Unifying Leakage Models: from Probing Attacks to Noisy Leakage

Alexandre Duc[1,⋆], Stefan Dziembowski[2,3,⋆⋆], and Sebastian Faust[1,⋆⋆⋆]

[1] Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
[2] University of Warsaw, Poland
[3] Sapienza University of Rome, Italy

Abstract. A recent trend in cryptography is to formally show the leakage resilience of cryptographic implementations in a given leakage model. One of the most prominent leakage models – the so-called bounded leakage model – assumes that the amount of leakage is a-priori bounded. Unfortunately, it has been pointed out that the assumption of bounded leakages is hard to verify in practice. A more realistic assumption is to assume that leakages are sufficiently noisy, following the engineering observation that real-world physical leakages are inherently noisy. While the noisy leakage assumption has first been studied in the seminal work of Chari et al. (CRYPTO 99), the recent work of Prouff and Rivain (Eurocrypt 2013) provides the first analysis of a full masking scheme under a physically motivated noise model. In particular, the authors show that a block-cipher implementation that uses an additive masking scheme is secure against noisy leakages. Unfortunately, the security analysis of Prouff and Rivain has three important shortcomings: (1) it requires leak-free gates, (2) it considers a restricted adversarial model (random message attacks), and (3) the security proof has limited application for cryptographic settings. In this work, we provide an alternative security proof in the same noisy model that overcomes these three challenges. We achieve this goal by a new reduction from noisy leakage to the important theoretical model of probing adversaries (Ishai et al – CRYPTO 2003). Our work can be viewed as a next step of closing the gap between theory and practice in leakage resilient cryptography: while our security proofs heavily rely on concepts of theoretical cryptography, we solve problems in practically motivated leakage models.

1 Introduction

Physical side-channel attacks that exploit leakage emitting from devices are an important threat for cryptographic implementations. Prominent sources of such physical leakages include the running time of an implementation [17], its power consumption [18] or electromagnetic radiation emitting from it [26]. A large body of recent applied and theoretical research attempts to incorporate the information an adversary obtains from the leakage into the security analysis and develops countermeasures to defeat common side-channel attacks [4, 14, 20, 1, 9, 31, 30]. While there is still a large gap between what theoretical models can achieve and what side-channel information is measured in practice, some recent important works propose models that better go align with the perspective of cryptographic engineering [29, 24, 30]. Our work follows this line of research by analyzing the security of a common countermeasure – the so-called masking countermeasure – in the model of Prouff and Rivain [24]. Our analysis works by showing that security in certain theoretical leakage models implies security in the model of [24], and hence may be seen as a first attempt to *unify* the large class of different leakage models used in recent results.

⋆ Supported by a grant of the Swiss National Science Foundation, 200021_143899/1.
⋆⋆ Received founding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement number 207908.
⋆⋆⋆ Received funding from the Marie Curie IEF/FP7 project GAPS, grant number: 626467.

The masking countermeasure. A large body of work on cryptographic engineering has developed countermeasures to defeat side-channel attacks (see, e.g., [19] for an overview). While many countermeasures are specifically tailored to protect particular cryptographic implementations (e.g., key updates or shielded hardware), a method that generically works for most cryptographic schemes is *masking* [13, 2, 23, 31]. The basic idea of a masking scheme is to secret share all sensitive information, including the secret key and all intermediate values that depend on it, thereby making the leakage independent of the secret data. The most prominent masking scheme is the Boolean masking: a bit b is encoded by a random bit string $(b_1, \ldots, b_n)$ such that $b = b_1 \oplus \ldots \oplus b_n$. The main difficulty in designing masking schemes is to develop masked operations, which securely compute on encoded data and ensure that *all* intermediate values are protected.

Masking against noisy leakages. Besides the fact that masking can be used to protect arbitrary computation, it has the advantage that it can be analyzed in formal security models. The first work that formally studies the soundness of masking in the presence of leakage is the seminal work of Chari et al. [4]. The authors consider a model where each share b_i of an encoding is perturbed by Gaussian noise and show that the number of noisy samples needed to recover the encoded secret bit b grows exponential with the number of shares. As stated in [4], this model matches real-world physical leakages that inherently are noisy. Moreover, many practical solutions exist to amplify leakage noise (see for instance the works of [6, 5, 19]).

One limitation of the security analysis given in [4] is the fact that it does not consider leakage emitting from masked computation. This shortcoming has been addressed in the recent important work of Prouff and Rivain [24], who extend at Eurocrypt 2013 the noisy leakage model of Chari et al. [4] to also include leakage from the masked operations. Specifically, they show that a variant of the construction of Ishai et al. [14] is secure even when there is noisy leakage from all the intermediate values that are produced during the computation. The authors of [24] also generalize the noisy leakage model of Chari et al. [4] to a wider range of leakage functions instead of considering only the Gaussian one. While clearly noisy leakage is closer to physical leakage occurring in real world, the security analysis of [24] has a number of shortcomings which puts strong limitations in which settings the masking countermeasure can be used and achieves the proved security statements. In particular, like earlier works on leakage resilient cryptography [8, 10] the security analysis of Prouff and Rivain relies on so-called *leak-free* gates. Moreover, security is shown in a restricted adversarial model that assumes that plaintexts are chosen uniformly during an attack and the adversary does not exploit joint information from the leakages and, e.g., the ciphertext. We discuss these shortcomings in more detail in the next section.

1.1 The work of Prouff and Rivain [25]

Prouff and Rivain [25] analyze the security of a block-cipher implementation that is masked with an additive masking scheme working over a finite field $\mathbb{F}$. More precisely, let t be the security parameter then a secret $s \in \mathbb{F}$ is represented by an encoding $(X_1, \ldots, X_t)$ such that each $X_i \leftarrow \mathbb{F}$ is uniformly random subject to $s = X_1 \oplus \ldots \oplus X_t$. As discussed above the main difficulty in designing secure masking schemes is to devise masked operations that work on masked values. To this end, Prouff and Rivain use the original scheme of Ishai et al. [14] augmented with some techniques from [3, 27] to work over larger fields and to obtain a more efficient implementation. The masked operations are built out of several smaller

components. First, a leak-free operation that refreshes encodings, i.e., it takes as input an encoding $(X_1, \ldots, X_t)$ of a secret s and outputs a freshly and independently chosen encoding of the same value. Second, a number of leaky elementary operations that work on a constant number of field elements. For each of these elementary operations the adversary is given leakage $f(X)$, where X are the inputs of the operation and f is a noisy function. Clearly, the noise-level has to be high enough so that given $f(X)$ the values of X is not completely revealed. To this end, the authors introduce the notion of a *bias*, which informally says that the statistical distance between the distribution of X and the conditional distribution $X|f(X)$ is bounded by some parameter.

While noisy leakages are certainly a step in the right direction to model physical leakage, we detail below some of the limitations of the security analysis of Prouff and Rivain [24]:

1. *Leak-free components:* The assumption of leak-free computation has been used in earlier works on leakage resilient computation [10, 8]. It is a strong assumption on the physical hardware and, as stated in [24], an important limitation of the current proof approach. The leak-free component of [24] is a simple operation that takes as input an encoding and refreshes it. While the computation of this operation is supposed to be completely shielded against leakage, the inputs and the outputs of this computation may leak. Notice that the leak-free component of [24] depends on the computation that is carried out in the circuit by takeing inputs. In particular, this means that the computation of the leak-free component depends on secret information, which makes it harder to protect in practice and is different from earlier works that use leak-free components [10, 8].
2. *Random message attacks:* The security analysis is given only for random message attacks. In particular, it is assumed that every masked secret is a uniformly random value. This is in contrast to most works in cryptography, which usually consider at least a chosen message attack. When applied to a block-cipher, their proof implies that the adversary has only access to the *leakage* of the system without knowing which plaintext was used nor which ciphertext was obtained. Hence, the proof does not cover chosen plaintext or chosen ciphertext attacks. However, it is true that it is not clear how chosen message attacks change the picture in standard DPA attacks [32].
3. *Mutual-information-based security statement:* The final statement of Theorem 4 in [24] only gives a bound on the mutual information of the key and the leakages from the cipher. In particular, this does not include information that an adversary may learn from exploiting joint information from the leakages and plaintext/ciphertext pairs. Notice that the use of mutual information gets particularly problematic under continuous leakage attacks, since multiple plaintext/ciphertext pairs information theoretically completely reveal the secret key. The standard security notion used, e.g., in Ishai et al. is simulation-based and covers such subtleties when dealing with Shannon information theory.
4. *Strong noise requirements:* The amount of noise that is needed depends on the number of shares and on the size of the field which might be a bit unnatural. Moreover, the noise is independently sampled for each of the elementary operation that have constant size.

1.2 Our contribution

We show in this work how to eliminate limitations 1-3 by a simple and elegant simulation-based argument and a reduction to the so-called t-probing adversarial setting [14] (that in this paper we call the *t-threshold-probing model* to emphasize the difference between this model

and the *random*-probing model defined later.). The t-threshold-probing model considers an adversary that can learn the value of t intermediate values that are produced during the computation and is often considered as a good approximation for modelling higher-order attacks. We notice that limitation 4 from above is what enables our security analysis. The fact that the noise is independent for each elementary operation allows us to formally prove security under an *identical* noise model as [24], but using a simpler and improved analysis. In particular, we are able to show that the original construction of Ishai et al. satisfies the standard simulation-based security notion under noisy leakages without relying on any leak-free components. We emphasize that our techniques are very different (and much simpler) than the recent breakthrough result of Goldwasser and Rothblum [12] who show how to eliminate leak-free gates in the bounded leakage model. We will further discuss related works in Section 1.3.

