robust probing model: Better safe than sorry. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2021(2), 136-158 (2021). https://doi.org/10. 46586/tches.v2021.i2.136-158, https://doi.org/10.46586/tches.v2021.i2. 136-158
- Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_26
- Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Çetin Kaya., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2002. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer, Heidelberg, Germany, Redwood Shores, CA, USA (Aug 13–15, 2003). https:// doi.org/10.1007/3-540-36400-5_3
- Choudary, M.O.: Efficient multivariate statistical techniques for extracting secrets from electronic devices. Tech. rep., University of Cambridge, Computer Laboratory (2015)
- Choudary, O., Kuhn, M.G.: Efficient template attacks. In: CARDIS. Lecture Notes in Computer Science, vol. 8419, pp. 253–270. Springer (2013)
- 33. Coron, J., Giraud, C., Prouff, E., Renner, S., Rivain, M., Vadnala, P.K.: Conversion of security proofs from one leakage model to another: A new issue. In: Schindler, W., Huss, S.A. (eds.) Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7275, pp. 69–81. Springer (2012). https://doi.org/10.1007/978-3-642-29912-4_6, https://doi.org/10. 1007/978-3-642-29912-4_6
- Coron, J.S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) Fast Software Encryption – FSE 2013. Lecture Notes in Computer Science, vol. 8424, pp. 410–424. Springer, Heidelberg, Germany, Singapore (Mar 11–13, 2014). https://doi.org/10.1007/ 978-3-662-43933-3_21
- Coron, J.S., Rondepierre, F., Zeitoun, R.: High order masking of look-up tables with common shares. IACR Transactions on Cryptographic Hardware and Embedded Systems 2018(1), 40-72 (2018). https://doi.org/10.13154/tches.v2018. i1.40-72, https://tches.iacr.org/index.php/TCHES/article/view/832
- Ding, A.A., Zhang, L., Durvaux, F., Standaert, F.X., Fei, Y.: Towards sound and optimal leakage detection procedure. In: Eisenbarth, T., Teglia, Y. (eds.) Smart Card Research and Advanced Applications. pp. 105–122. Springer International Publishing, Cham (2018)

- Dodis, Y.: Shannon impossibility, revisited. In: ICITS. Lecture Notes in Computer Science, vol. 7412, pp. 100–110. Springer (2012)
- Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 423– 440. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_24
- Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. Journal of Cryptology 32(1), 151–177 (Jan 2019). https://doi. org/10.1007/s00145-018-9284-1
- Duc, A., Faust, S., Standaert, F.X.: Making masking security proofs concrete or how to evaluate the security of any leaking device. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 401–429. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46800-5_16
- Duc, A., Faust, S., Standaert, F.: Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. J. Cryptol. 32(4), 1263-1297 (2019). https://doi.org/10.1007/s00145-018-9277-0, https: //doi.org/10.1007/s00145-018-9277-0
- Dziembowski, S., Faust, S., Skorski, M.: Noisy leakage revisited. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 159–188. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/ 978-3-662-46803-6_6
- Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Transactions on Cryptographic Hardware and Embedded Systems 2018(3), 89–120 (2018). https://doi.org/10.13154/tches.v2018.i3.89-120, https://tches.iacr.org/index.php/TCHES/article/view/7270
- 44. Gao, S., Marshall, B., Page, D., Oswald, E.: Share-slicing: Friend or foe? IACR Transactions on Cryptographic Hardware and Embedded Systems 2020(1), 152-174 (2019). https://doi.org/10.13154/tches.v2020.i1.152-174, https: //tches.iacr.org/index.php/TCHES/article/view/8396
- Gilbert Goodwill, B.J., Jaffe, J., Rohatgi, P., et al.: A testing methodology for side-channel resistance validation. In: NIST non-invasive attack testing workshop. vol. 7, pp. 115–136 (2011)
- Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Çetin Kaya., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES'99. Lecture Notes in Computer Science, vol. 1717, pp. 158–172. Springer, Heidelberg, Germany, Worcester, Massachusetts, USA (Aug 12– 13, 1999). https://doi.org/10.1007/3-540-48059-5_15
- Grégoire, B., Papagiannopoulos, K., Schwabe, P., Stoffelen, K.: Vectorizing higherorder masking. In: Fan, J., Gierlichs, B. (eds.) COSADE 2018: 9th International Workshop on Constructive Side-Channel Analysis and Secure Design. Lecture Notes in Computer Science, vol. 10815, pp. 23–43. Springer, Heidelberg, Germany, Singapore (Apr 23–24, 2018). https://doi.org/10.1007/978-3-319-89641-0_2
- Grosso, V., Standaert, F.X.: Masking proofs are tight and how to exploit it in security evaluations. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 385–412. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78375-8_13

