Towards Achieving Provable Side-Channel Security in Practice

Sonia Belaïd[1], Gaëtan Cassiers[3], Camille Mutschler[4,6], Matthieu Rivain[1], Thomas Roche[4], François-Xavier Standaert[5], and Abdul Rahman Taleb[1,2]

[1] CryptoExperts, Paris, France
[2] Sorbonne Université, CNRS, LIP6, 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 are powerful attacks that exploit a device's physical emanations to break the security of cryptographic implementations. Many countermeasures have been proposed against these attacks, especially the widely-used and efficient masking countermeasure. Nevertheless, proving the security of masked implementations is challenging. Current techniques rely on empirical approaches to validate the security of such implementations. On the other hand, the theoretical community introduced leakage models to provide formal proofs of the security of masked implementations. Meanwhile, these leakage models rely on physical assumptions that are difficult to satisfy in practice, and the literature lacks a clear framework to implement proven secure constructions on a physical device while preserving the proven security.
In this paper, we present a complete methodology describing the steps to turn an abstract masking scheme proven secure in a theoretical leakage model into a physical implementation satisfying provable security against side-channel attacks in practice. We propose new tools to enforce or relax the physical assumptions the indeal noisy leakage model rely on and provide novel ways of including them in a physical implementation. We also highlight the design goals for an embedded device to reach high levels of proven security, discussing the 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 to fully close the gap and get practical implementations proven secure against side-channel attacks on a physical device without any ideal assumption about the leakage.

Keywords: masking · provable side-channel security · random probing model · noisy leakage model · methodology · physical assumptions

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 [51], their implementation on physical devices can be vulnerable to the more powerful *side-channel attacks*. Such attacks exploit the underlying device's physical emanations, such as the execution time [51], device temperature [48], power consumption [52], or electromagnetic radiation [65] 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, Jutla, Rao, and Rohatgi [29], and by Goubin and Patarin [45] in 1999. It consists in splitting a sensitive variables 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 noisy leakage information from all the shares to learn something about the sensitive value. It has then been observed that it becomes exponentially harder to recover the secret as the number of shares grows [10,24,47].

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 [39,50,46,24]. The approaches currently found in the literature range from purely qualitative solutions such as leakage detection (e.g., ISO17825 [1,73]) or test vector leakage assessment (TVLA) [44], which aims 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 against side-channel attacks [24,25].

Having more formal and quantified security guarantees would be more satisfying but complicated as it needs to rely 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 is the t-probing model, introduced by Ishai, Sahai, and Wagner in 2003 [49]. 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 [71,66,34,12,35] thanks to its convenience to build security proofs, the 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* [64]. This model well captures the reality of embedded devices by assuming that each intermediate variable leaks a noisy function of its value. Meanwhile, proving security in the noisy leakage model [11,60] is more complex than in the probing model. Later, Duc, Dziembowski, and Faust [37,38] proposed a security reduction from the noisy model to the t-probing model. The reduction relies 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 the powerful *horizontal attacks*, and has been studied recently in many works [3,5,4,13,15,16,27].

The noisy leakage, random probing, and 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 to achieve proven security levels 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 independence* assumption), and the noise in the leakage of an operation is independent of the previous and following noises (*i.e.* the *noise independence* assumption).

The first assumption (data isolation) can be quickly broken, for instance, due to physical effects on a device. In particular, transitions occurring on memory buses or CPU registers between a pre-

viously processed value $\boldsymbol{x}_{i-1}$ and the current one $\boldsymbol{x}_i$ usually leak some information correlated to $\boldsymbol{x}_{i-1} \oplus \boldsymbol{x}_i$, which invalidates the data isolation principle [33,6]. At the hardware level, glitches further make the successive gates' leakages mutually dependent on their respective inputs [55,56,57]. At the software level, CPU synchronization limits, but does not eliminate, the issue of glitches. These issues can be avoided by adding registers and conrolling transitions [42,28] in hardware, and by trying to avoid transitions using assembly programming tricks in software [43,20,17]. However, these techniques still rely on abstract models for the leakage, and current techniques in the literature test this assumption only indirectly, by estimating the statistical security order of an implementation [7,69].

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 [47,31].

Contributions. In this paper, we contribute to the problem mentioned above by providing a detailed and precise analysis of the necessary steps to explicitly link physical implementations to the proofs in the random probing and noisy leakage models. Our contributions can be summarized as follows.

- First, we present a complete methodology describing the steps to turn an abstract masking scheme proven secure in the random probing model into a physical implementation satisfying provable security against side-channel attacks in practice. For this purpose, we discuss the different 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 need to be satisfied in practice and highlights the technical difficulties that need to be addressed before implementing formally secure circuits on a physical device.
- Then, we propose new tools to solve these technical difficulties. Namely,
  - we describe how to enforce data independence and propose a novel practical test to validate it on a physical implementation. Our test is a direct approach to test data independence, as opposed to current techniques in the literature. We also run our test on a real target to validate it, a `STM32F3` MCU, using NewAE's ChipWhisperer-Lite CW1173 board. While our test does not provide formal proof for the assumption, it is the first in the literature to directly tackle validating this hypothesis with a practical, dedicated procedure.
  - We offer a first 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 and measure the amount of noise that should be added to an implementation embedded in a commercially available component in order to reach high levels of proven security. 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 the necessary background to understand the concepts of the paper. In Section 3, we present all the steps of our methodology while leaving some important details to be discussed in later sections. Namely, in Section 4, we discuss the data independence assumption and present the proposed test. Then, in Section 5, we discuss the noise independence assumption and present our security relaxation. In Section 6, we discuss the estimation of the noisiness metrics for the noisy leakage model, which leads to the computation of the leakage probability in the random probing model. Finally, in Section 7, we present an example of a concrete AES implementation with complexity bounds given the security parameters estimated in Section 6 before concluding and providing discussions and perspectives in Section 8.

2 Technical Background

2.1 Notations

In the following, we shall denote $\mathcal{V}$ as a finite set called the variable space. For instance, we have $\mathcal{V} = \mathbb{F}_2$ when working with boolean values. We also denote $\mathcal{X}$ as the input space for the leakage, which differs from the variable space $\mathcal{V}$. For example, when considering 2-input operations, the input space is $\mathcal{X} = \mathcal{V}^2$. We also denote $\mathcal{Y}$ as the leakage distribution.

Finally, we use capital letters to denote random variables over a set or a distribution. For instance, 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 also denote $\boldsymbol{y}$ to be a leakage trace, *i.e.* a realization of $Y(\boldsymbol{x})$.

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 a finite set called the* variable space*,*
- $\mathcal{G} = \{g\}$ *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* $g : \mathcal{V}^\ell \to \mathcal{V}^m$*.*

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 $\mathcal{V}$.

An abstract circuit 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 $\boldsymbol{x} \in \mathcal{V}^\ell$ in the input gates, processing the gates from input gates to output gates, then reading the output $\boldsymbol{z} \in \mathcal{V}^m$ from the output gates. This is denoted by $\boldsymbol{z} = C(\boldsymbol{x})$. 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 $\boldsymbol{x}$).

Definition 2 (Circuit Compiler). *A* circuit compiler *is a triplet of algorithms* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *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* $\widehat{C}$*.*
– Enc *(input encoding) is a probabilistic algorithm that maps an input* $\boldsymbol{x} \in \mathcal{V}^{\ell}$ *to an encoded input* $\widehat{\boldsymbol{x}} \in \mathcal{V}^{\ell'}$*.*
– Dec *(output decoding) is a deterministic algorithm that maps an encoded output* $\widehat{\boldsymbol{z}} \in \mathcal{V}^{m'}$ *to a plain output* $\boldsymbol{z} \in \mathcal{V}^{m}$*.*

These three algorithms satisfy the following properties:

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

$$P\left[\mathsf{Dec}\big(\widehat{C}(\widehat{\boldsymbol{x}})\big) = C(\boldsymbol{x}) \;\middle|\; \widehat{\boldsymbol{x}} \leftarrow \mathsf{Enc}(\boldsymbol{x})\right] = 1 \ ,$$

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

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 $\mathbb{C} = (\mathcal{V}, \mathcal{G})$.