Our proof considers three different leakage models and shows connections between them. One may view our work as a first attempt to "reduce" the number of different leakage models, which is in contrast to many earlier works that introduced new leakage settings. Eventually, we are able to reduce the security in the noisy leakage model to the security in the t -threshold-probing model. This shows that, for the particular choice of parameters given in [24], security in the t-threshold-probing model implies security in the noisy leakage model. This goes align with the common approach of showing security against t-order attacks, which usually requires to prove security in the t–threshold-probing model. Moreover, it shows that the original construction of Ishai et al. that has been used in many works on masking (including the work of Prouff and Rivain) is indeed a sound approach for protecting against side-channel leakages when assuming that they are sufficiently noisy. We give some more details on our techniques below.

From noisy leakages to random probes. As a first step in our security proof we show that we can simulate any adversary in the noisy leakage model of Prouff and Rivain with an adversary in a simpler noise model that we name a *random probing adversary* and is similar to a model introduced in [14]. In this model, an adversary recovers an intermediate value with probability ϵ and obtains a special symbol $\perp$ with probability $1-\epsilon$. This reduction shows that this model is worth studiying, although from the engineering perspective it may seem unnatural.

From random probes to the t-threshold-probing model. We show how to go from the random probing adversary setting to the more standard t-threshold-probing adversary of Ishai et al. in [14]. This step is rather easy as due to the independency of the noise we can apply Chernoff's bound almost immediately. One technical difficulty is that the work of Prouff and Rivain considers joint noisy leakage from elementary operations, while the standard t-threshold-probing setting only talks about leakage from wires. Notice, however, that the elementary operations of [24] only depend on two inputs and, hence, it is not hard to extend the result of Ishai et al. to consider "gate probing adversary" by tolerating a loss in the parameters. Finally, our analysis enables us to show security of the masking based countermeasure without the limitations 1-3 discussed above.

Leakage resilient circuits with simulation-based security. In our security analysis we use the the framework of leakage resilient circuits introduced in the seminal work of Ishai et al. [14]. A circuit compiler takes as input the description of a cryptographic scheme C with secret key K, e.g., a circuit that describes a block cipher, and outputs a transformed circuit C'

and corresponding key K'. The circuit $C'[K']$ shall implement the same functionality as C running with key K, but additionally is resilient to certain well-defined classes of leakage. Notice that while the framework of [14] talks about circuits the same approach applies to software implementations, and we only follow this notation to abstract our description.

Moreover, our work uses the well-established simulation paradigm to state the security guarantees we achieve. Intuitively, simulation-based security says that whatever attack an adversary can carry out when knowing the leakage, he can also run (with similar success probability) by just having black-box access to C. In contrast to the approach based on Shannon information theory our analysis includes attacks that exploit joint information from the leakage and plaintext/ciphertext pairs. It seems impossible to us to incorporate the plaintext/ciphertext pairs into an analysis based on Shannon information theory. To see this, consider a block-cipher execution, where, clearly, when given a couple of plaintext/ciphertext pairs, the secret key is information theoretically revealed.[4] The authors of [24] are well aware of this problem and explicitly exclude such joint information. A consequence of the simulation-based security analysis is that we require an additional mild assumption on the noise – namely, that it is efficiently computable (see Section 3.1 for more details). While this is a standard assumption made in most works on leakage resilient cryptography, we emphasize that we can easily drop the assumption of efficiently computable noise (and hence considering the same noise model as [24]), when we only want to achieve the weaker security notion considered in [24]. Notice that in this case we are still able to eliminate the limitations 1 & 2 mentioned above.

1.3 Related work

Masking & leakage resilient circuits. A large body of work has proposed various masking schemes and studies their security in different security models (see, e.g., [13, 2, 23, 31, 27]). The already mentioned t-threshold-probing model has been considered in the work of Rivain and Prouff [27], who show how to extend the work of Ishai et al. to larger fields and propose efficiency improvements. In [25] it was shown that techniques from multiparty computation can be used to show security in the t-threshold-probing model. The work of Standaert et al. [31] studies masking schemes using the information theoretic framework of [29] by considering the Hamming weight model. Many other works analyze the security of the masking countermeasure and we refer the reader for further details to [24].

With the emerge of leakage resilient cryptography [20, 1, 9] several works have proposed new security models and alternative masking schemes. The main difference between these new security models and the t-threshold-probing model is that they consider *joint leakages* from large parts of the computation. The work of Faust et al. [10] extends the security analysis of Ishai et al. beyond the t-threshold-probing model by considering leakages that can be described by low-depth circuits (so-called AC^0 leakages). Faust et al. use leak-free component that have been eliminated by Rohtblum in [28] using computational assumptions. The recent work of Miles and Viola [21] proposes a new circuit transformation using alternating groups and shows security with respect to AC^0 and TC^0 leakages.

[4] More concretely: imagine an adversary that attacks a block-cipher implementation E_K, where K is the secret key. Then just by launching a known-plaintext attack he can obtain several pairs $V = (M_0, E_K(M_0)), (M_1, E_K(M_1)), \ldots$. Clearly a small number of such pairs is usually enough to determine K *information-theoretically*. Hence it makes no sense to require that "K is information-theoretically hidden given V and the side-channel leakage."

Another line of work considers circuits that are provably secure in the so-called continuous bounded leakage model [15, 11, 8, 12]. In this model, the adversary is allowed to learn arbitrary information from the computation of the circuit as long as the amount of information is bounded. The proposed schemes rely additionally on the assumption of "only computation leaks information" of Micali and Reyzin [20].

Noisy leakage models. The work of Faust et al. [10] also considers circuit compilers for noisy models. Specifically, they propose a construction with security in the binomial noise model, where each value on a wire is flipped independently with probability $p \in (0, 1/2)$. In contrast to the work of [24] and our work the noise model is restricted to binomial noise, but the noise rate is significantly better (constant instead of linear noise). Similar to [24] the work of Faust et al. also uses leak-free components. Besides these works on masking schemes, several works consider noisy leakages for concrete cryptographic schemes [9, 22, 16]. Typically, the noise model considered in these works is significantly stronger than the noise model that is considered for masking schemes. In particular, no strong assumption about the independency of the noise is made.

2 Preliminaries

We start with some standard definitions and lemmas about the statistical distance. If $\mathcal{A}$ is a set then $U \leftarrow \mathcal{A}$ denotes a random variable sampled uniformly from $\mathcal{A}$. Recall that if A and B are random variables over the same set $\mathcal{A}$ then the *statistical distance between A and B* is denoted as $\Delta(A; B)$, and defined as $\Delta(A; B) = \frac{1}{2}\sum_{a\in\mathcal{A}} |\mathbb{P}(A = a) - \mathbb{P}(B = a)| = \sum_{a\in\mathcal{A}} \max\{0, \mathbb{P}(A = a) - \mathbb{P}(B = a)\}$. If $\mathcal{X}, \mathcal{Y}$ are some events then by $\Delta((A|\mathcal{X})\ ;\ (B|\mathcal{Y}))$ we will mean the distance between variables A' and B', distributed according to the conditional distributions $P_{A|\mathcal{X}}$ and $P_{B|\mathcal{Y}}$. If $\mathcal{X}$ is an event of probability 1 then we also write $\Delta(A\ ;\ (B|\mathcal{Y}))$ instead of $\Delta((A|\mathcal{X})\ ;\ (B|\mathcal{Y}))$. If C is a random variable then by $\Delta(A\ ;\ (B|C))$ we mean $\sum \mathbb{P}(C = c) \cdot \Delta(A\ ;\ (B|(C = c)))$.

If A, B, and C are random variables then $\Delta((B; C) \mid A)$ denotes $\Delta((BA); (CA))$. It is easy to see that it is equal to $\sum_a \mathbb{P}(A = a) \cdot \Delta((B|A = a)\ ;\ (C|A = a))$. If $\Delta(A; B) \leq \epsilon$ then we say that A and B are *ϵ-close*. The "$\stackrel{d}{=}$" symbol denotes the equality of distributions, i.e., $A \stackrel{d}{=} B$ if and only if $\Delta(A; B) = 0$. We also have the following lemma, whose proof appears in Appendix B.

Lemma 1. *Let A, B be two (possibly correlated) random variables. Let B' be a variable distributed identically to B but independent from A. We have*

$$\Delta(A; (A|B)) = \Delta((B; B') \mid A). \tag{1}$$

3 Noise from set elements

We start with describing the basic framework for reasoning about the noise from elements of a finite set $\mathcal{X}$. Later, in Section 4, we will consider the leakage from the vectors over $\mathcal{X}$, and then, in Section 5, from the entire computation. The reason why we can smoothly use the analysis from Section 3.1 in the later sections is that, as in the work of Prouff and Rivain, we require that the noise is independent for all elementary operations. By elementary operations, [24] considers the basic underlying operations over the underlying field $\mathcal{X}$ used in

a masked implementation. In this work, we consider the same setting and type of underlying operations (in fact, notice that our construction is identical to theirs – except that we eliminate the leak-free gates and prove a stronger statement). Notice that instead of talking about elementary operations, we consider the more standard term of "gates" that was used in the work of Ishai et al. [14].