- Hutter, M., Schmidt, J.M.: The temperature side channel and heating fault attacks. Cryptology ePrint Archive, Report 2014/190 (2014), https://eprint.iacr.org/ 2014/190
- Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003). https://doi.org/10.1007/ 978-3-540-45146-4_27
- Journault, A., Standaert, F.X.: Very high order masking: Efficient implementation and security evaluation. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. Lecture Notes in Computer Science, vol. 10529, pp. 623–643. Springer, Heidelberg, Germany, Taipei, Taiwan (Sep 25– 28, 2017). https://doi.org/10.1007/978-3-319-66787-4_30
- Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO'96. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 1996). https://doi.org/10. 1007/3-540-68697-5_9
- Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_25
- Mangard, S.: Hardware countermeasures against DPA? A statistical analysis of their effectiveness. In: CT-RSA. Lecture Notes in Computer Science, vol. 2964, pp. 222–235. Springer (2004)
- 55. Mangard, S., Oswald, E., Standaert, F.: One for all all for one: unifying standard differential power analysis attacks. IET Inf. Secur. 5(2), 100–110 (2011)
- Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) Topics in Cryptology CT-RSA 2005. Lecture Notes in Computer Science, vol. 3376, pp. 351-365. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 14-18, 2005). https://doi.org/10.1007/978-3-540-30574-3_24
- 57. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) Topics in Cryptology CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3376, pp. 351–365. Springer (2005). https://doi.org/10.1007/978-3-540-30574-3_24, https://doi.org/10.1007/978-3-540-30574-3_24
- Mangard, S., Pramstaller, N., Oswald, E.: Successfully attacking masked AES hardware implementations. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3659, pp. 157–171. Springer (2005). https://doi.org/10.1007/11545262_12, https://doi.org/10.1007/11545262_12
- Marshall, B., Page, D., Webb, J.: MIRACLE: MIcRo-ArChitectural leakage evaluation A study of micro-architectural power leakage across many devices. IACR Transactions on Cryptographic Hardware and Embedded Systems 2022(1), 175– 220 (2022). https://doi.org/10.46586/tches.v2022.i1.175-220
- Masure, L., Dumas, C., Prouff, E.: A comprehensive study of deep learning for sidechannel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(1), 348–375 (2020)

- Masure, L., Standaert, F.X.: Prouff & rivain's formal security proof of masking, revisited: Tight bounds in the noisy leakage model. Cryptology ePrint Archive, Paper 2023/883 (2023), https://eprint.iacr.org/2023/883, https://eprint. iacr.org/2023/883
- McCann, D., Oswald, E., Whitnall, C.: Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. In: USENIX Security Symposium. pp. 199–216. USENIX Association (2017)
- 63. Müller, N., Knichel, D., Sasdrich, P., Moradi, A.: Transitional leakage in theory and practice unveiling security flaws in masked circuits. IACR Transactions on Cryptographic Hardware and Embedded Systems 2022(2), 266–288 (2022). https: //doi.org/10.46586/tches.v2022.i2.266-288
- Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: Sok: Deep learning-based physical side-channel analysis. ACM Comput. Surv. 55(11), 227:1–227:35 (2023)
- Prest, T., Goudarzi, D., Martinelli, A., Passelègue, A.: Unifying leakage models on a Rényi day. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 683– 712. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26948-7_24
- 66. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 142–159. Springer, Heidelberg, Germany, Athens, Greece (May 26–30, 2013). https://doi.org/10.1007/978-3-642-38348-9_9
- Quisquater, J., Samyde, D.: Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In: E-smart. Lecture Notes in Computer Science, vol. 2140, pp. 200–210. Springer (2001)
- Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2010. Lecture Notes in Computer Science, vol. 6225, pp. 413– 427. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–20, 2010). https://doi.org/10.1007/978-3-642-15031-9_28
- Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer, Heidelberg, Germany, Edinburgh, UK (Aug 29 – Sep 1, 2005). https://doi.org/10.1007/11545262_3
- 70. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 September 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer (2005). https://doi.org/10.1007/11545262_3, https://doi.org/10.1007/11545262_3
- Schneider, T., Moradi, A.: Leakage assessment methodology A clear roadmap for side-channel evaluations. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2015. Lecture Notes in Computer Science, vol. 9293, pp. 495–513. Springer, Heidelberg, Germany, Saint-Malo, France (Sep 13–16, 2015). https://doi.org/10.1007/978-3-662-48324-4_25
- 72. Schneider, T., Moradi, A.: Leakage assessment methodology extended version. J. Cryptogr. Eng. 6(2), 85–99 (2016). https://doi.org/10.1007/ s13389-016-0120-y, https://doi.org/10.1007/s13389-016-0120-y

- Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) Topics in Cryptology CT-RSA 2006. Lecture Notes in Computer Science, vol. 3860, pp. 208–225. Springer, Heidelberg, Germany, San Jose, CA, USA (Feb 13–17, 2006). https://doi.org/10.1007/11605805_14
- Standaert, F., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: CHES. Lecture Notes in Computer Science, vol. 5154, pp. 411–425. Springer (2008)
- Whitnall, C., Oswald, E.: A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 256–284. Springer, Heidelberg, Germany, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/ 978-3-030-34618-8_9
- 76. Whitnall, C., Oswald, E.: A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 256–284. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_9, https://doi.org/10. 1007/978-3-030-34618-8_9

Supplementary Material

A Preliminaries

A.1 Circuit Compiler

Definition 7 (Circuit Compiler). A circuit compiler *is a triplet of algorithms* (CC, Enc, Dec) *defined as follows:*

- CC (circuit compilation) is a deterministic algorithm that takes as input an abstract circuit C from a family of circuits $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ and outputs a randomized circuit \hat{C} .
- Enc (input encoding) is a probabilistic algorithm that maps an input $x \in \mathcal{V}^{\ell}$ to an encoded input $\widehat{x} \in \mathcal{V}^{\ell'}$.
- Dec (output decoding) is a deterministic algorithm that maps an encoded output $\widehat{z} \in \mathcal{V}^{m'}$ to a plain output $z \in \mathcal{V}^m$.

These three algorithms satisfy the following properties:

- *Correctness:* For every circuit C of input length ℓ , and for every $x \in \mathcal{V}^{\ell}$, we have

$$P\left[\mathsf{Dec}\left(\widehat{C}(\widehat{x})\right) = C(x) \mid \widehat{x} \leftarrow \mathsf{Enc}(x)\right] = 1$$

where $\widehat{C} = \mathsf{CC}(C)$.

- **Efficiency:** For some security parameter $\lambda \in \mathbb{N}$, the running time of CC(C)is $poly(\lambda, |C|)$, the running time of $Enc(\boldsymbol{x})$ is $poly(\lambda, |\boldsymbol{x}|)$ and the running time of $Dec(\hat{\boldsymbol{z}})$ is $poly(\lambda, |\hat{\boldsymbol{z}}|)$, where $poly(\lambda, q) = O(\lambda^{k_1}q^{k_2})$ for some constants k_1, k_2 .

A.2 Wire Vs. Gate Random Probing Leakage

We define the gate leakage procedures in the random probing model, analogously to the wire leakage procedures from Section 2.3. In other words, in the gate leakage setting, each gate leaks its internal state with probability p during the evaluation of a circuit C from a family $\mathbb{C} = (\mathcal{V}, \mathcal{G})$, where all the gate leakage events are mutually independent. The internal state of the gate can be seen as a function which depends on its inputs. Similarly to the wire leakage setting, we define the following *leaking-gates sampler*

$$G \leftarrow \mathsf{LeakingGates}(C, p)$$
,

which outputs a set G of gate labels instead of wire labels. We also define the following *assign-gates sampler*

$$oldsymbol{g} \leftarrow \mathsf{AssignGates}(C, G, oldsymbol{x})$$

which assigns to each gate of label in G, its internal state during the evaluation of C (*i.e.* g is the assignment of the internal states of the gates of C with label in G for an evaluation on input x).

By convention, we do not consider leakage on the output gates of a circuit, since when composing several circuits, these gates become input gates to the next circuit.

Figure 4 illustrates the difference between both leakages on a toy circuit.



Fig. 4: Toy circuit illustrating random probing leakage. Dashed circles (a, b, c) are input gates, while the dotted circle (d) is the output gate. In the wire leakage setting, each wire leaks with probability p, while in the gate leakage setting, each gate leaks its internal state with probability p^2 . Lemma 1 states that if the circuit is (p, ε) -RP secure in the wire setting, then it is (p^2, ε) -RP secure in the gate setting.

A.3 Proof of Lemma 1

Proof. Let C be an abstract circuit with ℓ inputs and suppose that C is (p, ε) -random probing secure in the wire leakage setting. Then, there exists a simulator that we shall denote $\operatorname{Sim}_{wire}$ such that $\operatorname{Sim}_{wire}(C) \approx_{\varepsilon} \mathcal{L}_p^{wire}(C, \operatorname{Enc}(\boldsymbol{x}))$. We now construct another simulator $\operatorname{Sim}_{gate}$ as follows. $\operatorname{Sim}_{gate}$ starts by running $\operatorname{Sim}_{wire}$, and if $\operatorname{Sim}_{wire}$ fails (or aborts), then $\operatorname{Sim}_{gate}$ aborts too. Otherwise, for each gate g in C, if all input wires to g are simulated and output by $\operatorname{Sim}_{wire}$, we let $\operatorname{Sim}_{gate}$ output a simulation of the inner state of g using the simulation of its input wires by $\operatorname{Sim}_{wire}$. Note that this is possible since the inner state of g only depends on its input wires. Since we have that $\operatorname{Sim}_{wire}(C) \approx_{\varepsilon} \mathcal{L}_p^{wire}(C, \operatorname{Enc}(\boldsymbol{x}))$, then each wire in C is simulated by $\operatorname{Sim}_{wire}$ with probability p independently of all the other wires. Consequently, each gate in C is simulated by $\operatorname{Sim}_{gate}$ aborts $\operatorname{Sim}_{gate}$ aborts if and only if $\operatorname{Sim}_{wire}$ aborts with probability ε , we get that $\operatorname{Sim}_{gate}(C) \approx_{\varepsilon} \mathcal{L}_p^{pare}(C, \operatorname{Enc}(\boldsymbol{x}))$. Hence, C is (p^2, ε) -random probing secure in the gate leakage setting, which concludes the proof.