Wire leakage In this 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. In order 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

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

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.

Gate leakage Analogously, in the gate leakage setting, each gate leaks its internal state with probability p during the evaluation of a circuit C, 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*

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

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

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.

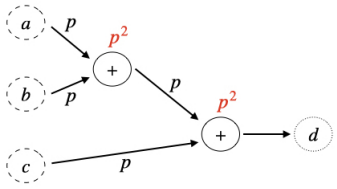


Fig. 1: Toy circuit illustrating RP leakage. Dashed circles (a, b, c) are input gates, while the dotted circle (d) is the output gate. In the wire leakage setting, each wire leaks with probability p, while in the gate leakage setting, each gate leaks its internal state with probability p^2. Lemma 1 states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

We can now formally define the (wire or gate) random probing leakage of a circuit. Figure 1 illustrates the wire and gate leakage on a toy circuit.

Definition 3 (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*

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

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

Definition 4 (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* $\boldsymbol{x} \in \mathcal{V}^\ell$*:*

$$\mathsf{Sim}(C) \approx_\varepsilon \mathcal{L}_p^{wire}(C, \mathsf{Enc}(\boldsymbol{x})) \ . \tag{1}$$

A circuit compiler $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *is* (p, ε)-random probing secure *in the wire leakage setting if for every circuit* C *the compiled circuit* $\widehat{C} = \mathsf{CC}(C)$ *is* $(p, |C| \cdot \varepsilon)$*-random probing secure in the wire leakage setting where* $|C|$ *is the size of 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* $\mathcal{L}_p^{wire}$*. In other words, only Equation 1 changes where we consider a simulator* Sim *such that*

$$\mathsf{Sim}(C) \approx_\varepsilon \mathcal{L}_p^{gate}(C, \mathsf{Enc}(\boldsymbol{x})) \ . \tag{2}$$

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.

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* $\varepsilon' = \varepsilon$*.*

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

2.4 Noisy Leakage Model

The noisy leakage model was formalized in [64]. 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 is defined as a function that takes two arguments: the value $\boldsymbol{x}$ held by the accessed part of the internal state and a random string ρ long enough to model the leakage noise. Each execution leaks the values $\big(f_i(\boldsymbol{x}_i, \rho_i)\big)_i$ 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 a noisy functions) applied to all the sub-computation input in the computation. 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 $f_i(\boldsymbol{x})$ where $\boldsymbol{x}$ is the accessed value. Note that $f_i(\boldsymbol{x})$ is not the result of a function but it can be seen as the output of a probabilistic algorithm. In particular, $f_i(\boldsymbol{x})$ 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 5 (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*

$$\beta(X|Y) := \sum_{y \in \mathrm{Range}(f)} \Pr(Y = y) \cdot \Delta((X \mid Y = y); X) \;\leq\; \delta \;, \tag{3}$$

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 [64], the authors use the L_2 norm. The authors of [37] then suggested to use L_1 norm (normalized by $\frac{1}{2}$). It was later suggested in [63] 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 [63] since the relative error is averaged over the distribution of the leakage Y in Equation 3.

As recalled hereafter, the noisy leakage metrics based on the L_1 statistical distance 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 6 (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:*

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

Definition 7. *Let X, Y be random variables over $\mathcal{X}, \mathcal{Y}$ respectively. We can define the L1 statistical distance (SD) as follows:*

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

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

$$\mathrm{ARE}(X|Y) = \mathbb{E}_{Y=y} \big[\max_x |\mathrm{PMI}_{X,Y}(x, y)| \big] \;.$$

From random probing to noisy leakage security In [37], 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 defined w.r.t. the metric $\beta(X|Y) = \mathrm{SD}(X|Y)$ 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 $\beta(X|Y) = \mathrm{ARE}(X|Y)$ in the work of Prest, Goudarzi, Martinelli, and Passelègue [63]. Those security reductions directly hold from the following key lemma.

Lemma 2 ([37,63]). *Let $\phi_p : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ the randomized function defined for every $p \in [0, 1]$ as*

$$\phi_p(x) = \begin{cases} \perp & \text{with probability } 1 - p \\ x & \text{with probability } p \end{cases} \tag{4}$$

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

$$f(x) = f'(\phi_p(x)) \quad \text{with} \quad \begin{cases} p \leq \delta \cdot |\mathcal{X}| & \text{if } \mathrm{SD}(X|f(X)) \leq \delta \\ p \leq \delta & \text{if } \mathrm{ARE}(X|f(X)) \leq \delta \end{cases} \tag{5}$$

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 [36] and the Mutual Information can (under some conditions) be expressed using the Signal-to-Noise Ratio (SNR) [53] and the correlation coefficient [19] thanks to [54]. 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 [37] using the SD metric as the *DDF reduction*, and to that from [63] 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 since it does not impose any restriction on the form of the leakage distribution. 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 $\mathcal{N}(\boldsymbol{m}_x, \Sigma)$ for some parameters $\boldsymbol{m}_x$ (mean vector) and Σ (covariance matrix), see *e.g.* [30,67]. 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 3.

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:

[7] Recall that 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 $\mathcal{X} = \mathcal{V}^2$.

– **Data isolation.** This assumption implies that a leakage function f_i corresponding to the elementary operation $g_i(\boldsymbol{x}_i)$ only depends on the current state $\boldsymbol{x}_i$ and not on previously accessed parts of the state: ..., $\boldsymbol{x}_{i-2}$, $\boldsymbol{x}_{i-1}$. The leakage is then assumed to respect some *data isolation* between successive elementary operations. However, as mentioned in Section 1, physical effects such as glitches and transitions will likely break this implicit assumption. Hence, one should take special care and enforce data isolation for our model to be valid.

– **Independent noise assumption.** The noisy leakage model assumes independence of the leakage noises from the successive elementary operations. Formally, the random tape ρ_i in each $f_i(\boldsymbol{x}_i, \rho_i)$ 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 above assumptions as the *idealized noisy leakage model*. We will see how to relax or enforce those physical assumptions so that the security of a physical implementation can reduce to that of an abstract implementation in the idealized model, which then reduces to the random probing security.

3 Metholodogy

The theoretical community introduced many construction 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. Current works in the literature do not explain all stages nor state all of the hypotheses to allow proven security claims in practice. In this section, we rigorously exhibit the steps to turn an abstract random probing secure masking scheme into a physical implementation satisfying provable security against side-channel attacks. To this purpose, the distribution of the side-channel leakage of the target device must first be characterized. We also discuss the necessity to satisfy the ideal physical assumptions of the noisy leakage model and propose some practical tools to relax or enforce them.

3.1 Overview

The proposed methodology relies on the following abstract and physical inputs:

– an abstract circuit family $\mathbb{C} = (\mathcal{V}, \mathcal{G})$,
– an RPS circuit compiler for $\mathbb{C}$,
– a cryptographic algorithm expressed as an abstract circuit $C \in \mathbb{C}$,
– a target device for the physical implementation,
– a target side-channel acquisition tool,
– a target security level λ (in bits).