3.1 Modeling noise

Let us start with a discussion defining what it means that a randomized function $Noise : \mathcal{X} \to \mathcal{Y}$ is "noisy". We will assume that $\mathcal{X}$ is finite and rather small: typical choices for $\mathcal{X}$ would be GF(2) (the "Boolean case"), or GF(2^8), if we want to deal with the AES circuit. The set $\mathcal{Y}$ corresponds to the set of all possible noise measurements and may be infinite, except when we require the "efficient simulation" (we discuss it further at the end of this section). As already informally described in Section 1.1 our basic definition is as follows: we say that the function *Noise* is δ-*noisy* if

$$\delta = \Delta(X; (X|Noise(X)). \tag{2}$$

Of course for (2) to be well-defined we need to specify the distribution of X. The idea to define noisy functions by comparing the distributions of X and "X conditioned on $Noise(X)$" comes from [24], where it is argued that the most natural choice for X is a random variable distributed uniformly over $\mathcal{X}$. We also adopt this convention and assume that $X \leftarrow \mathcal{X}$. We would like to stress, however, that in our proofs we will apply *Noise* to inputs $\hat{X}$ that are not necessarily uniform and in this case the value of $\Delta(\hat{X}; (\hat{X}|Noise(\hat{X}))$ may obviously be some non-trivial function of δ. Of course if $X \leftarrow \mathcal{X}$ and $X' \leftarrow \mathcal{X}$ then $Noise(X')$ is distributed identically to $Noise(X)$, and hence, by Lemma 1, Eq. (2) is equivalent to:

$$\delta = \Delta((Noise(X); Noise(X')) \mid X), \tag{3}$$

where X and X' are uniform over $\mathcal{X}$. Note that at the beginning this definition may be a bit counter-intuitive, as *smaller* δ means *more* noise: in particular we achieve "full noise" if $\delta = 0$, and "no noise" if $\delta \approx 1$. Let us compare this definition with the definition of [24]. In a nutshell: the definition of [24] is similar to ours, the only difference being that instead of the statistical distance Δ in [24] the authors use a distance based on the Euclidean norm. More precisely, they start with defining d as: $\mathrm{d}(X;Y) := \sqrt{\sum_{x\in\mathcal{X}}(\mathbb{P}(X=x) - \mathbb{P}(Y=y))^2}$, and using this notion they define β as:

$$\beta(X|Noise(X)) := \sum_{y\in\mathcal{Y}} \mathbb{P}(Noise(X)=y) \cdot \mathrm{d}(X\ ;\ (X|Noise(X)=y))$$

(where X is uniform). In the terminology of [24] a function *Noise* is "δ-noisy" if $\delta = \beta(X|Noise(X))$. Observe that the right hand side of our noise definition in Eq. (2) can be rewritten as:

$$\sum_{b\in\mathcal{Y}} \mathbb{P}(Noise(X)=y) \cdot \Delta(X\ ;\ (X|Noise(X)=y)),$$

hence the only difference between their approach and ours is that we use Δ where they use the distance d. The authors do not explain why they choose this particular measure. We believe that our choice to use the standard definition of statistical distance Δ is more natural in this setting, since, unlike the "d" distance, it has been used in hundreds of cryptographic papers

in the past. The popularity of the Δ distance comes from the fact that it corresponds to an intuitive concept of the "indistinguishability of distributions" — it is well-known, and simple to verify, that $\Delta(X;Y) \leq \delta$ if and only if no adversary can distinguish between X and Y with advantage better than δ.[5] Hence, e.g., (3) can be interpreted as:

> δ is the maximum probability, over all adversaries $\mathcal{A}$, that $\mathcal{A}$ distinguishes between the noise from a uniform X that is *known to him*, and a uniform X' that is *unknown to him*.

It is unclear to us if a d distance has a similar interpretation. We emphasize, however, that the choice whether to use Δ or β is not too important, as the following inequalities between these measures hold for every X and Y distributed over $\mathcal{X}$ (cf. [24]):

$$\frac{1}{2} \cdot \mathrm{d}(X;Y) \leq \Delta(X;Y) \leq \frac{\sqrt{|\mathcal{X}|}}{2} \cdot \mathrm{d}(X;Y),$$

and consequently

$$\frac{1}{2} \cdot \beta(X|Noise(X)) \leq \Delta(X;(X|Noise(X)) \leq \frac{\sqrt{|\mathcal{X}|}}{2} \cdot \beta(X|Noise(X)). \tag{4}$$

Hence, we decide to stick to the "Δ distance" in this paper. However, to allow for comparison between our work and the one of [24] we will at the end of the paper present our results also in terms of the β measure (this translation will be straightforward, thanks to the inequalities in (4)).In [24] (cf. Theorem 4) the result is stated in form of Shannon information theory. While such an information theoretic approach may be useful in certain settings [29], we follow the more "traditional" approach and provide an efficient simulation argument. As discussed in the introduction, this also covers a setting where the adversary exploits joint information of the leakage and, e.g., the plaintext/ciphertext pairs. We emphasize, however, that our results can easily be expressed in the information theoretic language, thanks to the following bound: $I(A;B) \leq (2N/\ln 2) \cdot \Delta(A;(A|B))$, where A is uniformly distributed over a set of cardinality N. This result comes from Proposition 1 in [24] combined with the inequality $\beta(A|B) \leq 2\Delta(A;(A|B))$, cf. (4).

The issue of "efficient simulation" To achieve the strong simulation-based security notion, we need an additional requirement on the leakage, namely, that the leakage can efficiently be "simulated" – which typically requires that the noise function is efficiently computable. In fact, for our proofs to go through we actually need something slightly stronger, namely that *Noise is efficiently decidable* by which we mean that (a) there exists a randomized poly-time algorithm that computes it, and (b) the set $\mathcal{Y}$ is finite and for every x and y the value of $\mathbb{P}\left(Noise(x) = y\right)$ is computable in polynomial time. While (b) may look like a strong assumption we note that in practice for most "natural" noise functions (like the Gaussian noise with a known parameter, measured with a very good, but finite, precision) it is easily satisfiable.

Recall that the results of [24] are stated without taking into consideration the issue of the "efficient simulation". Hence, if one wants to compare our results with [24] then one can simply drop the efficient decidability assumption on the noise. To keep our presentation concise and clean, also in this case the results will be presented in a form "for every adversary $\mathcal{A}$ there exists an (inefficient) simulator $\mathcal{S}$". Here the "inefficient simulator" can be an arbitrary machine, capable, e.g., of sampling elements from *any* probability distributions.

[5] This formally means that for every $\mathcal{A}$ we have $|\mathbb{P}\left(\mathcal{A}(X) = 1\right) - \mathbb{P}\left(\mathcal{A}(Y) = 1\right)| \leq \delta$.

3.2 Simulating noise by ϵ-identity functions

Lemma 2 below is our main technical tool. Informally, it states that every δ-noisy function $Noise : \mathcal{X} \to \mathcal{Y}$ can be represented as a composition $Noise' \circ \varphi$ of efficiently computable randomized functions $Noise'$ and φ, where φ is a "$\delta \cdot |\mathcal{X}|$-identity function", defined in Definition 1 below.

Definition 1. *A randomized function $\varphi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ is an* ϵ-identity *if for every x we have that either $\varphi(x) = x$ or $\varphi(x) = \perp$ and $\mathbb{P}(\varphi(x) \neq \perp) = \epsilon$.*

This will allow us to reduce the "noisy attacks" to the "random probing attacks", where the adversary learns each wire (or a gate, see Section 5.5) of the circuit with probability ϵ. Observe also, that thanks to the assumed independence of noise, the events that the adversary learns each element are independent, which, in turn, will allow us to use the Chernoff bound to prove that with a good probability the number of wires that the adversary learns is small.

Lemma 2. *Let $Noise : \mathcal{X} \to \mathcal{Y}$ be a δ-noisy function. Then there exist $\epsilon \leq \delta \cdot |\mathcal{X}|$ and a randomized function $Noise' : \mathcal{X} \cup \{\perp\} \to \mathcal{X}$ such that for every $x \in \mathcal{X}$ we have*

$$Noise(x) \stackrel{d}{=} Noise'(\varphi(x)), \tag{5}$$

where $\varphi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ is the ϵ-identity function. Moreover, if Noise is efficiently decidable then $Noise'(\varphi(x))$ is computable in time that is expected polynomial in $|\mathcal{X}|$.

The proof appears in Appendix C.

4 Leakage from vectors

In this section we describe the leakage models relevant to this paper. We start with describing the models abstractly, by considering leakage from an arbitrary sequence $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$, where $\mathcal{X}$ is some finite set and ℓ is a parameter. The adversary $\mathcal{A}$ will be able to obtain some partial information about $(x_1, \ldots, x_\ell)$ via the games described below. Note that we do not specify the computational power of $\mathcal{A}$, as the definitions below make sense for both computationally-bounded or infinitely powerful $\mathcal{A}$.

Noisy model. For $\delta \geq 0$ a *δ-noisy adversary on $\mathcal{X}^\ell$* is a machine $\mathcal{A}$ that plays the following game against an oracle that knows $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$:

1. $\mathcal{A}$ specifies a sequence $\{Noise_i : \mathcal{X} \to \mathcal{Y}\}_{i=1}^{\ell}$ of noisy functions such that every $Noise_i$ is δ_i'-noisy, for some $\delta_i' \leq \delta$ and mutually independent noises.
2. $\mathcal{A}$ receives $Noise_1(x_1), \ldots, Noise_\ell(x_\ell)$ and outputs some value $out_{\mathcal{A}}(x_1, \ldots, x_\ell)$.

If $\mathcal{A}$ works in polynomial time and the noise functions specified by $\mathcal{A}$ are efficiently decidable then we say that $\mathcal{A}$ *is poly-time-noisy.*

Random probing model. For $\epsilon \leq 0$ a *ϵ-random-probing adversary on $\mathcal{X}^\ell$* is a machine $\mathcal{A}$ that plays the following game against an oracle that knows $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$:

1. $\mathcal{A}$ specifies a sequence $(\epsilon_1, \ldots, \epsilon_\ell)$ such that each $\epsilon_i \leq \epsilon$.
2. $\mathcal{A}$ receives $\varphi_1(x_1), \ldots, \varphi_\ell(x_\ell)$ and outputs some value $out_{\mathcal{A}}(x_1, \ldots, x_\ell)$, where each φ_i is the ϵ_i-identity function with mutually independent randomness.

A similar model was introduced in the work of Ishai, Sahai and Wagner [14] to obtain circuit compilers with linear blow-up in the size.