B Procedures Proposals and Practical Experiments

B.1 Step 2: Testing Data Isolation

Independence in longer operation sequences. Let us now discuss the dependencies between operations that are not adjacent. In this section, we argue that, given an understanding of the evaluated processor's architecture and within reasonable physical assumptions, the absence of dependency between adjacent operations ensures the independence of leakage for non-adjacent operations.

Considering the processor (excluding the memory), we first assume that the "core" leakage for any clock cycle is a function of all the state stored in the processor (and the input data, *e.g.*, on the memory bus). This "core" leakage

may then get filtered (*i.e.*, undergo a linear transformation) before it is measured (*linear physics* hypothesis, denoted LP). Next, given a sufficiently simple processor, we may assume that when the processor executes m identical nonbranching/conditional instructions, the microarchitectural state of the processor does not depend on the state computed by the first of these operations (provided that the other operations do not also compute this state, and a few cycles after the last instruction retires) — m-state-erasing (denoted m-SE) hypothesis. Concretely, for an elementary processor whose state is only the architectural state, the 2-SE hypothesis is satisfied. For more complex processors, m might be larger (or even not exist). For a simple in-order processor, m-SE with m close to the pipeline depth appears as a reasonable assumption.

Our operations all follow the same structure: load the operands in two registers, perform a logic instruction, and store the result (always using the same registers). Then, between two operations, we execute several "cleaning" operations that operate on constant public data (*i.e.* whitening operations). The m-SE hypothesis, combined with LP, implies independent leakage when m - 1 cleanings separate the operations. Our two operations test presented in the previous section is a way to validate the hypotheses (and the m parameter).

Finally, regarding the memory leakage, it is reasonable to assume LP for the static leakage from the memory cells and m-SE for the remaining logic. Let us conclude this section by remarking that if an open-source processor is used, the analysis of leakage independence is greatly simplified. Indeed, as the hardware is known, we may apply the robust-probing leakage model to instructions sequences (or to verify the m-SC hypothesis).

Practical Experiments We perform the data isolation test on a real target, a **STM32F3** MCU based on an ARM Cortex-M4. Such targets are cheap and readily available. For the side-channel acquisition setup, we use the NewAE's ChipWhisperer-Lite CW1173 board together with a CW308 UFO board to connect an NAE-CW308T-STM32F3 target board (embedding the **STM32F3**). The **STM32F3** clock frequency is set to 7.37MHz. A simple way to set up the acquisition is to follow the NewAE tutorial¹².

Thanks to ChipWhisperer-Lite, one can easily acquire the power consumption of the target board with an ADC synchronized with the STM32F3 clock. That way, the acquisition sampling rate can be as low as four samples per CPU cycle and capture informative side-channel traces. To ease the acquisition and trace processing, we use the NewAE's trigger mechanism.

We implement operations as routines described in Section 3.1. Since the cost of the data isolation test is quadratic in the number of elementary operations, we limit ourselves to four elementary operations for this proof-of-concept: 8bit XOR, 8-bit AND, 8-bit Right Shift, and 8-bit Left Shift. Note that these operations would be enough to implement a masked bit-sliced AES at any chosen order for instance. Of course, more elementary operations would make the

¹² e.g. https://wiki.newae.com/Tutorial_A8_32bit_AES

implementation more efficient but would imply a higher cost in side-channel characterization.

In our example, each elementary operation relates to a single ARM-CortexM4 instruction, simplifying the analysis (listing the intermediate variables of each operation is trivial) while not mandatory for the methodology. We implement the four operations in assembly as shown in Figure 5.

xor_func:	[m 0	and_func:	[_ 0		
]	Lru]	Lru	left_shift_func	right_shift_func
ldr r1,]	[r1	ldr r1,]	[r1	: ldr r0, [r0]	: ldr r0, [r0]
eor r0, r0	r1	and r0, r0	r1	mov r0, r0, LSL 1	mov r0, r0, LSR 1
str r0,	[r2	str r0,	[r2	str r0, [r1]	str r0, [r1]

```
void whitening(void) {
  xor_func(a1Ptr, b1Ptr,
    c1Ptr);
  xor_func(a2Ptr, b2Ptr,
    c2Ptr);
  xor_func(a3Ptr, b3Ptr,
    c3Ptr);
}
```

Fig. 5: Elementary Operations (xor, and, left shift, right shift) and whitening as implemented on the STM32F3 MCU.