The output of the methodology is a physical implementation of the cryptographic algorithm C on the target device, which achieves λ bits of proven side-channel security for the target side-channel acquisition tool and under relaxed or empirically verified physical assumptions. The necessary steps of our methodology are the following:

1. Implementing abstract gates;
2. Enforcing and testing data isolation;
3. Characterizing the leakage distribution;
4. Relaxing the noise independence;
5. Estimating the noisy leakage parameter;
6. Compiling the cryptographic implementation.

Remark 1. Our methodology is described in the context of a software implementation. The physical elementary calculation corresponds to elementary software routines. We discuss a generalization of the methodology to the case of hardware implementations in Section 8.

3.2 Step 1: Implementing Abstract Gates

The first step of our methodology is correctly implementing abstract gates in the form of software routines. A physical elementary operation abstracted as a gate by the 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 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 ALU instruction.
  str r0, [r2]
```

with the following C signature:

```
void operation_xor(const uint32* aPtr,
                   const uint32* bPtr,
                   uint32* cPtr);
```

We define a routine for each abstract gate $g \in \mathcal{G}$. When executed on the target device, the compiled code corresponding to such a routine 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);
operation3(a3Ptr, b3Ptr, c3Ptr);
...
```

The routines operation1, operation2, operation3, ... 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), (a3Ptr, b3Ptr, c3Ptr), ... are constant addresses triplets which encode the data dependency of the implementation, i.e., the wires in the abstract circuit.[8]

[8] The proposed implementation style is admittedly not very efficient. This paper mainly targets security and simplicity, translating the definition of a circuit in the leakage models to a physical implementation and leaving optimization to future works.

3.3 Step 2: Enforcing and Testing 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 to use the security reduction. Our methodology first focuses on the data isolation assumption, 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. Indeed, this is rarely true in practice, as elementary operations successively executed might leak jointly on their manipulated data. After the execution of an elementary operation, the data it has processed might have changed 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 the physical state of the CPU, which depends on the previously processed data. This is a well-known issue in the side-channel literature. In particular, this data non-isolation implies the so-called transition leakage observed and analyzed in many works [61,58]. 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. It would further leak 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 3.2, as the side-channel security can no longer be reduced to the random probing model.

Since data isolation is essential for the security proofs to hold and is a crucial step for the methodology, a developer needs to ensure it on a physical device. Since data isolation is essential for the security proofs to hold and is a crucial step for the methodology, a developer must ensure it on a physical device. In our work, we tackle this issue by proposing a way of enforcing it inspired by previous works in the literature (for instance [20,28]), and then propose a novel way of testing it on a given 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. Such an approach is a non-exact science, so we need to assess the soundness of the inserted whitening empirically. Although natural, the above way might not suffice to ensure data isolation on some devices. The effectiveness of a whitening routine depends on the microarchitecture of the device's CPU. Therefore, a developer might have to 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 the 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 $g_i(\boldsymbol{x}_i)$ does not jointly depend on the current state $\boldsymbol{x}_i$ and previously accessed parts of the state: ..., $\boldsymbol{x}_{i-2}$, $\boldsymbol{x}_{i-1}$.

In other words, f_i depends on the current state and has at most linear dependencies on the previous parts ..., $\boldsymbol{x}_{i-2}$, $\boldsymbol{x}_{i-1}$. With this relaxation, we still provide independence between the inputs of the different operations, *i.e.* data isolation.

Testing data isolation. In Section 4, we propose a novel way to test the effectiveness of a data whitening routine. The principle 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 in the leakage distribution, no function jointly depends on the inputs of both operations. We refer to Section 4 for a formal description of the test and some experimental results.

3.4 Step 3: Characterizing the Leakage Distribution

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 [32,67] that given 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 $Y(\boldsymbol{x})$ plus an additive Gaussian noise:

$$\mathcal{Y}_{\boldsymbol{x}} = d(\boldsymbol{x}) + \mathcal{N}(\mathbf{0}, \Sigma) \ , \tag{6}$$

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.:

$$d(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i h_i(\boldsymbol{x}) \ . \tag{7}$$

In our methodology, we propose relying on linear regression to estimate the deterministic leakage $d(\cdot)$. It consists in acquiring a first set of ℓ_1 traces which measure the leakage during the execution of the operation on ℓ_1 inputs generated uniformly at random and using this set to infer the coefficients $\{\alpha_i\}_{i=1,\ldots,m}$ of the leakage. Then, we can compute the sample 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.

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 effect [42].

We stress that although we propose linear regression for the leakage estimation, any other estimation method could be used with our methodology. Template attacks [30] and their combination with dimensionality reduction [72,26] are natural candidates for this purpose. Recent progresses towards exploiting machine learning as a modeling tool for side-channel analysis are eligible as well [59,62].

To validate our method, we test it on a `STM32F3` MCU using NewAE's ChipWhisperer-Lite CW1173 board and measure the efficiency of our estimation by comparing the inferred distribution $\mathcal{Y}_{\boldsymbol{x}}$ and the actual leakage. More details and experimental results are given in Section 7.

3.5 Step 4: Relaxing the Noise Independence

Next, we consider the noise independence assumption 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 as 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.

Our methodology suggests relaxing the noise independence hypothesis by splitting the noise distribution into several distributions while minimizing the leakage on each operation. In other words, given k successive elementary operations of inputs $\{(\boldsymbol{x}_i)\}_{i=1,\ldots,k}$, we can express the global leakage distribution as

$$\mathcal{Y} = \sum_{i=1}^{k} d_i(\boldsymbol{x}_i) + \mathcal{N}(\mathbf{0}, \Sigma) \tag{8}$$

where $d_i(\boldsymbol{x}_i)$ are the different deterministic signals of the operations, and N is the global noise. Thanks to the data isolation enforcement and test from Section 3.3, 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 8.

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 $\mathcal{Y}_i = d_i(\boldsymbol{x}_i) + \mathcal{N}(\mathbf{0}, \Sigma_i)$ for every $i \in [k]$. 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 an $\mathcal{Y}$ sample. More details on the security reduction and the formal description are given in Section 5. We aim to find such matrices Σ_i, which sum to the original covariance matrix Σ while minimizing the mutual information between the leakage and the signals.

3.6 Step 5: Estimating the Noisy Leakage Parameter

The noisy leakage model is defined with a security parameter representing the noise level on a leaking device. We refer to this parameter as the noisy leakage parameter δ of the δ-noisy leakage model. As explained in Section 2.4, reducing the noisy leakage model to the random probing model provides the leakage probability in the latter. More precisely, to achieve (δ, ε)-security in the noisy leakage model, an abstract circuit/implementation 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_1 metric, $\gamma = 1$ for the $\mathrm{SD}_{\mathrm{RE}}$ metric).

The factor γ depends on the chosen noisy leakage metric. There are different options available. The original security reduction [37] (DDF reduction) relies on the statistical distance between a (uniform) variable X and the same variable conditioned on its leakage Y: $\delta = E_y[\mathrm{SD}_1((X|Y = \boldsymbol{y}); X)]$. 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 $\boldsymbol{x}$ of an elementary operation), hence losing tightness in the tolerated leakage rate through the reduction. In a more recent work [63] (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, contrary to the statistical distance, which is an average-case metric. When computing $\delta = \mathrm{ARE}(X; X|Y)$, the reduction to the random probing model yields tighter results with $p = \delta$. The tighter result comes from the random probing model being a worst-case model, which better matches the definition of ARE.

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. We rely on the pointwise mutual information to compute both metrics as in Definition 7. Since, in both cases, we need to compute expected values over the set of possible leakage values, we propose to use a Monte Carlo method to empirically compute the expected value and conclude when it starts to converge towards a fixed value. More details on the evaluation are given in Section 6.