Threshold probing model. For $t = 0, \ldots, \ell$ a *t-treshold-probing adversary on* $\mathcal{X}^\ell$ is a machine $\mathcal{A}$ that plays the following game against an oracle that knows $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$:

1. $\mathcal{A}$ specifies a set $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\} \subseteq \{1, \ldots, \ell\}$ of cardinality at most t,
2. $\mathcal{A}$ receives $(x_{i_1}, \ldots, x_{i_{|\mathcal{I}|}})$ and outputs some value $out_{\mathcal{A}}(x_1, \ldots, x_\ell)$.

4.1 Simulating the noisy adversary by a random-probing adversary

The following lemma shows that every δ-noisy adversary can be simulated by a $\delta \cdot |\mathcal{X}|$-random probing adversary.

Lemma 3. *Let $\mathcal{A}$ be a δ-noisy adversary on $\mathcal{X}^\ell$. Then there exists a $\delta \cdot |\mathcal{X}|$-random-probing adversary $\mathcal{S}$ on $\mathcal{X}^\ell$ such that for every $(x_1, \ldots, x_\ell)$ we have*

$$out_{\mathcal{A}}(x_1, \ldots, x_\ell) \stackrel{d}{=} out_{\mathcal{S}}(x_1, \ldots, x_\ell). \tag{6}$$

Moreover, if $\mathcal{A}$ is poly-time-noisy, then $\mathcal{S}$ works in time polynomial in $|\mathcal{X}|$.

Intuitively, this lemma easily follows from Lemma 2 applied independently to each element of $(x_1, \ldots, x_\ell)$. The formal proof appears in Appendix D.

4.2 Simulating the random-probing adversary by a threshold-probing adversary

In this section we show how to simulate every δ-random probing adversary by a threshold adversary. This simulation, unlike the one in Section 4 will not be perfect in the sense that the distribution output by the simulator will be identical to the distribution of the original adversary only when conditioned on some event that happens with a large probability. We start with the following lemma, whose proof, which is a straightforward application of the Chernoff bound, appears in Appendix E.

Lemma 4. *Let $\mathcal{A}$ be an ϵ-random-probing adversary on $\mathcal{X}^\ell$. Then there exists a $(2\epsilon\ell - 1)$-threshold-probing adversary $\mathcal{S}$ on $\mathcal{X}^\ell$ operating in time linear in the working time of $\mathcal{A}$ such that for every $(x_1, \ldots, x_\ell)$ we have*

$$\Delta(out_{\mathcal{A}}(x_1, \ldots, x_\ell) \; ; \; out_{\mathcal{S}}(x_1, \ldots, x_\ell) \mid out_{\mathcal{S}}(x_1, \ldots, x_\ell) \neq \perp) = 0, \tag{7}$$

where

$$\mathbb{P}\left(out_{\mathcal{S}}(x_1, \ldots, x_\ell) = \perp\right) \leq \exp\left(-\frac{\epsilon\ell}{3}\right). \tag{8}$$

The following corollary, whose proof appears in Appendix F, combines Lemma 3 and 4 together, and will be useful in the sequel.

Corollary 1. *Let $d, \ell \in \mathbb{N}$ with $\ell > d$ and let $\mathcal{A}$ be a $d/(4\ell \cdot |\mathcal{X}|)$-noisy adversary on $\mathcal{X}^\ell$. Then there exists an $(d/2 - 1)$-threshold-probing adversary $\mathcal{S}$ such that*

$$\Delta(out_{\mathcal{A}}(x_1, \ldots, x_\ell) \; ; \; out_{\mathcal{S}}(x_1, \ldots, x_\ell) \mid out_{\mathcal{S}}(x_1, \ldots, x_\ell) \neq \perp) = 0 \tag{9}$$

and $\mathbb{P}\left(out_{\mathcal{S}}(x_1, \ldots, x_\ell) = \perp\right) \leq \exp(-d/12)$. Moreover, if $\mathcal{A}$ is poly-time-noisy then $\mathcal{S}$ works in time polynomial $\ell \cdot |\mathcal{X}|$.

5 Leakage from computation

In this section we address the main topic of this paper, which is the noise-resilience of cryptographic computations. Our main model will be the model of arithmetic circuits over a finite field. First, in Section 5.1 we present our security definitions, and then, in Section 5.2 we describe a secure "compiler" that transforms any cryptographic scheme secure in the "black-box" model into one secure against the noisy leakage (it is essentially identical to the transformation of [14] later extended in [27]). Finally, in the last section we present our security results.

5.1 Definitions

A *(stateful arithmetic) circuit Γ over a field* $\mathbb{F}$ is a directed graph whose nodes are called *gates*. Each gate γ can be of one of the following types: an *input gate* γ^{inp} of fan-in zero, an *output gate* γ^{out} of fan-out zero, a *random gate* γ^{rand} of fan-in zero, a *multiplication gate* $\gamma^{\times}$ of fan-in 2, an *addition gate* γ^{+} of fan-in 2, a *subtraction gate* γ^{-} of fan-in 2, a *constant gate* γ^{const}, and a memory gate γ^{mem} of fan-in 1. Following [14] we assume that the fan-out of every gate is at most 3. The only cycles that are allowed in Γ must contain exactly 1 memory gate. The *size* $|\Gamma|$ *of the circuit* Γ is defined to be the total number of its gates. The numbers of input gates, output gates and memory gates will be denoted $|\Gamma.inp|$, $|\Gamma.out|$, and $|\Gamma.mem|$, respectively.

The computation of Γ is performed in several "rounds" numbered $1, 2, \ldots$. In each of them the circuit will take some input, produce an output and update the memory state. Initially, the memory gates of Γ are preloaded with some initial "state" $k_0 \in \mathbb{F}^{|\Gamma.mem|}$. At the beginning of the ith round the input gates are loaded with elements of some vector $a_i \in \mathbb{F}^{|\Gamma.inp|}$ called the *input for the ith round*. The computation of Γ in the ith round depends on a_i and on the memory state k_{i-1}. It proceeds in a straightforward way: if all the input wires of a given gate are known then the value on its output wire can be computed naturally: if γ is a multiplication gate with input wires carrying values a and b, then its output wire will carry the value $a \cdot b$ (where "$\cdot$" is the multiplication operation in $\mathbb{F}$), and the addition and the subtraction gates are handled analogously. We assume that the random gates produce a fresh random field element in each round. The *output of the ith round* is read-off from the output gates and denoted $b_i \in \mathbb{F}^{|\Gamma.out|}$. The *state after the ith round* is contained in the memory gates and denoted k_i. For $k \in \mathbb{F}^{|\Gamma.mem|}$ and a sequence of inputs $(a_1, \ldots, a_m)$ (where each $a_i \in \mathbb{F}^{|\Gamma.inp|}$) let $\Gamma(k, a_1, \ldots, a_m)$ denote the sequence $(B_1, \ldots, B_m)$ where each B_i is the output of Γ with $k_0 = k$ and inputs $a_1, \ldots, a_m$ in rounds $1, 2, \ldots$. Observe that, since Γ is randomized, hence $\Gamma(k, a_1, \ldots, a_m)$ is a random variable.

A *black-box circuit adversary* $\mathcal{A}$ is a machine that adaptively interacts with a circuit Γ via the input and output interface. Then $out\left(\mathcal{A} \overset{bb}{\leftrightarrows} \Gamma(k)\right)$ denotes the output of $\mathcal{A}$ after interacting with Γ whose initial memory state is $k_0 = k$. A *δ-noisy circuit adversary* $\mathcal{A}$ is an adversary that has the following additional ability: after each ith round $\mathcal{A}$ gets some partial information about the internal state of the computation via the noisy leakage functions. More precisely: let $(X_1, \ldots, X_\ell)$ be the random variable denoting the values on the wires of $\Gamma(k)$ in the ith round. Then $\mathcal{A}$ plays the role of a δ-noisy adversary in a game against $(X_1, \ldots, X_\ell)$ (c.f. Section 4), namely: he choses a sequence $\{Noise_i : \mathbb{F} \to \mathcal{Y}\}_{i=1}^{\ell}$ of functions such that every $Noise_i$ is δ_i-noisy for some $\delta_i \leq \delta$ and he receives $Noise_1(X_1), \ldots, Noise_\ell(X_\ell)$. Let

$out\left(\mathcal{A} \overset{noisy}{\leftrightarrows} \Gamma(k)\right)$ denote the output of such an $\mathcal{A}$ after interacting with Γ whose initial memory state is $k_0 = k$.

We can also replace, in the above definition, the "δ-noisy adversary" with the "ϵ-random probing adversary". In this case, after each ith round $\mathcal{A}$ choses a sequence $(\epsilon_1, \ldots, \epsilon_\ell)$ such that each $\epsilon_i \leq \epsilon$ and he learns $\varphi_1(X_1), \ldots, \varphi_\ell(X_\ell)$, where each φ_i is the ϵ_i-identity function. Let $out\left(\mathcal{A} \overset{rnd}{\leftrightarrows} \Gamma(k)\right)$ denote the output of such $\mathcal{A}$ after interacting with Γ whose initial memory state is $k_0 = k$.

Analogously we can replace the "δ-noisy adversary" with the "t-threshold probing adversary" obtaining an adversary that after each ith round $\mathcal{A}$ learns t elements of $(X_1), \ldots, \varphi_\ell(X_\ell)$. Let $out\left(\mathcal{A} \overset{thr}{\leftrightarrows} \Gamma(k)\right)$ denote the output of such $\mathcal{A}$ after interacting with Γ whose initial memory state is $k_0 = k$.