To perform our data isolation test, we need to capture the side-channel execution traces of two consecutive elementary operations separated by a whitening process and use the test to validate or not data isolation between the two operations. This approach must be re-iterated for all combinations of two successive elementary operations.

The whitening process does not have to be the same for all pairs of elementary operations, but for our operations selection, the acquisition setup, and the chip, a single whitening process allows us to pass all tests: three consecutive xor operations with constant public inputs¹³. The corresponding assembly code is shown in Figure 5, where {aiPtr, biPtr, ciPtr} for $i \in \{1, 2, 3\}$ are memory pointers to the two constant operands {ai, bi} and the memory location to store the result ci.

The test procedure is applied as follows:

¹³ Using only two consecutive xor operations was not enough to pass all tests for all pairs of elementary operations.

 For each pair of elementary operations, the target code is the following sequence of calls:

```
whitening();
operation1(a1Ptr, b1Ptr, c1Ptr);
whitening();
operation2(a2Ptr, b2Ptr, c2Ptr);
```

The inputs are pre-generated and loaded in memory once for all before iterating on the target code.

- Triggers surround the target code to ease the trace acquisition.
- For randomly chosen values $(\boldsymbol{x}_1^{\alpha}, \boldsymbol{x}_2^{\alpha}), (\boldsymbol{x}_1^{\beta}, \boldsymbol{x}_2^{\beta})^{14}$ we collect 10⁶ traces for each set $T_{(\alpha,\alpha)}, T_{(\alpha,0)}, T_{(0,\alpha)}, T_{(\beta,\beta)}, T_{(\beta,0)}, T_{(0,\beta)}$, for a total of 6×10^6 traces.
- From the sets of traces, we test if Equation (13) holds.

The above process is applied for each pair of elementary operations and iterated 2-3 times for different values of $(\boldsymbol{x}_1^{\alpha}, \boldsymbol{x}_2^{\alpha}), (\boldsymbol{x}_1^{\beta}, \boldsymbol{x}_2^{\beta})$. Figure 6 illustrates the test result for the pair of operations (xor, and) and a single choice of fixed inputs. Figure 6b (resp. Figure 6c) illustrates the captured leakage (through the T-Test) of the computation of xor (resp. and) and the manipulation of its inputs/outputs. Namely, Figure 6b shows high T-test values at the moment of the execution of xor with a residual leakage slowly decreasing afterward. While Figure 6c shows high T-test values later at the moment of the execution and, and no leakage is detected before, ensuring that the inputs of and were not manipulated before the execution of the operation. Then, Figure 6d illustrates the captured leakage of both xor and and simultaneously, including the individual leakages of xor and and. Figure 6e is the result of our proposed test: it represents the captured leakage of both xor and and simultaneously while individual leakages of xor and and are removed. The T-Test results show that the data isolation process (whitening function) successfully removes the combined leakage of xor and and.

B.2 Steps 3 and 4: Computing the Leakage Function and Relaxing Noise Independence

We use the experimental setting described in Section 3.7, where we consider operations with at most 2 inputs of 8 bits. We start by inferring each elementary operation's deterministic part of the leakage function separately. We use the linear regression with a specific choice of basis of functions $\mathcal{H} = \{h_1, \ldots, h_m\}$. The result of the linear regression is the set of $\{\alpha_i\}_i$ such that, for all inputs (a, b) of the selected elementary operation,

$$d(a,b) = \sum_{i=1}^{m} \alpha_i h_i(a,b) \tag{22}$$

holds. In order to capture the leakage function fully, we construct the basis of functions as follows (where n is the bit-length of the inputs a and b, here n = 8):

 $[\]overline{\mathbf{1}^{4}}$ where \boldsymbol{x}_{i} (resp. $\boldsymbol{x}_{i}^{\prime}$) contains the two inputs of operation*i*



(e) Proposed T-Test

Fig. 6: Data Isolation Test. Blue (with whitening). Orange (without whitening).

- $-h_0(a,b) = 1$
- for all $i \in [1 \cdots n]$, $h_i(a, b)$ returns the ith bit of a
- for all $i \in [1 \cdots n]$, $h_{n+i}(a, b)$ returns the ith bit of b
- for all $(i, j) \in [1 \cdots 2n]^2$, with i < j, $h_{i,j}(a, b)$ returns $h_i(a, b) \oplus h_j(a, b)$

This gives a total of $m = 1 + 2 \cdot n + n \cdot (2 \cdot n - 1) = 2 \cdot n^2 + n + 1$ functions in the function basis \mathcal{H} .