3.7 Step 6: Compiling the Cryptographic Implementation

At this stage, we have estimated the leakage parameter of each isolated noise-independent elementary operation. As defined Section 3.6, 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 target RPS compiler which, for the obtained leakage probability p and the target security level $\varepsilon = 2^{-\lambda}$, transforms C into a randomized circuit $\widehat{C}$ functionally equivalent to 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-bit of side-channel security for the target side-channel acquisition tool and under relaxed or empirically verified physical assumptions. The overall process and provable security guaranty are wrapped up hereafter.

3.8 Wrapping up

Figure 2 gives a global picture of our methodology. 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. 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.

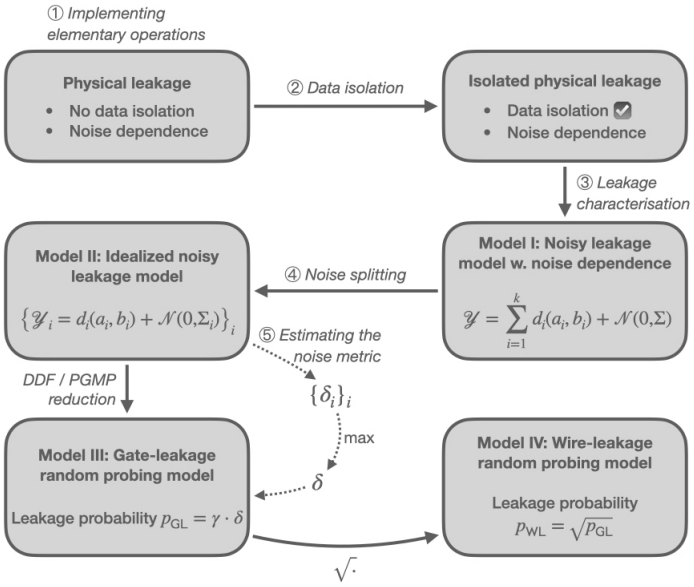


Fig. 2: Illustration of the methodology.

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 $(p_{\mathsf{WL}}, \varepsilon)$-security in the wire-leakage random probing model (Model IV). From the argument given above, we have that $\widehat{C}$ achieves $(p_{\mathsf{GL}}, \varepsilon)$-security in the gate-leakage random probing model (Model III). 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 $\{\delta_i\}$ (Model II). 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 $\mathcal{Y}$ (Model I) 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 following assumptions:

1. (Step 2) The data isolation effectively ensures that the deterministic signal can be expressed as a sum $\sum_{i=1}^{k} d_i(a_i, b_i)$;
2. (Step 3) 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. The assumption is that an adversary cannot effectively exploit the approximation error between this ideal world and the actual leakage to increase the advantage beyond $\varepsilon = 2^{-\lambda}$.

While our methodology exhibits the necessary steps to implement theoretically proven circuits in the random probing model on physical devices, it raises many questions and limitations of the current security reductions and their usability in practice. In the following sections, we discuss our proposed methods of relaxing or enforcing the physical assumptions and computing the noisy leakage parameter on the targeted device before finally exhibiting some possible applications. As we tackle these issues, we discuss the limitations of our approach and demonstrate the need for more efforts to bridge the gap between the theoretical and practical communities.

4 Enforcing and 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) [70]. 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 [40]. 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 a value), 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 [56,57] 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 [40].

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 pairs of operations. We develop a method to test the independence of the leakage functions associated with these operations. Then, we use an argument based on physics to extend the result of this test to long sequences of operations.

4.1 Leakage independence for adjacent operations

Let us consider two operations op_1 and op_2, each of them operating on two inputs (say $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$). 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

$$Y(\boldsymbol{x}_1, \boldsymbol{x}_2) = d_1(\boldsymbol{x}_1) + d_2(\boldsymbol{x}_2) + N \tag{9}$$

where d_1 and d_2 are the deterministic functions (like in [68]) 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.

We use the following statistical test to verify that Equation 9 holds:

1. Fix the inputs of the two operations to $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $(\boldsymbol{x}'_1, \boldsymbol{x}'_2)$ (similarly to *fixed-vs-fixed* leakage assessment).
2. Acquire a set $T_{(1,0)}$ of ℓ traces corresponding to executing the operations with the inputs set to $(\boldsymbol{x}_1, \boldsymbol{0})$ and compute the average trace $\overline{T}_{(1,0)}$.
3. Do the same thing for $T_{(2,0)}$ with inputs $(\boldsymbol{x}'_1, \boldsymbol{0})$, $T_{(0,1)}$ with inputs $(\boldsymbol{0}, \boldsymbol{x}_2)$, $T_{(0,2)}$ with inputs $(\boldsymbol{0}, \boldsymbol{x}'_2)$, $T_{(1,1)}$ with inputs $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $T_{(2,2)}$ with inputs $(\boldsymbol{x}'_1, \boldsymbol{x}'_2)$. We hence acquire in total $6 \cdot \ell$ traces.
4. Compute the following

$$T'_{(1,1)} = T_{(1,1)} - \overline{T}_{(1,0)} - \overline{T}_{(0,1)}$$
$$T'_{(2,2)} = T_{(2,2)} - \overline{T}_{(2,0)} - \overline{T}_{(0,2)}$$

   (*i.e.*, from each trace in $T_{(1,1)}$ (resp.$T_{(2,2)}$) , we substract $\overline{T}_{(1,0)} + \overline{T}_{(0,1)}$ (resp. $\overline{T}_{(2,0)} + \overline{T}_{(0,2)}$)).
5. Compute the statistical mean equality test on the sets $T'_{(1,1)}$ and $T'_{(2,2)}$ as

$$t = \frac{\overline{T'}_{(1,1)} - \overline{T'}_{(2,2)}}{\sqrt{\frac{s^2_{(1,1)} + s^2_{(0,1)} + s^2_{(1,0)} + s^2_{(2,2)} + s^2_{(2,0)} + s^2_{(0,2)}}{\ell}}} \tag{10}$$

   where $s^2_{(i,j)}$ is the unbiased estimator for the population variance of $T_{(i,j)}$. If no significant difference pops up (*e.g.*, $|t| < 4.5$), conclude that Equation 9 holds (at least, we could not contradict it).

The motivation for this test is that under the null hypothesis (*i.e.*, Equation 9 holds), $\overline{T}_{(1,0)}$ converges to $d_1(\boldsymbol{x}_1) + d_2(\boldsymbol{0})$ and $\overline{T}_{(0,1)}$ to $d_1(\boldsymbol{0}) + d_2(\boldsymbol{x}_2)$. Therefore, the distribution of $T'_{(1,1)}$ converges

to the distribution of $N + d_1(\mathbf{0}, \mathbf{0}) + d_2(\mathbf{0}, \mathbf{0})$, and likewise for $T'_{(2,2)}$. 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 Gaussianity is a reasonable assumption. However, in case of doubt, further statistical tests can be performed. For instance, the methodology can be extended to tests capturing noise dependencies [22], even though we do not see a physical reason why a dependency could appear as the covariance between noise samples (that would mean that the noise is multiplicative of the leakage).

Finally, we note that such a statistical test can only show that dependencies cannot be spotted with a given number of measurements, which does not demonstrate independence. However, if the test cannot spot a dependency between leakages for a given ℓ, it seems safe to assume that this dependency cannot be exploited to mount an attack in significantly less than ℓ traces. Compared to classical higher-order TVLA, our test has more statistical power: as a first-order test (independently of the order), it is less sensitive to noise than higher-order tests.

4.2 Independence in longer operation sequences

Let us now discuss the dependencies between operations that are not adjacent. 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.

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 non-branching/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).

4.3 Experimental Validation

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 used 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 a NewAE tutorial[9].

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 NewAE's trigger mechanism.