Definition 2. *Consider two stateful circuits Γ and Γ' (over some field $\mathbb{F}$) and a randomized encoding function Enc. We say that Γ' is a* (δ, ξ)-noise resilient implementation of a circuit Γ w.r.t. *Enc if the following holds for every $k \in \mathbb{F}^{|\Gamma.inp|}$:*

1. *the input-output behavior of $\Gamma(k)$ and $\Gamma'(Enc(k))$ is identical, i.e.: for every sequence of inputs $a_1, \ldots, a_m$ and outputs $b_1, \ldots, b_m$ we have*

$$\mathbb{P}\left(\Gamma(k, a_1, \ldots, a_m) = (b_1, \ldots, b_m)\right) = \mathbb{P}\left(\Gamma'(Enc(k), a_1, \ldots, a_m) = (b_1, \ldots, b_m)\right)$$

   and
2. *for every δ-noisy circuit adversary $\mathcal{A}$ there exists a black-box circuit adversary $\mathcal{S}$ such that*

$$\Delta\left(out\left(\mathcal{S} \overset{bb}{\leftrightarrows} \Gamma(k)\right) \;;\; out\left(\mathcal{A} \overset{noisy}{\leftrightarrows} \Gamma'(Enc(k))\right)\right) \leq \xi. \tag{10}$$

The definition of Γ' being a (ϵ, ξ)-random-probing resilient implementation of a circuit Γ *is identical to the one above, except that Point 2 is replaced with:*

2'. for every ϵ-random-probing circuit adversary $\mathcal{A}$ there exists a black-box circuit adversary $\mathcal{S}$ such that

$$\Delta\left(out\left(\mathcal{S} \overset{bb}{\leftrightarrows} \Gamma(k)\right) \;;\; out\left(\mathcal{A} \overset{rnd}{\leftrightarrows} \Gamma'(Enc(k))\right)\right) \leq \xi.$$

The definition of Γ' being a (t, ξ)-threshold-probing resilient implementation of a circuit Γ *is identical to the one above, except that Point 2 is replaced with:*

2". for every t-threshold-probing circuit adversary $\mathcal{A}$ there exists a black-box circuit adversary $\mathcal{S}$ such that

$$\Delta\left(out\left(\mathcal{S} \overset{bb}{\leftrightarrows} \Gamma(k)\right) \;;\; out\left(\mathcal{A} \overset{thr}{\leftrightarrows} \Gamma'(Enc(k))\right)\right) \leq \xi.$$

In all cases above we will say that Γ' is a an implementation Γ with efficient simulation *if the simulator $\mathcal{S}$ works in time polynomial in $\Gamma' \cdot |\mathbb{F}|$ as long as $\mathcal{A}$ is poly-time and the noise functions specified by $\mathcal{A}$ are efficiently decidable.*

5.2 The implementation

In this section we describe the circuit compiler of [14], generalized to larger fields in [27]. Let Γ be a stateful arithmetic circuit and let $d \in \mathbb{N}$ be a parameter. The encoding function Enc_+ that we use is also standard and is often called the "additive masking". It is defined as: $Enc_+(x) := (X_1, \ldots, X_d)$, where $X_1, \ldots, X_d$ are uniform such that $X_1 + \cdots + X_d = x$.

At a high level, each wire w in the original circuit Γ is represented by a *wire bundle* in Γ', consisting of d wires $\overrightarrow{w} = (w_1, \ldots, w_d)$, that carry an *encoding* of w. The gates in C are replaced gate-by-gate with so-called *gadgets*, computing on encoded values. The main difficulty is to construct gadgets that remain "secure" even if their internals may leak.

Because the transformed gadgets in Γ' operate on encodings, Γ' needs to have a subcircuit at the beginning that encodes the inputs and another subcircuit at the end that decodes the outputs. We will deal with the output decoding later. The input encoding is easy to implement for our encoding function Enc_+: to encode an input x one simply uses the random gates to generate $d-1$ field elements $x_1, \ldots, x_{d-1}$ and then computes x_d as $x_1 + \cdots + x_{d-1} - x$. Clearly this can be done using d addition and subtraction gates. Recall that the memory gates of Γ are assumed to be preloaded with field elements that already encode k using the encoding Enc_+ (cf. (10)), hence there is no need to encode k.

Each constant gate γ_c^{const} in Γ can be transformed into d constant gates in Γ', the first of them being γ_c^{const} and the remaining ones being γ_0^{const}. This is trivially correct as $c = c + 0 + \cdots + 0$. Every random gate γ^{rand} in Γ is transformed into d random gates in Γ'. This works since, clearly, a uniformly random encoding $(X_1, \ldots, X_d)$ encodes a uniformly random element of $\mathbb{F}$.

What remains to show is how the operation (addition, subtraction, and multiplication) gates are handled. Consider a gate γ in Γ. Let a and b be its input wires and let $\overrightarrow{a} = (a_1, \ldots, a_d)$ and $\overrightarrow{b} = (b_1, \ldots, b_d)$ be their corresponding wire bundles in Γ'. Let the output wire bundle in Γ' will be $(c_1, \ldots, c_d)$. The cases when γ is an addition or subtraction gate are actually easy to deal with, thanks to the linearity of the encoding function. For example, if γ is an addition gate γ^+ then each c_i can be computed using an addition gate γ^+ in Γ' with input wires a_i and b_i (this is obviously correct as $(a_1 + b_1) + \cdots + (a_d + b_d) = (a_1 + \cdots a_d) + (b_1 + \cdots + b_d)$). The subtraction is handled analogously. The only tricky case is when γ is the multiplication gate. In this case the circuit Γ' generates, for every $1 \leq i < j \leq d$, a random field element $z_{i,j}$ (this is done using the random gates in Γ'). Then, for every $1 \leq j < i \leq d$ it computes $z_{i,j} := a_i b_j + a_j b_i - z_{j,i}$, and finally he computes each c_i (for $i = 1, \ldots, d$) as $c_i := a_i b_i + \sum_{i \neq j} z_{i,j}$. To see why this computation is correct consider the sum $c = c_1 + \cdots + c_d$ and observe that every $z_{i,j}$ in it appears exactly once with plus sign and once with a minus sign, and hence it cancels out. Moreover each term $a_i b_j$ appears in the formula for c exactly once. Hence c is equal to $\sum_{i,j \in \{1,\ldots,n\}} a_i b_j = \left(\sum_{i=1}^{d} a_i\right)\left(\sum_{j=1}^{d} b_j\right) = ab$. It is straightforward to verify that the total number of gates in this gadget is $3.5 \cdot d^2$. This finishes the description of the compiler.

The multiplication gadget above turns out to be useful as a building block for "refreshing" of the encoding. More concretely, suppose we have a wire bundle $\overrightarrow{a} = (a_1, \ldots, a_d)$ and we wish to obtain another bundle $\overrightarrow{b} = (b_1, \ldots, b_d)$ such that $\overrightarrow{b}$ is a fresh encoding of $Dec_+(\overrightarrow{a})$. This can be achieved by a *Refresh* sub-gadget constructed as follows. First, create an encoding $(1, 0, \ldots, 0)$ of 1 (using d constant gates), and multiply $(1, 0, \ldots, 0)$ and $\overrightarrow{a}$ together using the multiplication protocol above. Since $(1, 0, \ldots, 0)$ is an encoding of 1, hence the result will

be an encoding of $1 \cdot a = a$. The multiplication can be done with $3.5 \cdot d^2$ gates, and hence altogether this gadget uses $3.5 \cdot d^2 + 2 \cdot d$ gates.

We can now use the *Refresh* sub-gadget to construct the output gadgets in Γ'. Let γ^{out} be an output gate in Γ with an input wire a. Then in Γ' it is transformed into the following: let $\overrightarrow{a}$ be the wire bundle corresponding to a. First apply the *Refresh* sub-gadget, and then calculate the sum $b_1 + \cdots + b_d$ (where $(b_1, \ldots, b_d)$ is the output of *Refresh*) and output the result.

The refreshing gadget is also useful to provide security of the memory encoding in the multi-round scenario. More precisely, we assume that every memory state gets refreshed at the end of each round by the *Refresh* procedure. It is easy to see that without this "refreshing" the contents of the memory would eventually leak completely to the adversary even if he probes a very limited number (say: 1) of wires in each round. For more details see [14].

5.3 Security in the probing model [14]

In [14] it is shown that the compiler from the pervious section is secure against probing attacks in which the adversary can probe at most $\lfloor (d-1)/2 \rfloor$ wires in each round.[6] This parameter may be a bit disappointing as the number of probes that the adversary needs to break the security does not grow with the size of the circuit. This assumption may seem particularity unrealistic for large circuits Γ. Fortunately, [14] also shows a small modification of the construction from Section 5.2 that is resilient to a larger number of probes, provided that the number of probes from each gadget is bounded. Before we present it let us argue why the original construction is not secure against such attacks. To this end, assume that our circuit Γ has a long sequence of wires $a_1, \ldots, a_m$, where each a_i (for $i > 1$) is the result of adding to a_{i-1} (using an addition gate) a 0 constant (that was generated using a γ_0^{const} gate). It is easy to see that in the circuit Γ' all the wire bundles $\overrightarrow{a_1}, \ldots, \overrightarrow{a_m}$ (where each $\overrightarrow{a_i}$ corresponds to a_i) will be identical. Hence, the adversary that probes even a single wire in each addition gadget in Γ' will learn the encoding of a_1 completely as long as $m \geq d$. Fortunately one can deal with this problem by "refreshing" the encoding after each subtraction and addition gate exactly in the same way as done before, i.e. by using the *Refresh* sub-gadget.

Lemma 5 ([14]). *Let Γ be an arbitrary stateful arithmetic circuit over some field $\mathbb{F}$.Let Γ' be the circuit that results from the procedure described above. Then Γ' is a $(\lfloor (d-1)/2 \rfloor \cdot |\Gamma|, 0)$-threshold-probing resilient implementation of a circuit Γ (with efficient simulation), provided that the adversary does not probe each gadget more than $\lfloor (d-1)/2 \rfloor$ times in each round.*

We notice that [14] also contains a second transformation with blow-up $\tilde{O}(d\,|\Gamma|)$. It may be possible that this transformation can provide better noise parameters as is achieved by Theorem 2. However, due to the hidden parameters in the $\tilde{O}$-notation we do not get a straightforward improvement of our result. In particular, using this transformation the size of the transformed circuit depends also on an additional statistical security parameter, which will affect the tolerated noise level.