For each elementary operation, the target code is the following concatenation

```
operation(aPtr, bPtr, cPtr);
whitening();
```

surrounded by triggers to ease the trace acquisition (similarly to Section B.1). The inputs to the operations are randomly generated (using a Mersenne Twister RNG) on the chip before each execution of the target code. Then, we collect 10^6 traces in a single set and apply linear regression over the function basis \mathcal{H} on each independent time sample. The output is the set of vectors $\{\alpha_i\}_i$.

Figure 7a illustrates the convergence of the L2 norm of the coefficient vectors at two different time samples for the xor operation. We consider one vector where the SNR ratio is high (*i.e.*, more information leakage) and one where the ratio is low. We can see that for both vectors, the L2 norm converges from a few hundred traces, meaning that the linear regression can quickly estimate the coefficients of the deterministic part of the function. This behavior can further be explained by the low noise in the leakage depicted through the covariance matrix in Figure 7b. We compute this covariance matrix on the same time samples of the operation as for the deterministic function. The covariance matrix clearly shows low noise levels, which implies more information leakage. We can also observe through Figure 7a that for a sample with high SNR, the coefficients converge to more significant values than for a sample with low SNR, which gives more confidence about the results, since for samples with more information leakage, the deterministic functions should have more weight than when there is not much information leakage.



(a) Leakage Function – Deterministic Part (b) Leakage Function – Noise Part, Σ

Fig. 7: Linear Regression of the Leakage Function.

Next, we must infer the noise covariance following the relaxation from Section 5.2. Our experimental results on the chipwhisperer show that the noise levels on the chipwhisperer we use are very low. Namely, Figure 7b shows the noise covariance matrix computed from a set of traces with fixed input value for the same operation. This low noise failed our attempts to apply the optimization problem of Section 5.2. In this case, we can apply the trivial noise decomposition from Section 5.2, making the security reduction work at the cost of decreasing the noise levels even further as the size of the circuit grows.

B.3 Step 4: Estimating the Noisy Leakage Parameter

We use the same experimental setting from Section B.1, where we consider operations with at most 2 inputs of 8 bits. Then, iterating through all possible values in \mathcal{X} for a given \boldsymbol{y} amounts to performing 2¹⁶ iterations. An extra 2¹⁶ iterations are performed to compute the max for ARE or the sum for SD. Indeed, iterating over all possible values in \mathcal{X} does not scale well when considering larger inputs. In such a case, other methods can be applied to make the computation tractable. For instance, one can use the nearest-neighbor-based approach from [25] to efficiently and quickly compute the conditional probabilities and to find the max over $\boldsymbol{x} \in \mathcal{X}$ in the case of ARE. We leave the computation of the noisiness metric for larger inputs as an open research question.



Fig. 8: ARE and SD Monte-Carlo convergence as described in Section 3.5.

To simplify our analysis, and since we already observe inefficient noise levels on the chipwhisperer, we estimate the ARE and SD metrics using the original covariance matrix from Figure 7b to exhibit the noisiness levels achievable on this device in the best cases. We end our estimations by discussing the challenge of designing hardware that generates enough noise to implement circuits secure in the noisy leakage mode with reasonable security levels and finding optimal ways to solve the relaxation on such a device.

Figure 8 shows the convergence of the Monte Carlo estimation of ARE and SD metrics for the four operations as considered in Section B.1. The curves show that both metrics converge to a stable value after around 4000 samples for each operation. For the ARE metric, it converges to a maximal value of $\approx 2^{13.4}$ for the xor operation, which is enormous as this value is the same as the leakage

probability in the random probing model (*c.f.* Section 2). Recall that the final ARE value in the noisy leakage model is the maximal ARE among all operations. This result means no constructions secure in the random probing model can be used on this device. We compare this value to the SD metric, which converges to a maximal value of $\approx 2^{-0.0093}$ for the xor operation. While this value is much lower, we recall that the reduction to the random probing model with the SD metric (*c.f.* Section 2) induces a factor of 2^{16} , equal to the size of the input space (2 inputs of 8 bits each). In other words, the leakage probability in the random probing model using the SD reduction would be almost 2^{16} , which is even higher than with the ARE reduction. We observe that the values of the SD and ARE metrics are smaller for the shift operations than for the xor and and operations. We argue that this comes from the fact that the xor and and operations have two operands of 8 bits and perform an additional instruction between registers, contrary to the shift operations, leading to more information leakage and hence higher values for the noisiness metrics.