We implement operations as routines described in Section 3.2. 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: 8-bit XOR, 8-bit AND, 8-bit Right Shift, and 8-bit Left Shift.

These operations would be enough to implement a masked bit-sliced AES at any chosen order. Of course, more elementary operations would make the 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) but is not mandatory for the methodology. We implement the four operations in assembly as shown in Figure 3.

```
xor_func:
  ldr r0, [r0]
  ldr r1, [r1]
  eor r0, r1 r0
  str r0, [r2]
```

```
and_func:
  ldr r0, [r0]
  ldr r1, [r1]
  and r0, r1 r0
  str r0, [r2]
```

```
left_shift_func:
  ldr r0, [r0]
  mov r0, r0, LSL 1
  str r0, [r1]
```

```
right_shift_func:
  ldr r0, [r0]
  mov r0, r0, LSR 1
  str r0, [r1]
```

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

Fig. 3: 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 inputs. The corresponding

[9] *e.g.* `https://wiki.newae.com/Tutorial_A8_32bit_AES`

assembly code is shown in Figure 3, where $\{\mathsf{aiPtr}, \mathsf{biPtr}, \mathsf{ciPtr}\}$ for $i \in \{1, 2, 3\}$ are memory pointers to the two constant operands $\{\mathsf{ai}, \mathsf{bi}\}$ and the memory location to store the result ci.

The test procedure is applied as follows:

- For each pair of elementary operations, the target code is the following concatenation

```
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 $(\boldsymbol{x}_1, \boldsymbol{x}_2), (\boldsymbol{x}'_1, \boldsymbol{x}'_2)$[10] we collect 10^6 traces for each set $T_{(1,1)}$, $T_{(1,0)}$, $T_{(0,1)}$, $T_{(2,2)}$, $T_{(2,0)}$, $T_{(0,2)}$, for a total of $6{\times}10^6$ traces.
- From the sets of traces, we test if Equation (9) holds.

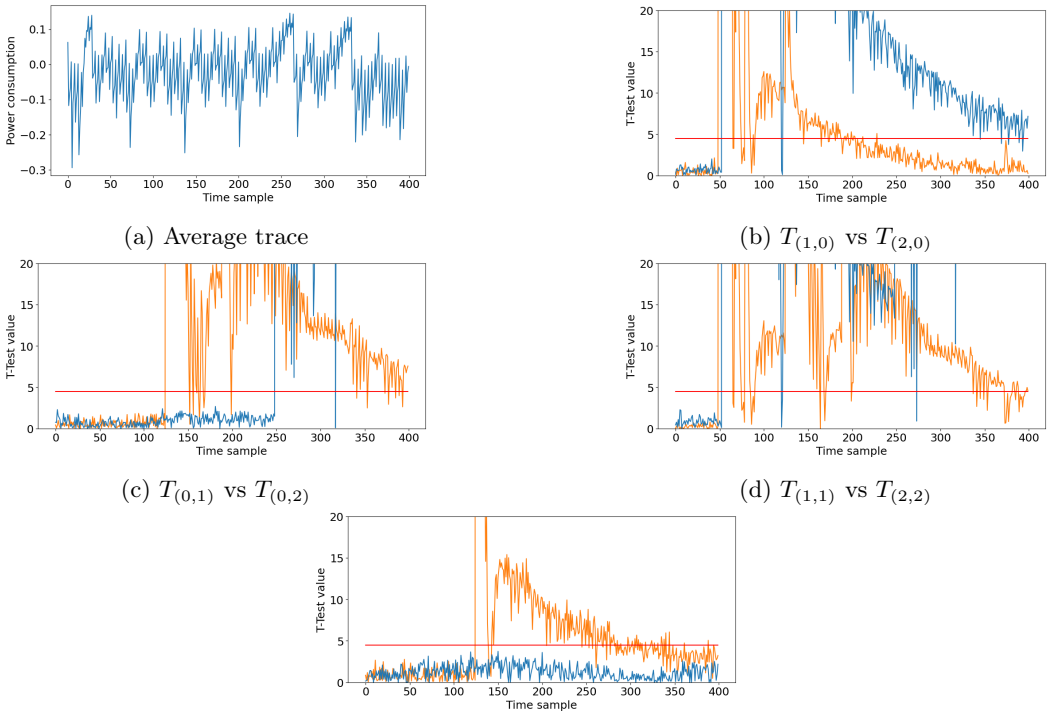
The above process is applied for each pair of elementary operations and iterated 2-3 times for different values of $(\boldsymbol{x}_1, \boldsymbol{x}_2), (\boldsymbol{x}'_1, \boldsymbol{x}'_2)$. Figure 4 illustrates the test result for the pair of operations (xor, and) and a single choice of fixed inputs. Figure 4b (resp. Figure 4c) illustrates the captured leakage (through the T-Test) of the computation of xor (resp. and) and the manipulation of its inputs/outputs. Namely, Figure 4b shows high T-test values at the moment of the execution of xor with a residual leakage slowly decreasing afterward. While Figure 4c 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 4d illustrates the captured leakage of both xor and and simultaneously, including the individual leakages of xor and and. Figure 4e 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.

5 Relaxing the Noise Independence

A necessary physical assumption in the noisy leakage model is the noise independence. In the idealized model, the noise occurring during the execution of an operation is independent of the noise that occurs before and after it. While enforcing data isolation is relatively possible, as described in Section 4, enforcing noise independence is hard to achieve, and we propose a way in our methodology to relax it instead as discussed in Section 3.5.

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

[10] where $\boldsymbol{x}_i$ (resp. $\boldsymbol{x}'_i$) contains the two inputs of `operation`i



(a) Average trace

(b) $T_{(1,0)}$ vs $T_{(2,0)}$

(c) $T_{(0,1)}$ vs $T_{(0,2)}$

(d) $T_{(1,1)}$ vs $T_{(2,2)}$

(e) Proposed T-Test

Fig. 4: Data Isolation Test. Blue (with whitening). Orange (without whitening).

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 the $\{Y_i\}_{i\in\{0,\ldots,k\}}$ 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 $\{Y_i\}_{i\in\{0,\ldots,k\}}$. 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 question.

5.1 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 $N_i = \mathcal{N}\big(\mathbf{0}, (1/k) \cdot \Sigma\big)$ 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 $Y_i = S_i + \mathcal{N}\big(\mathbf{0}, (1/k) \cdot \Sigma\big)$, with noise and signal independence between the Y_i.

Meanwhile, the above decomposition increases information leakage as the number of operations grows by decreasing the amount of noise occurring on each operation. Such an increase in the information leakage means decreasing the security level in the noisy leakage model when applying the reduction after the relaxation (*i.e.* increasing the ARE leakage). In other words, this decomposition scales poorly with the size of the circuit.

Hence, the noise decomposition, in addition to ensuring the independence of the components, should minimize this leakage. As explained in Section 3.7, 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. In the following, we tackle this by suggesting a formulation of the decomposition as an optimization problem with a trivial solution and leave the question of optimally solving it for future research.

5.2 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 the noise is identically distributed in each sub-trace and that the dependence (*i.e.* the covariance matrix) between 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:

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 apart operations is never reached.

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

$$\Sigma = \begin{pmatrix} \Sigma'_0 & \Sigma'_1 & & & \\ \Sigma'_1 & \Sigma'_0 & \Sigma'_1 & & \\ & \Sigma'_1 & \Sigma'_0 & \Sigma'_1 & \\ & & \ddots & \ddots & \ddots \\ & & & & \end{pmatrix} \tag{11}$$

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

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 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 5 for $\ell_{max} = 1$.

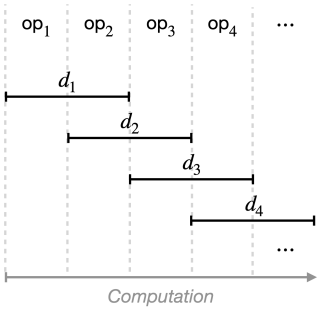


Fig. 5: Data dependency spanning.

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 5, and with $N_i \sim \mathcal{N}\big(0, (1/3)\Sigma\big)$ 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 11). Namely, it can be written as:

$$A^{-1} = \begin{pmatrix} B_0 & B_1^T & & & \\ B_1 & B_0 & B_1^T & & \\ & B_1 & B_0 & B_1^T & \\ & & \ddots & \ddots & \ddots \\ & & & & \end{pmatrix} \tag{12}$$

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

$$\begin{cases} A^{-1} \cdot L_1 := (S'_1 + S'_4 + \ldots) + X_1 \\ A^{-1} \cdot L_2 := (S'_2 + S'_5 + \ldots) + X_2 \\ A^{-1} \cdot L_3 := (S'_3 + S'_6 + \ldots) + X_3 \end{cases} \tag{13}$$

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 signal S'_i, S'_{i+3}, ... are strictly disjoint (meaning that they are non-zero over disjoint time

samples). This is due to the structure of A^{-1} (see Equation 12) 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

$$\begin{cases} X_1 = X'_1 + X'_4 + \dots \\ X_2 = X'_2 + X'_5 + \dots \\ X_3 = X'_3 + X'_6 + \dots \end{cases} \tag{14}$$

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 $1/k$.

5.3 Towards Optimal Noise Splitting

While better than the trivial solution, the above method is still non-optimal since it roughly splits the noise in 3 (or $d_{max} + \ell_{max} + 1$) 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. In that case, it should receive a smaller amount of splitted noise than the signal S_{i+1} on these time samples.

Once again, we state our optimization problem for the simpler case of $d_{max} = \ell_{max} = 1$ but stress that it can be generalized to higher depths. We recall that our goal is to split the global covariance matrix into $k+1$ covariance matrices Σ_0, Σ_1, …, Σ_k such that

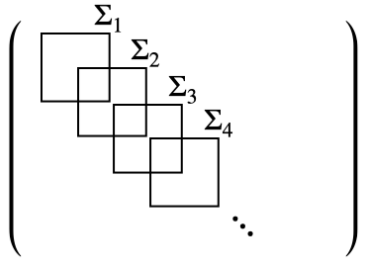
$$\Sigma = \Sigma_0 + \Sigma_1 + \cdots + \Sigma_k \tag{15}$$

such that the leakage is split into n leakages:

$$Y_i := S_i + N_i \quad \text{with} \quad N_i \sim \mathcal{N}(0, \Sigma_i) \ .$$

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 $i \neq 0$, ensuring signal and noise independence between the components.

Given the data-dependency spanning (*c.f.* Figure 5), the Σ_i matrix 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

$$\Sigma_i := \begin{pmatrix} \bar{\Sigma}_0^{(0)} & \bar{\Sigma}_1 \\ \bar{\Sigma}_1 & \bar{\Sigma}_0^{(1)} \end{pmatrix} \tag{16}$$

with

$$\bar{\Sigma}_0^{(0)} + \bar{\Sigma}_0^{(1)} \leq \Sigma'_0 \ , \tag{17}$$

and

$$\bar{\Sigma}_1 \leq \Sigma'_1 \ , \tag{18}$$

we ensure Equation 15, 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 (*e.g.* ARE). 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 17 and Equation 18, and for Σ_i symmetric and positive semi-definite.

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 16 (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. We present the above optimization problem using any leakage metric δ_i corresponding to the leakage Y_i. Ideally, we would find the decomposition as the one that minimizes our SD or ARE leakage metric. Meanwhile, choosing metrics simpler to express can lead to optimization problems with simpler constraints that are theoretically solvable with current tools more efficiently. 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. Minimizing the SNR ultimately leads to minimizing the SD or ARE and 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 Estimating the Noisy Leakage Parameter

Recall that once data isolation is enforced and tested as described in Section 4, we can empirically characterize the leakage distribution of each elementary operation as described in Section 3.4. Assuming that the leakage takes the form of Equation 6, we compute the coefficients in the deterministic function for each elementary operation (as defined in Equation 24). We propose using linear regression, but any other leakage characterization method holds with our methodology. Then, we can apply the noise relaxation described in Section 6 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 related to the noisy leakage model. Given an elementary operation, this parameter represents the amount of information that can be deduced about the inputs $\boldsymbol{x}$ of the operation, given the leakage trace $\boldsymbol{y}$. Different metrics can be used to measure this amount, as discussed in Section 3.6 and Section 2.4. The security reduction from the noisy leakage model to the random probing in the literature uses SD or ARE metrics. We propose to estimate both metrics to measure their effect on the reduction to the random probing model, where our secure constructions are. First, we look at the SD and ARE expressions using the pointwise mutual information recalled in Section 3.6. We have

$$\begin{aligned} ARE &= \mathbb{E}_Y \max_{X=\boldsymbol{x}} |PMI| \\ &= \mathbb{E}_Y \max_{X=\boldsymbol{x}} \Big| \frac{P[X=\boldsymbol{x}, Y=\boldsymbol{y}]}{P[X=\boldsymbol{x}] \cdot P[Y=\boldsymbol{y}]} - 1 \Big| \\ &= \mathbb{E}_Y \max_{X=\boldsymbol{x}} \Big| \frac{P[Y=\boldsymbol{y}|X=\boldsymbol{x}]}{\sum_{X=\boldsymbol{x}'} P[Y=\boldsymbol{y}|X=\boldsymbol{x}']} \cdot \frac{1}{P[X=\boldsymbol{x}]} - 1 \Big| \end{aligned} \tag{19}$$

As for the SD, we get:

$$\begin{aligned} SD &= \frac{1}{2} \mathbb{E}_Y \mathbb{E}_X |PMI| \\ &= \mathbb{E}_Y \frac{1}{2} \sum_{X=\boldsymbol{x}} \Big| \frac{P[Y=\boldsymbol{y}|X=\boldsymbol{x}]}{\sum_{X=\boldsymbol{x}'} P[Y=\boldsymbol{y}|X=\boldsymbol{x}']} \cdot \frac{1}{P[X=\boldsymbol{x}]} - 1 \Big| \end{aligned} \tag{20}$$