[6] Strictly speaking the proof of [14] considers only the case when $\mathbb{F} = \mathrm{GF}(2)$. It was observed in [27] that it can be extended to any finite field, as the only properties of GF(2) that are used in the proof are the field axioms.

5.4 Resilience to noisy leakage from the wires

We now show that the construction from Section 5.3 is secure against the noisy leakage. More precisely, we show the following.

Theorem 1. *Let Γ be an arbitrary stateful arithmetic circuit over some field $\mathbb{F}$. Let Γ' be the circuit that results from the procedure described in Section 5.3. Then Γ' is a $(\delta, |\Gamma| \cdot \exp(-d/12))$-noise-resilient implementation of Γ (with efficient simulation), where*

$$\delta := ((28d + 16)\,|\mathbb{F}|)^{-1} = O(1/(d \cdot |\mathbb{F}|)).$$

This lemma is proven by combining Corollary 1 that reduces the noisy adversary to the probing adversary, with Lemma 5 that shows that the construction from Section 5.3 is secure against probing. The full proof appears in Appendix G.

5.5 Resilience to noisy leakage from the gates

The model of Prouff and Rivain is actually slightly different than the one considered in the previous section. The difference is that they assume that the noise is generated by the *gates*, not by the *wires*. This can be formalized by assuming that each noise function *Noise* is applied to the "contents of a gate". We do not need to specify exactly what we mean by this. It is enough to observe that the contents of each gate γ can be described by at most 2 field elements: obviously if γ is a random gate, output gate, or memory gate then its entire state in a given round can be described by one field element, and if γ is an operation gate then it can be described by two field elements that correspond to γ's input. Hence, without loss of generality we can assume that the noise function is defined over the domain $\mathbb{F} \times \mathbb{F}$.

Formally, we define a *δ-gate-noisy circuit adversary* $\mathcal{A}$ as a machine that, besides of having black box access to a circuit $\Gamma(k)$, can, after each ith round, get some partial information about the internal state of the computation via the δ-noisy leakage functions applied to the gates (in a model described above). Let $out\left(\mathcal{A} \overset{g\text{-}noisy}{\leftrightarrows} \Gamma(k)\right)$ denote the output of such $\mathcal{A}$ after interacting with Γ whose initial memory state is $k_0 = k$.

We can accordingly modify the definition of noise-resilient circuit implementations (cf. Definition 2). We say that Γ' is a (δ, ξ)-*input-gate-noise resilient implementation of a circuit Γ w.r.t. Enc* if for every k and every δ-noisy circuit adversary $\mathcal{A}$ described above there exists a black-box circuit adversary $\mathcal{S}$ working in time polynomial in $\Gamma' \cdot |\mathbb{F}|$ such that

$$\Delta\left(out\left(\mathcal{S} \overset{bb}{\leftrightarrows} \Gamma(k)\right)\ ;\ out\left(\mathcal{A} \overset{g\text{-}noisy}{\leftrightarrows} \Gamma'(Enc(k))\right)\right) \leq \xi. \tag{11}$$

It turns out that the transformation from Section 5.3 also works in this model, although with different parameters. More precisely we have the following theorem, whose proof appears in Appendix H.[7]

[7] Note that our result holds only when the number of shares is large. For small values of d (e.g., $d = 2, 3, 4$) like those considered in [31], our result does not give meaningful bounds. This is similar to the work of Prouff and Rivain [24] and it is an interesting open research question to develop security models that work for small security parameters.

Theorem 2. *Let Γ be an arbitrary stateful arithmetic circuit over some field $\mathbb{F}$. Let Γ' be the circuit that results from the procedure described in Section 5.3. Then Γ' is a $(\delta, |\Gamma| \cdot \exp(-d/24))$-noise-resilient implementation of Γ (with efficient simulation), where*

$$\delta := \left((28d + 16) \cdot |\mathbb{F}|^2\right)^{-1} = O(1/(d \cdot |\mathbb{F}|^2)). \tag{12}$$

Comparison with [24]. As described in the introduction, our main advantage over [24] is the removal of the assumption about the existence of the leak-free gates, a stronger security model — chosen message attack, instead of a random message attack, and a more meaningful security statement. Still, it is interesting to compare our noise parameters with the parameters of [24]. Let us analyze how much noise is needed by [24] to ensure that the adversary obtains exponentially small information from leakage. The reader should keep in mind that both in our paper, and in [24] "more noise" means that a certain quantity, δ, in our case, is *smaller*. Hence, the larger δ is, the stronger the result becomes (as it means that *less* noise is required for the security to hold).

The main result of [24] is Theorem 4 on page 154. Unfortunately, the statement of this theorem is asymptotic treating $|\mathbb{F}|$ as constant, and hence to get a precise bound on how much noise is required one needs to inspect the proof. The bound on the noise can be deduced from the part of the proof entitled "Security of Type 3 Subsequences", where the required noise is inversely-proportional to "$\lambda(d)$", and this last value is linear in $d \cdot |\mathbb{F}|^3$ (note that $|\mathbb{F}|$ is denoted by N in [24], and d is a security parameter identical to ours). Hence their δ is $O(1/(d \cdot |\mathbb{F}|^3))$. Our Lemma 2 requires a more liberal bound (cf. (12)), and hence can be viewed as stronger, however, as explained in Section 3.1, the notion of distance in [24] is slightly different than the standard "statistical distance" that we use. Fortunately, one can use Eq. (4) to translate our bound into their language. It turns out that in this case our and their bounds are asymptotically identical ($O(1/(d \cdot |\mathbb{F}|^3))$). This is shown in Corollary 2 below, whose proof appears in Appendix I. Note that this translation is unidirectional, in the sense that their "$O(1/(d \cdot |\mathbb{F}|^3))$" bound does *not* imply a bound "$O(1/(d \cdot |\mathbb{F}|^2))$" in our sense.

Corollary 2. *Let Γ be an arbitrary stateful arithmetic circuit over some field $\mathbb{F}$. Let Γ' be the circuit that results from the procedure described in Section 5.3. Then Γ' is a $(\delta', |\Gamma| \cdot \exp(-d/24))$-noise-resilient implementation of Γ (with efficient simulation)* when the noise is defined using the β distance*, where*

$$\delta' = \left((14d + 8) \cdot |\mathbb{F}|^3\right)^{-1} = O(1/(d \cdot |\mathbb{F}|^3)).$$

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In *TCC*, pages 474–495, 2009.
2. Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably Secure Masking of AES. In *Selected Areas in Cryptography*, pages 69–83, 2004.
3. Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-Order Masking Schemes for S-Boxes. In *FSE*, pages 366–384, 2012.
4. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO*, pages 398–412, 1999.
5. Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *CHES*, pages 252–263, 2000.

6. Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and Improvement of the Random Delay Countermeasure of CHES 2009. In *CHES*, pages 95–109, 2010.
7. Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, 2009.
8. Stefan Dziembowski and Sebastian Faust. Leakage-Resilient Circuits without Computational Assumptions. In *TCC*, pages 230–247, 2012.
9. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In *FOCS*, pages 293–302, 2008.
10. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In *EUROCRYPT*, pages 135–156, 2010.
11. Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *CRYPTO*, pages 59–79, 2010.
12. Shafi Goldwasser and Guy N. Rothblum. How to Compute in the Presence of Leakage. In *FOCS*, pages 31–40, 2012.
13. Louis Goubin and Jacques Patarin. DES and Differential Power Analysis (The "Duplication" Method). In *CHES*, pages 158–172, 1999.
14. Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, pages 463–481, 2003.
15. Ali Juma and Yevgeniy Vahlis. Protecting Cryptographic Keys against Continual Leakage. In *CRYPTO*, pages 41–58, 2010.
16. Jonathan Katz and Vinod Vaikuntanathan. Signature Schemes with Bounded Leakage Resilience. In *ASIACRYPT*, pages 703–720, 2009.
17. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO'96*, pages 104–113, 1996.
18. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO'99*, pages 388–397, 1999.
19. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
20. Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In *TCC*, pages 278–296, 2004.
21. Eric Miles and Emanuele Viola. Shielding circuits with groups. In *STOC*, pages 251–260, 2013.
22. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
23. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *FSE*, pages 413–423, 2005.
24. Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.
25. Emmanuel Prouff and Thomas Roche. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In *CHES*, pages 63–78, 2011.
26. Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *E-smart*, pages 200–210, 2001.
27. Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In *CHES*, pages 413–427, 2010.
28. Guy N. Rothblum. How to Compute under AC0 Leakage without Secure Hardware. In *CRYPTO*, pages 552–569, 2012.
29. François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT*, pages 443–461, 2009.
30. François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions. In *CRYPTO (1)*, pages 335–352, 2013.
31. François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The World Is Not Enough: Another Look on Second-Order DPA. In *ASIACRYPT*, pages 112–129, 2010.
32. Nicolas Veyrat-Charvillon and François-Xavier Standaert. Adaptive Chosen-Message Side-Channel Attacks. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 186–199, 2010.

A Basic probability-theoretic facts

Here, we state some basic lemmas that will be used later in the appendix.

Lemma 6. *For any random variables A and B and an event $\mathcal{E}$ we have*

$$\Delta((A;B)|\neg\mathcal{E}) \leq \Delta(A;B) + \mathbb{P}(\mathcal{E}),$$

where $\neg\mathcal{E}$ denotes the negation of $\mathcal{E}$.