Such values for the ARE and SD noisiness metrics imply critical leakage levels on this component, making attacks most likely possible with very few traces. It also matches the conclusions of previous works (*e.g.* [25]) on this component. In addition, such levels of noisiness metrics make it challenging to have provably secure implementations on the device. To show the amount of noise that needs to be added to this component to be able to use secure constructions from the literature, we present in the following section concrete results on the AES cipher and use artificial noise that we add to the samples to demonstrate the obtained security levels.

B.4 Step 5: Applying a Random Probing Secure Compiler

To achieve arbitrary levels of security in the random probing model, current literature proposes using an *expanding compiler* [13,15,16]. We recall that the latter consists of recursively applying some base gadgets to the original circuit until achieving the desired security level. After k applications, the achieved random probing security is $\varepsilon = f^k(p_{WL})$ where p_{WL} is the random probing wire-leakage probability and f is the simulation failure function achieved by the set of gadgets. The maximal tolerated leakage probability for current 3-share and 5-share constructions is around $p_{WL} \approx 2^{-7.5}$ [15]. In our context, p_{WL} is the square root of the maximal ARE metric over the different operations, meaning that the maximum tolerated ARE is of $ARE \approx 2^{-15}$. This value is very far from what we estimate in Section B.3 on the chipwhisperer.

Adding artificial noise and impact on ARE / SD. We illustrate the impact of noise on security in the noisy leakage model by adding artificial noise to the traces acquired with the chipwhisperer. For simplicity, we add noise to each time sample of each trace, drawn from a univariate Gaussian distribution of mean 0 and standard deviation σ . We illustrate the evaluation of the ARE value for the xor operation, which showed the highest ARE and SD values in Section B.3. This operation's signal variance is about $\sigma_{\rm signal}^2 \approx 10^{-5}$ at the leakiest point during the execution of the operation. Table 1 shows the values of convergence for the SD and ARE metrics as done in Section B.3, after adding different amounts of noise to the traces (*i.e.* different σ values). The table shows that the ARE value reaches 2^{-7} when adding a univariate Gaussian noise of mean 0 and standard deviation $\sigma = 5$ to each sample in the traces. Recall that this corresponds to a leakage probability of $2^{-3.5}$ in the random probing model. Meanwhile, the SD value reaches 2^{-10} , which must then be multiplied by 2^{16} to obtain the leakage probability in the random probing model (because we consider 2-input 8-bit xor operation), making the reduction still not usable. These results showcase the difference in the tightness of the reduction from the noisy leakage to the random probing model using the SD and ARE metrics on this device. We also recall that the reduction using the ARE metric is theoretically tighter (c.f. Section 2) because the latter is a worst-case metric, matching the definition of the random probing model, a worst-case model. We then remark that the values of the ARE and SD metrics decrease as the σ value increases by the same factor. For instance, the ARE and SD values are halved whenever the σ is doubled.

To use random probing secure gadgets from the literature, as mentioned above, we need to tolerate a leakage probability of almost $2^{-7.5}$, translating to an ARE value of 2^{-15} . This value is reached when adding Gaussian noise with a significant standard deviation $\sigma \approx 1280$.

Table 1: ARE and SD values after adding noise to the leakage traces on the chipwhisperer.

σ	ARE	\mathbf{SD}	
5	2^{-7}	2^{-10}	
10	2^{-8}	2^{-11}	
20	2^{-9}	2^{-12}	
40	2^{-10}	2^{-13}	
1280	2^{-15}	2^{-18}	

Application to AES. We now illustrate the impact of the implementation's noise level on the complexity of the expanding compiler in the random probing model. We choose a provably secure implementation of AES as in [13], under the verified and relaxed leakage assumptions. We consider a bitslice implementation of AES using the 8-bit bitwise operations (xor, and, left shift logical). Table 2 gives the operation counts for such an implementation. The copy operation outputs two values equal to the single input value, and the rnd operation outputs a fresh uniform random value. N_g denotes the number of operations for the operation g in the circuit.

Table 2: AES operations complexity.

AFS Operation	Complexity		
ALS Operation	$(N_{\text{XOR}}, N_{\text{Isl}}, N_{\text{copy}}, N_{\text{and}}, N_{\text{rnd}})$		
AddRoundKey	(16, 0, 0, 0, 0)		
SubBytes	(174, 0, 111, 64, 0)		
Linear layer	(54, 16, 46, 0, 0)		
AES-128 (10 rounds)	(2440, 160, 1570, 640, 0)		

For the s-box, we use the optimized Boolean circuit from [18]. This circuit computes the AES s-box with 32 ANDs, 83 XORs and 4 NOTs. Moreover, it involves 111 copies. In our context, NOTs are computed as XORs with a constant operand, which makes 32 ANDs + 87 XORs + 111 copies. For a full SubBytes layer, composed of 16 bitsliced s-boxes, this makes 32 ANDs + 87 XORs + 174 XORs + 222 copies in terms of 16-bit operations, which is 64 ANDs + 174 XORs + 222 copies in terms of 8-bit operations.