From the equations above, we need to compute the sum of the conditional probabilities $P[Y = \boldsymbol{y}|X = \boldsymbol{x}']$ for $\boldsymbol{x}' \in \mathcal{X}$ 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 19 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 20. Finally, we must compute the expected value for ARE and SD for Y.

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

$$P[Y=\boldsymbol{y}|X=\boldsymbol{x}] = \frac{1}{\sqrt{(2\pi)^n \Big|\Sigma\Big|}} \exp\Big(-\frac{1}{2} (\boldsymbol{y} - d(\boldsymbol{x}))^T \Sigma^{-1} (\boldsymbol{y} - d(\boldsymbol{x})) \Big) \tag{21}$$

where n is the number of samples in $\boldsymbol{y}$ [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

$$ARE = \frac{1}{k} \sum_{Y=\boldsymbol{y}} \max_{X=\boldsymbol{x}} \left| \frac{P[Y=\boldsymbol{y}|X=\boldsymbol{x}]}{\sum_{X=\boldsymbol{x}'} P[Y=\boldsymbol{y}|X=\boldsymbol{x}']} \cdot \frac{1}{P[X=\boldsymbol{x}]} - 1 \right| , \tag{22}$$

and the SD is computed as

$$SD = \frac{1}{2 \cdot k} \sum_{Y=\boldsymbol{y}} \sum_{X=\boldsymbol{x}} \left| \frac{P[Y=\boldsymbol{y}|X=\boldsymbol{x}]}{\sum_{X=\boldsymbol{x}'} P[Y=\boldsymbol{y}|X=\boldsymbol{x}']} \cdot \frac{1}{P[X=\boldsymbol{x}]} - 1 \right| . \tag{23}$$

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 later in our experimental results that the convergence curve reaches a plateau after some number of samples, which corresponds to the value of $\mathbb{E}_Y$.

6.1 Experimental Validation

We use the same experimental setting from Section 4.3, 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 2^{16} iterations. An extra 2^{16} 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 [26] 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.

Characterizing the leakage distribution. 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,

$$d(a, b) = \sum_{i=1}^{m} \alpha_i h_i(a, b) \tag{24}$$

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 $n = 8$):

- $h_0(a, b) = 1$
- 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 $h_i(a, b) \oplus h_j(a, b)$

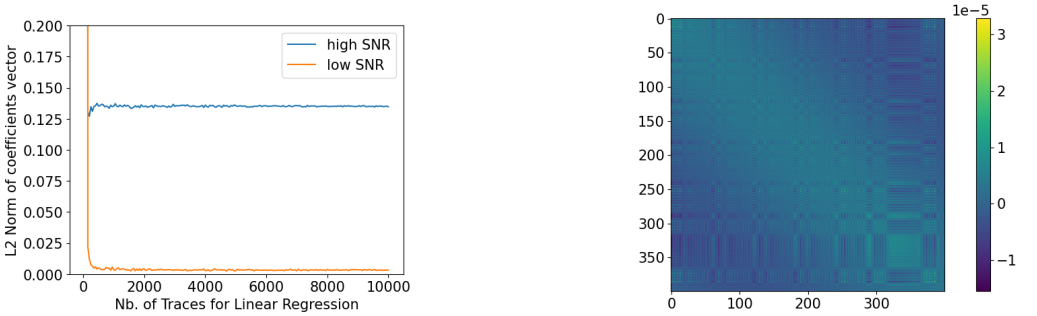
This gives a total of $m = 1 + 2 \cdot n + n \cdot (2 \cdot n - 1) = 2 \cdot n^2 + n + 1$ functions in the function basis $\mathcal{H}$.

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 4.3). 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 6a 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 6b. 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 6a 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.

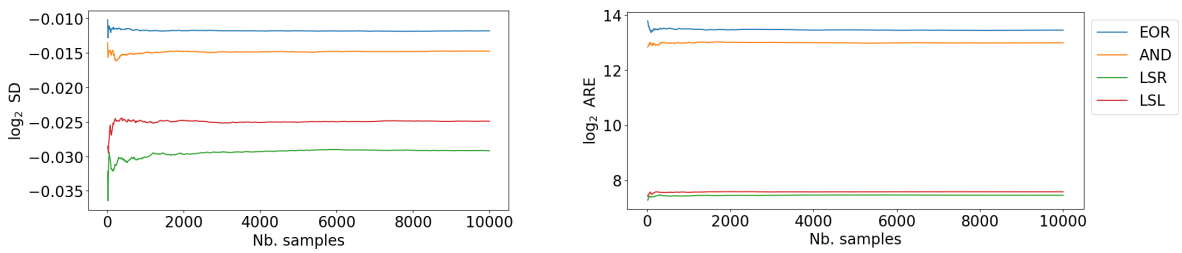


(a) Leakage Function – Deterministic Part

(b) Leakage Function – Noise Part, Σ

Fig. 6: Linear Regression of the Leakage Function.

Estimating ARE and SD metrics. Before estimating the noisiness metrics, we must infer the noise covariance following the relaxation from Section 5. Our experimental results on the chipwhisperer show that the noise levels on the chipwhisperer we use are very low. Namely, Figure 6b 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 5.3. In this case, we can apply the trivial noise decomposition from Section 5, making the security reduction work at the cost of decreasing the noise levels even further as the size of the circuit grows. 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 6b to exhibit



(a) SD

(b) ARE

Fig. 7: ARE and SD Monte-Carlo convergence as described in Section 6.

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 7 shows the convergence of the Monte Carlo estimation of ARE and SD metrics for the four operations as considered in Section 4.3. 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.* [26]) 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.

7 Discussion on the Application of the Methodology

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 $\varepsilon = f^k(p_{\mathsf{WL}})$ 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 $p_{\mathsf{WL}} \approx 2^{-7.5}$ [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 6.1 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 6.1. This operation's signal variance is about $\sigma^2_{\text{signal}} \approx 10^{-5}$ 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 6.1, 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**	**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

Table 2: AES operations complexity.

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

gives the operation counts for such an implementation which are detailed in Section A. 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.

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 $\mathbf{N}^k \cdot \overrightarrow{N}_{\text{AES}}$ (*c.f.* [13]) where $\mathbf{N}$ is the gadget gate-count matrix defined as

$$\mathbf{N} = \left(\overrightarrow{N}_{\text{XOR}} \mid \overrightarrow{N}_{\text{LSH}} \mid \overrightarrow{N}_{\text{copy}} \mid \overrightarrow{N}_{\text{AND}} \mid \overrightarrow{N}_{\text{RND}}\right) , \tag{25}$$

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

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

Gadget	**Complexity** $(N_{\text{XOR}}, N_{\text{LSH}}, N_{\text{copy}}, N_{\text{AND}}, N_{\text{RND}})$
G_{refresh}	$(4, 0, 2, 0, 2)$
G_{XOR}	$(11, 0, 4, 0, 4)$
G_{LSL}	$(4, 3, 2, 0, 2)$
G_{copy}	$(8, 0, 7, 0, 4)$
G_{AND}	$(40, 0, 29, 9, 17)$

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

Table 3 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 (4.33×10^9

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.

Table 4: Masked AES for different levels of expansion.

Expansion level k	**Masked AES Complexity**	**ARE for** $2^{-\lambda}$ **security**		
		$\lambda = 32$	$\lambda = 64$	$\lambda = 128$
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}

8 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 the heuristic solutions used so far. It is based on several steps that combine different models and metrics introduced in 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, that still imply the ideal hypotheses without large tightness gap. We also propose and demonstrate an experimental methodology to validate the relaxed hypotheses.

When applying our novel methodology "end-to-end" to protect an AES implementation on a COTS Cortex-M4 microcontroller, and identify a 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 has been pointed as a security issue in the practical software masking literature [26,23,21] and is therefore not an issue specific to our methodology. It is indeed hard to find simple COTS microcontrollers with high noise levels. More complex MCUs are generally more noisy, but it is harder to study and ensure isolation on these (at least when treating them as black-boxes). The noise level is therefore a difficulty for research but not 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 will help with ensuring isolation with high confidence and reasonable effort, for example by using instruction-set power leakage contracts [17]. Let us finally note that our methodology is also applicable to hardware implementations which are typically more noisy and perform operations in parallel, although the approach for ensuring isolation (*i.e.* avoiding glitches, transitions and couplings) will be different [42,28].

Regarding the tightness of the security proof, we have very noise requirements and need a large number of shares for a given security level (table 4). These requirements may seem excessive to a practitionners and the security level given seems very 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 now discuss some of the sources of the non-tightness in our security bounds and discuss directions for improving them.

First, the masking scheme we consider is based on the expansion technique [13]. While this scheme has state of the art minimum noise level requirement, the scaling of the security level with the noise (*i.e.* the noisy leakage parameter p) is suboptimal: an optimal masking scheme with n shares would scale as $\mathcal{O}(p^n)$, while ours has a lower exponent [15,27]. This issue can be solved by using tighter random probing security proofs such as the ones based one probe distribution tables [27], although that approach requires further work to ensure composability in large-scale circuits.

The next step in the proof is the reduction of noisy leakage to random probing. The use of 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 the the SD, cancelling part of the gain, but this may be due to the very 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 move towards the reduction to the average random probing model [41], which would relax the noise requirements, at the cost of a stronger security model for the masking scheme.

Finally, our reduction of gate-probing to wire-probing involves a square loss in the random probing parameter. This appears to be a proof artifact that might be improved upon.

On the practical side of the bridge, our methodology relies on well-defined and realistic physical assumptions. We propose experimental tests for some of these assumptions, while other are simply assumed to hold using physics-based arguments. While our methodology makes a significant step towards experimentally-validated hypotheses, there is still some margin for improvement, in order to replace the remaining assumptions by experimental tests, even though these assumptions may appear sound in the first place (*e.g.* noise Gaussianity, m-SE, LP). Further, many of these assumptions are qualitative (*e.g.* independence, isolation). Due to the nature of statistical tests, such assumptions can never be proven, only invalidated. While heuristics based on statistical power and effect size may be used [74], a fully-proven approach would rather rely on quantitative variants of these assumptions, which can be proven with high confidence using statistical tests. Finally, there is some room for efficiency improvements in our methodology, for example by optimizing the noise splitting or by reducing the 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 the improvement of 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 side and the practical side.