Lemma 7 (Chernoff bound, see, e.g, [7], Theorem 1.1). *Let $Z = \sum_{i=1}^{n} Z_i$, where Z_i's are random variables independently distributed over $[0,1]$. Then for every $\xi > 0$ we have*

$$\mathbb{P}(Z \geq (1+\xi)\mathbb{E}(Z)) \leq \exp\left(-\frac{\xi^2}{3}\mathbb{E}(Z)\right)$$

B Proof of Lemma 1

We have

$$\begin{aligned}
&\Delta(A;(A|B)) \\
&= \sum_b \frac{1}{2} \cdot \mathbb{P}(B=b) \cdot \sum_a |\mathbb{P}(A=a) - \mathbb{P}(A=a|B=b)| \\
&= \frac{1}{2} \sum_{a,b} |\mathbb{P}(B=b)\cdot\mathbb{P}(A=a) - \mathbb{P}(B=b)\cdot\mathbb{P}(A=a|B=b)| \\
&= \frac{1}{2} \sum_{a,b} \left|\mathbb{P}\left(B'=b \wedge A=a\right) - \mathbb{P}(B=b\wedge A=a)\right| \qquad (13) \\
&= \Delta((B;B') \mid A),
\end{aligned}$$

where in (13) we used the fact that B' is a variable distributed identically to B and it is independent from A. □

C Proof of Lemma 2

We consider only the case when *Noise* is efficiently decidable, and hence the *Noise*$'$ function that we construct will be efficiently computable. The case when *Noise* is not efficiently decidable is handled in an analogous way (the proof is actually simpler as the only difference is that we do not need to argue about the efficiency of the sampling algorithms). Let X and X' be uniform over $\mathcal{X}$. For every $y \in \mathcal{Y}$ define

$$\pi(y) = \min_{x\in\mathcal{X}}(\mathbb{P}(Noise(x) = y)).$$

Clearly π is computable in time polynomial in $|\mathcal{X}|$. Obviously π is usually not a probability distribution since it does not sum up to 1. The good news is that it sums up "almost" to 1

provided δ is sufficiently small. This is shown below. Let $\epsilon := 1 - \sum_{y\in\mathcal{Y}} \pi(y)$. We now have

$$\begin{aligned}
\epsilon &= \overbrace{\sum_{y\in\mathcal{Y}} \mathbb{P}\left(Noise(X') = y\right)}^{=1} - \sum_{y\in\mathcal{Y}} \pi(y) \\
&= \sum_{y\in\mathcal{Y}} \mathbb{P}\left(Noise(X') = y\right) - \min_{x\in\mathcal{X}}(\mathbb{P}\left(Noise(x) = y\right)) \\
&= \sum_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}}(\mathbb{P}\left(Noise(X') = y\right) - \mathbb{P}\left(Noise(x) = y\right)) \\
&\leq \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} \max(0, \mathbb{P}\left(Noise(X') = y\right) - \mathbb{P}\left(Noise(x) = y\right))) \qquad (14) \\
&= \sum_{x\in\mathcal{X}} \Delta(Noise(x); Noise(X')) \\
&= |\mathcal{X}| \cdot \Delta((Noise(X); Noise(X')) \mid X) \\
&= \delta \cdot |\mathcal{X}|, \qquad (15)
\end{aligned}$$

where (14) comes from the fact that the maximum of positive values cannot be larger than their sum[8], and (15) follows from the assumption that the *Noise* function is δ-noisy. Now, let $Noise'(x)$ be a distribution defined as follows: for every $y \in \mathcal{Y}$ and every $x \neq \perp$ let:

$$\mathbb{P}\left(Noise'(x) = y\right) = (\mathbb{P}\left(Noise(x) = y\right) - \pi(y))/\epsilon, \qquad (16)$$

and otherwise:

$$\mathbb{P}\left(Noise'(\perp) = y\right) = \pi(y)/(1-\epsilon). \qquad (17)$$

We will later show how to sample $Noise'$ efficiently. Obviously this will automatically imply that (16) and (17) define probability distributions over $\mathcal{Y}$ (which may not be obvious at the first sight). First, however, let us show (5). To this end take any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and observe that

$$\begin{aligned}
&\mathbb{P}\left(Noise'(\varphi(x)) = y\right) \\
&= \mathbb{P}\left(\varphi(x) = x\right) \cdot \mathbb{P}\left(Noise'(x) = y\right) + \mathbb{P}\left(\varphi(x) = \perp\right) \cdot \mathbb{P}\left(Noise'(\perp) = y\right) \\
&= \epsilon \cdot (\mathbb{P}\left(Noise(x) = y\right) - \pi(y))/\epsilon + (1-\epsilon) \cdot \pi(y)/(1-\epsilon) \\
&= \mathbb{P}\left(Noise(x) = y\right) - \pi(y) + \pi(y) \\
&= \mathbb{P}\left(Noise(x) = y\right).
\end{aligned}$$

Which implies (5). What remains is to show how to sample $Noise'$ efficiently. Let us first show an efficient algorithm $\mathsf{Alg}_1(x)$ for computing $Noise'(x)$ for $x \neq \perp$:

$\underline{\mathsf{Alg}_1(x):}$

[8] More precisely, for every $\{Z_x\}_{x\in\mathcal{X}}$ we have:

$$\begin{aligned}
\max_{x\in\mathcal{X}}(Z_x) &\leq \sum_{x: Z_x \geq 0} Z_x \\
&= \sum_x \max(0, Z_x),
\end{aligned}$$

where in our case $Z_x := \mathbb{P}\left(Noise(x) = y\right)$.

1. Sample y from $Noise(x)$.
2. With probability $\pi(y)/\mathbb{P}\left(Noise(x)=y\right)$ resample y, i.e.: go back to Step 1.
3. Output y.

We now argue that $\mathsf{Alg_1}(x)$ indeed computes $Noise'(x)$ efficiently. Let $R_1 \in \{1,2,\ldots\}$ be a random variable denoting the number of times the algorithm $\mathsf{Alg_1}(x)$ performed Step 1. First observe that the probability of jumping back to Step 1 in Step 2 is equal to

$$\sum_y \mathbb{P}\left(Noise(x)=y\right)\cdot\pi(y)/\mathbb{P}\left(Noise(x)=y\right) = \sum_y \pi(y) \tag{18}$$

$$= 1-\epsilon \tag{19}$$

Therefore the probability of *not* jumping back to Step 1 in Step 2 is ϵ, and hence the expected number $\mathbb{E}\left(R_1\right)$ of the executions of Step 1 in $\mathsf{Alg_1}(x)$ is equal to $\sum_{i=1}^{n} i\cdot(1-\epsilon)^{i-1}\cdot\epsilon = 1/\epsilon$. Moreover for every $i=0,1,\ldots$ we have:

$$\begin{aligned}
&\mathbb{P}\left(\mathsf{Alg_1}(x)=y \wedge R_1 = i \mid R_1 \geq i\right)\\
&= \mathbb{P}\left(Noise(x)=y\right)\cdot\left(1-\left(\pi(y)/\mathbb{P}\left(Noise(x)=y\right)\right)\right)\\
&= \mathbb{P}\left(Noise(x)=y\right)-\pi(y)
\end{aligned}$$

Hence

$$\begin{aligned}
&\mathbb{P}\left(\mathsf{Alg_1}(x)=y\right)\\
&= \sum_{i=0}^{\infty}\mathbb{P}\left(\mathsf{Alg_1}(x)=y\wedge R_1=i\right)\\
&= \sum_{i=0}^{\infty}\mathbb{P}\left(\mathsf{Alg_1}(x)=y\wedge R_1=i \mid R_1\geq i\right)\cdot\mathbb{P}\left(R_1\geq i\right)\\
&= \left(\mathbb{P}\left(Noise(x)=y\right)-\pi(y)\right)\cdot\sum_{i=1}^{\infty}\mathbb{P}\left(R_1\geq i\right)\\
&= \left(\mathbb{P}\left(Noise(x)=y\right)-\pi(y)\right)\cdot\mathbb{E}\left(R_1\right)\\
&= \left(\mathbb{P}\left(Noise(x)=y\right)-\pi(y)\right)/\epsilon,
\end{aligned}$$

as required in (16). We now present an efficient algorithm $\mathsf{Alg_2}$ for computing $Noise'(\perp)$. Fix an arbitrary element $x_0 \in \mathcal{X}$, and execute the following.

$\mathsf{Alg_2}$:

1. Sample y from $Noise(x_0)$.
2. With probability $1-\left(\pi(y)/\mathbb{P}\left(Noise(x_0)=y\right)\right)$ resample y, i.e.: go back to Step 1.
3. Output y.

By a similar argument as in the case of $\mathsf{Alg_1}$ we obtain that the expected number R_2 of times the algorithm $\mathsf{Alg_2}$ performs Step 1 is equal to $\mathbb{E}\left(R_2\right)=1/(1-\epsilon)$. Moreover for every $i=1,2,\ldots$ we have:

$$\mathbb{P}\left(\mathsf{Alg_2}=b\wedge R_2=i \mid R_2\geq i\right)=\pi(y),$$

which, in turn, implies that $\mathbb{P}(\mathsf{Alg}_2(x) = y) = \pi(y)/(1-\epsilon)$, and hence the output of Alg_2 satisfies (17). Clearly, the expected running time of both algorithms is polynomial in $|\mathcal{X}|$ and $\mathbb{E}(R)$, where R is the number of execution of Step 1 in Alg_1 or Alg_2. We obviously have

$$\begin{aligned}\mathbb{E}(R) &= \mathbb{E}(R_1|\varphi(x) \neq \perp) \cdot \mathbb{P}(\varphi(x) \neq \perp) + \mathbb{E}(R_2|\varphi(x) = \perp) \cdot \mathbb{P}(\varphi(x) = \perp) \\ &= (1/\epsilon) \cdot \epsilon + (1/(1-\epsilon)) \cdot (1-\epsilon) \\ &= 2.\end{aligned}$$

Hence, the expected running time of $Noise'(\varphi(x))$ is polynomial in $|X|$. □