For the linear layer, we rely on the *fixslicing approach* proposed in [2]. For the linear layer in one round, this approach requires 27 32-bit XORs, 16 wordwise rotations, 16 byte-wise rotations and 23 copies. In our context, word-wise rotations are free since they are by multiples of 8. A byte-wise rotation requires 2 LSHs (one left shift, one right shift). This makes a total of 108 XORs + 32 LSH + 92 copies in terms of 8-bit operations for the MixColumn layer for two blocks, which is 54 XORs + 16 LSH + 46 copies per block.¹⁵

We apply the expanding compiler proposed in [13] with the 3-share gadgets proposed in [15]. The LSL gadget applies the LSL operation to each input share before refreshing the sharing using a refresh gadget. Table 3 summarizes the complexities of these gadgets. As for the failure functions for the set of gadgets, we compute them using the verification tool IronMask [14].

The operation counts after k applications of the expanding compiler is given by $\mathbf{N}^k \cdot \vec{N}_{\text{AES}}$ (c.f. [13]) where **N** is the gadget gate-count matrix defined as

$$\mathbf{N} = (\vec{N}_{\text{XOR}} \mid \vec{N}_{\text{LSH}} \mid \vec{N}_{\text{copy}} \mid \vec{N}_{\text{AND}} \mid \vec{N}_{\text{RND}}) , \qquad (23)$$

and \overrightarrow{N}_{AES} is the gate-count vector for AES given by the last row of Table 2.

Table 4 summarizes the complexities of the obtained masked AES with expansion levels $k \in \{1, 2, 3, 4\}$. For each expansion level, it further gives the maximal value of the ARE in order to reach a provable security of $\varepsilon = 2^{-\lambda}$, for $\lambda \in \{32, 64, 128\}$. In order to compute the ARE value, we use the failure functions computed with **IronMask** and numerically estimate the required leakage probability p to achieve the security level given the expansion level k. This translates to the required noise level (or ARE) on the physical device to achieve

¹⁵ We note that this is for the even rounds, while odd rounds are further optimized in [2] but we consider the same count for all the rounds.

 $\begin{tabular}{|c|c|c|c|c|} \hline {\bf Complexity} \\ \hline {\bf Gadget} & \hline {\bf Complexity} \\ \hline {\bf (} N_{\rm XOR}, N_{\rm LSH}, N_{\rm copy}, N_{\rm AND}, N_{\rm RND}) \\ \hline {\bf G}_{\rm refresh} & (4,0,2,0,2) \\ \hline {\bf G}_{\rm XOR} & (11,0,4,0,4) \\ \hline {\bf G}_{\rm LSL} & (4,3,2,0,2) \\ \hline {\bf G}_{\rm copy} & (8,0,7,0,4) \\ \hline {\bf G}_{\rm AND} & (40,0,29,9,17) \\ \hline \end{tabular}$

Table 3: Complexities for the 3-share gadgets from [15] achieving (t = 1, f)-RPE.

the proven security level. We also recall that the ARE value is then obtained as p^2 .

Table 4 shows that by doing one level of expansion, which consists of replacing each gate of the circuit with the corresponding gadget, the required levels of ARE are very high and reach 2^{-136} for 128-bit security. As we further apply the expansion, this required level decreases to almost 2^{-30} for 128-bit security, but the complexity of the circuit becomes quickly impractical $(4.33 \times 10^9 \text{ operations})$. Meanwhile, to have an ARE value of 2^{-30} on the chip we use for our tests, for example, huge amounts of noise must be added ($\sigma \approx 5 \cdot 2^{23}$, c.f. Table 1).

Our results emphasize the trade-off between the physical noise on a device and the complexity of circuit implementation on this device with proven security. Higher noise levels lead to less complex constructions while achieving such noise requires specialized hardware that enables considerable noise independent of the operations. This also emphasizes the challenge of constructing such hardware, taking provable security into account and the limitations of the noise levels that can be achieved in practice. We recall that the chipwhisperer we use is far from suitable for such a case, and the question of having adapted hardware needs to be studied further in the literature.

While the complexities obtained through our results are yet to be practical, they show that it is possible to obtain physical implementations with provable security and that the noisy leakage security of the considered device highly influences the complexity of the constructions.

Expansion	Masked AES	ARE 1	for $2^{-\lambda}$	security
level k	Complexity	$\lambda = 32$	$\lambda = 64$	$\lambda = 128$
1	0.24 Mop	2^{-40}	2^{-72}	2^{-136}
2	6.14 Mop	2^{-28}	2^{-44}	2^{-76}
3	163 Mop	2^{-22}	2^{-30}	2^{-46}
4	4.33 Gop	2^{-18}	2^{-22}	2^{-30}

Table 4: Masked AES for different levels of expansion.