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, January 2016.
2. Alexandre Adomnicai and Thomas Peyrin. Fixslicing AES-like ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):402–425, 2021. `https://tches.iacr.org/index.php/TCHES/article/view/8739`.
3. Miklós Ajtai. Secure computation with information leaking to an adversary. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 715–724, San Jose, CA, USA, June 6–8, 2011. ACM Press.
4. Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.
5. Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
6. Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.
7. Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.
8. Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *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*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, 2017.
9. Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39, Santa Barbara, CA, USA, August 17–19, 2016. Springer, Heidelberg, Germany.
10. Julien Béguinot, Wei Cheng, Sylvain Guilley, Yi Liu, Loïc Masure, Olivier Rioul, and François-Xavier Standaert. Removing the field size loss from duc et al.'s conjectured bound for masked encodings. In *COSADE*, volume 13979 of *Lecture Notes in Computer Science*, pages 86–104. Springer, 2023.
11. Julien Béguinot, Wei Cheng, Sylvain Guilley, Yi Liu, Loïc Masure, Olivier Rioul, and François-Xavier Standaert. 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*, pages 86–104. Springer, 2023.
12. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
13. Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Abdul Rahman Taleb. Random probing security: Verification, composition, expansion and new constructions. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

14. Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. Ironmask: Versatile verification of masking security. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 142–160. IEEE, 2022.
15. Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 313–343, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany.
16. Sonia Belaïd, Matthieu Rivain, Abdul Rahman Taleb, and Damien Vergnaud. Dynamic random probing expansion with quasi linear asymptotic complexity. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 157–188, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany.
17. Roderick Bloem, Barbara Gigerl, Marc Gourjon, Vedad Hadzic, Stefan Mangard, and Robert Primas. Power contracts: Provably complete power leakage models for processors. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 381–395, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.
18. Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *Journal of Cryptology*, 26(2):280–312, April 2013.
19. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
20. Olivier Bronchain and Gaëtan Cassiers. 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.
21. Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. Cryptology ePrint Archive, Report 2021/817, 2021. `https://eprint.iacr.org/2021/817`.
22. Olivier Bronchain, Tobias Schneider, and François-Xavier Standaert. Multi-tuple leakage detection and the dependent signal issue. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):318–345, 2019.
23. Olivier Bronchain and François-Xavier Standaert. 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://tches.iacr.org/index.php/TCHES/article/view/8973`.
24. Olivier Bronchain and François-Xavier Standaert. 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*, pages 202–234, 2021.
25. Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Teglia. Multi-variate high-order attacks of shuffled tables recomputation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 475–494, Saint-Malo, France, September 13–16, 2015. Springer, Heidelberg, Germany.
26. Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient regression-based linear discriminant analysis for side-channel security evaluations: Towards analytical attacks against 32-bit implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 270–293, 2023.
27. Gaëtan Cassiers, Sebastian Faust, Maximilian Orlt, and François-Xavier Standaert. Towards tight random probing security. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 185–214, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.
28. Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):136–158, 2021. `https://tches.iacr.org/index.php/TCHES/article/view/8790`.
29. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.
30. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany.
31. Marios O Choudary. Efficient multivariate statistical techniques for extracting secrets from electronic devices. Technical report, University of Cambridge, Computer Laboratory, 2015.

32. Omar Choudary and Markus G. Kuhn. Efficient template attacks. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.
33. Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.
34. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany.
35. Jean-Sébastien Coron, Franck Rondepierre, and Rina Zeitoun. High order masking of look-up tables with common shares. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):40–72, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/832`.
36. Yevgeniy Dodis. Shannon impossibility, revisited. In *ICITS*, volume 7412 of *Lecture Notes in Computer Science*, pages 100–110. Springer, 2012.
37. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.
38. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. *Journal of Cryptology*, 32(1):151–177, January 2019.
39. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
40. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *J. Cryptol.*, 32(4):1263–1297, 2019.
41. Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 159–188, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
42. Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. 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://tches.iacr.org/index.php/TCHES/article/view/7270`.
43. Si Gao, Ben Marshall, Dan Page, and Elisabeth Oswald. Share-slicing: Friend or foe? *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):152–174, 2019. `https://tches.iacr.org/index.php/TCHES/article/view/8396`.
44. Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.
45. Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Heidelberg, Germany.
46. Benjamin Grégoire, Kostas Papagiannopoulos, Peter Schwabe, and Ko Stoffelen. Vectorizing higher-order masking. In Junfeng Fan and Benedikt Gierlichs, editors, *COSADE 2018: 9th International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 10815 of *Lecture Notes in Computer Science*, pages 23–43, Singapore, April 23–24, 2018. Springer, Heidelberg, Germany.
47. Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 385–412, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.
48. Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. Cryptology ePrint Archive, Report 2014/190, 2014. `https://eprint.iacr.org/2014/190`.
49. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

50. Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643, Taipei, Taiwan, September 25–28, 2017. Springer, Heidelberg, Germany.
51. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.
52. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.
53. Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.
54. Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Inf. Secur.*, 5(2):100–110, 2011.
55. Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365, San Francisco, CA, USA, February 14–18, 2005. Springer, Heidelberg, Germany.
56. Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
57. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
58. Ben Marshall, Dan Page, and James Webb. 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.
59. Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020.
60. Loïc Masure and François-Xavier Standaert. 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`.
61. David McCann, Elisabeth Oswald, and Carolyn Whitnall. Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. In *USENIX Security Symposium*, pages 199–216. USENIX Association, 2017.
62. Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11):227:1–227:35, 2023.
63. Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a Rényi day. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.
64. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.
65. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
66. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427, Santa Barbara, CA, USA, August 17–20, 2010. Springer, Heidelberg, Germany.
67. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46, Edinburgh, UK, August 29 – September 1, 2005. Springer, Heidelberg, Germany.

68. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.
69. Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513, Saint-Malo, France, September 13–16, 2015. Springer, Heidelberg, Germany.
70. Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptogr. Eng.*, 6(2):85–99, 2016.
71. Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225, San Jose, CA, USA, February 13–17, 2006. Springer, Heidelberg, Germany.
72. François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.
73. Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.
74. Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In Steven D. Galbraith and Shiho Moriai, editors, *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*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284. Springer, 2019.

A Operation Counts for AES

We detail hereafter the operation counts for the considered bitslice implementation of AES with 8-bit logical operations.

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 + 111 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 word-wise 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.[12]

[12] 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.