D Proof of Lemma 3

Without loss of generality assume that $\mathcal{A}$ simply outputs all the information that he gets. Thus (6) can be rewritten as:

$$(Noise_1(x_1), \ldots, Noise_\ell(x_\ell)) \stackrel{d}{=} out_{\mathcal{S}}(x_1, \ldots, x_\ell), \tag{20}$$

where $Noise_i$'s are the δ_i-noisy functions chosen by $\mathcal{A}$. By Lemma 2 for each i there exists $\epsilon_i \leq \delta_i \cdot |\mathcal{X}| \leq \delta \cdot |\mathcal{X}|$ and a randomized function $Noise'_i : \mathcal{X} \cup \{\perp\} \to \mathcal{X}$, such that for every $x \in \mathcal{X}$ we have

$$Noise_i(x) \stackrel{d}{=} Noise'_i(\varphi_i(x)), \tag{21}$$

where $\varphi_i : \mathcal{X}_i \to \mathcal{X}_i \cup \{\perp\}$ is the ϵ_i-identity function and $Noise'_i(\varphi_i(x))$ is computable in time polynomial in $|\mathcal{X}|$. We now describe the actions of $\mathcal{S}$. The sequence that he specifies is $(\epsilon_1, \ldots, \epsilon_\ell)$. After receiving $(y_1, \ldots, y_\ell)$ (equal to $(\varphi_1(x_1), \ldots, \varphi_\ell(x_\ell))$) he outputs

$$out(x_1, \ldots, x_\ell) := (Noise'_1(y_1), \ldots, Noise'_\ell(y_\ell))$$

(this clearly takes time that is expected polynomial in $\ell \cdot |\mathcal{X}|$). We now have

$$\begin{aligned}&(Noise'_1(y_1), \ldots, Noise'_\ell(y_\ell)) \\ \stackrel{d}{=}\ &(Noise'_1(\varphi_1(x_1)), \ldots, Noise'_\ell(\varphi_\ell(x_1))) \\ \stackrel{d}{=}\ &(Noise_1(x_1), \ldots, Noise_\ell(x_1))\end{aligned} \tag{22}$$

where (22) comes from (21). This implies (20) and hence it finishes the proof. □

E Proof of Lemma 4

As in the proof of Lemma 3 we assume that the simulated adversary $\mathcal{A}$ outputs all the information that he received. Moreover, since for $\epsilon' \leq \epsilon$ every ϵ'-identity function φ' can be simulated by the ϵ-identity function φ,[9] hence we can assume that each ϵ_i specified by $\mathcal{A}$ is equal to ϵ. Thus, we need to show a $2\epsilon\ell$-thereshold-probing simulator $\mathcal{S}$ such that for every $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$ we have

$$\Delta(\varphi_1(x_1), \ldots, \varphi_\ell(x_\ell)\ ;\ out_{\mathcal{S}}(x_1, \ldots, x_\ell) \mid out_{\mathcal{S}}(x_1, \ldots, x_\ell) \neq \perp) = 0, \tag{23}$$

[9] Just set $\varphi'(x) := \varphi(x)$ with probability ϵ'/ϵ, and $\varphi'(x) = \perp$ otherwise. Then clearly $\mathbb{P}(\varphi'(x) = x) = \epsilon \cdot \epsilon'/\epsilon = \epsilon'$.

(where each φ_i is the ϵ-identity function) and (8) holds. The simulator $\mathcal{S}$ proceeds as follows. First he chooses a sequence $(Z_1, \ldots, Z_\ell)$ of independent random variables in the by setting, for each i:

$$Z_i := \begin{cases} 1 \text{ with probability } \epsilon_i \\ 0 \text{ otherwise.} \end{cases}$$

Let Z denote the number of Z_i's equal to 1, i.e, $Z := \sum_{i=1}^{\ell} Z_i$. If $Z \geq 2\ell\epsilon$ then $\mathcal{S}$ outputs $\perp$. Otherwise he specifies the set $\mathcal{I}$ as $\mathcal{I} := \{i : Z_i = 1\}$. He receives $(x_{i_1}, \ldots, x_{i_{|\mathcal{I}|}})$. For all the remaining i's (i.e. those not in the set $\mathcal{I}$) the simulator sets $x_i := \perp$. He outputs $(x_1, \ldots, x_\ell)$. It is straightforward to see that $\mathcal{S}$ is $(2\epsilon\ell - 1)$-threshold-probing and that (23) holds. What remains is to show (8). Since $\mathbb{E}(Z) = \epsilon\ell$,

$$\begin{aligned} \mathbb{P}(Z \geq 2\ell\epsilon) &= \mathbb{P}(Z \geq 2\mathbb{E}(Z)) \\ &\leq \exp\left(-\frac{\epsilon\ell}{3}\right), \end{aligned} \tag{24}$$

where (24) comes from the Chernoff bound with $\xi = 1$ (cf. Lemma 7, Appendix A). This finishes the proof. □

F Proof of Corollary 1

By Lemma 3 there exists a $d/(4\ell)$-random-probing adversary $\mathcal{A}'$ whose output is distributed identically to the output of $\mathcal{A}$. In turn, by Lemma 4 for $t = 2 \cdot (d/(4\ell)) \cdot \ell = d/2$ there exists a $(t-1)$-threshold-probing adversary $\mathcal{S}$ whose output, conditioned on not being equal to $\perp$, is distributed identically to the output of $\mathcal{A}'$, and such that $\mathbb{P}(out_{\mathcal{S}}(x_1, \ldots, x_\ell) = \perp) \leq \exp(-d/12)$.

If $\mathcal{A}$ is poly-time noisy then clearly the expected working time of $\mathcal{A}'$ is polynomial in $\ell \cdot |\mathcal{X}|$. Since the working time of $\mathcal{S}$ is linear in the working time of $\mathcal{A}$ hence this finishes the proof. □

G Proof of Lemma 1

Let $\mathcal{A}$ be a δ-noisy circuit adversary attacking Γ'. We construct an efficient black-box simulator $\mathcal{S}$ such that for every k it holds that

$$\Delta\left(out\left(\mathcal{S} \overset{bb}{\leftrightarrows} \Gamma(k)\right) \;;\; out\left(\mathcal{A} \overset{noisy}{\leftrightarrows} \Gamma'(Enc(k))\right)\right) \leq |\Gamma| \cdot \exp(-d/12). \tag{25}$$

Observe that in our construction every gates gets transformed into a gadget of at most $3.5 \cdot d^2 + 2 \cdot d$ gates. Since each gate can have at most 2 inputs hence the total number of wires in a gadget is $\ell := 7 \cdot d^2 + 4 \cdot d$. Let $\gamma^1, \ldots, \gamma^{|\mathcal{G}|}$ be the gates of Γ. For each $i = 1, \ldots, \ell$ let the wires in the gadget in Γ' that corresponds to γ^i be denoted $(x_1^i, \ldots, x_\ell^i)$. Since $\delta = d/(4\ell|\mathbb{F}|)$, we can use Corollary 1 and simulate the noise from each $(x_1^i, \ldots, x_\ell^i)$ by a $(d/2-1)$-threshold-probing adversary $\mathcal{S}^i$ working in time polynomial in $\ell \cdot |\mathcal{X}|$. The simulation is perfect, unless $\mathcal{S}^i$ outputs $\perp$, which, by Corollary 1 happens with probability at most $\exp(-d/12)$. Hence, by the union-bound the probability that *some* $\mathcal{S}^i$ outputs $\perp$ is at most $|\Gamma| \cdot \exp(-d/12)$. Denote this event $\mathcal{E}$.

From Lemma 5 we know that every probing adversary that attacks Γ' by probing at most $\lfloor (d-1)/2 \rfloor \geq d/2 - 1$ wires from each gadget can be perfectly simulated in polynomial time by an adversary $\mathcal{S}$ with a black-box access to Γ . Hence, $\mathcal{A}$ can also be simulated perfectly by a black-box access to Γ conditioned on the fact that $\mathcal{E}$ did not occur. Hence we get

$$\Delta\left(out\left(\mathcal{S}^{\overset{bb}{\leftrightarrows}}\Gamma(k)\right)|\neg\mathcal{E}\ ;\ out\left(\mathcal{A}^{\overset{noisy}{\leftrightarrows}}\Gamma'(Enc(k))\right)\right)=0.$$

This, by Lemma 6 (Appendix A), implies (25). Obviously $\mathcal{S}$ works in time polynomial in $|\Gamma| \cdot d^2 \cdot |\mathbb{F}|$, which is polynomial in $\Gamma' \cdot |\mathbb{F}|$. This finishes the proof. □

H Proof of Theorem 2

The proof is similar to the one of Lemma 1 so we only describe the key differences. Let $\mathcal{A}$ be a δ-noisy adversary. The number ℓ corresponds now to the number of gates in each gadget, and hence it is equal to $3.5 \cdot d^2 + 2 \cdot d$. It is therefore straightforward to calculate that δ defined in (12) is equal to $(d/2)/(4\ell \cdot |\mathbb{F}|^2)$. Since the *Noise* function has domain of size $|\mathbb{F}|^2$ we can use Corollary 1 obtaining that $\mathcal{A}$ can be simulated by an adversary $\mathcal{S}$ that probes each gadget in less that $d/2$ positions. Since now each "position" corresponds to a gate in the circuit, hence the adversary needs to probe up to two wires to determine its value. Therefore $\mathcal{S}$ probes less than d wires in each gadget. Since d is now $1/2$ of what it was in the proof of Corollary 1, hence the error probability becomes $\exp(-d/12) = \exp(-d/24)$. □

I Proof of Corollary 2

From (4) with $\mathcal{X} = \mathbb{F} \times \mathbb{F}$ it follows that if *Noise* is δ'-noisy with respect to the β distance, then it is $(|\mathbb{F}| \cdot \delta'/2)$-noisy in the standard sense. Since this last value is equal to δ defined in (12), hence we can use Theorem 2 obtaining that Γ' is a $(\delta', |\Gamma| \cdot \exp(-d/24))$-noise-resilient implementation of Γ when the noise is defined using the β distance. □