From Random Probing to Noisy Leakages Without Field-Size Dependence

Gianluca Brian[1] * Stefan Dziembowski[2] Sebastian Faust[3]

[1] ETH Zurich, Zurich, Switzerland
[2] University of Warsaw and IDEAS NCBR, Warsaw, Poland
[3] TU Darmstadt, Darmstadt, Germany

March 4, 2024

Abstract

Side channel attacks are devastating attacks targeting cryptographic implementations. To protect against these attacks, various countermeasures have been proposed – in particular, the so-called masking scheme. Masking schemes work by hiding sensitive information via secret sharing all intermediate values that occur during the evaluation of a cryptographic implementation. Over the last decade, there has been broad interest in designing and formally analyzing such schemes. The random probing model considers leakage where the value on each wire leaks with some probability ϵ. This model is important as it implies security in the noisy leakage model via a reduction by Duc et al. (Eurocrypt 2014). Noisy leakages are considered the "gold-standard" for analyzing masking schemes as they accurately model many real-world physical leakages. Unfortunately, the reduction of Duc et al. is non-tight, and in particular requires that the amount of noise increases by a factor of $|\mathbb{F}|$ for circuits that operate over $\mathbb{F}$ (where $\mathbb{F}$ is a finite field). In this work, we give a generic transformation from ε-random probing to δ-average probing, with $\delta \approx \varepsilon^2$, which avoids this loss of $|\mathbb{F}|$. Since the average probing is identical to the noisy leakage model (Eurocrypt 2014), this yields for the first time a security analysis of masked circuits where the noise parameter in the noisy leakage model is independent of $|\mathbb{F}|$. The latter is particularly important for cryptographic schemes operating over large fields, e.g., the AES or the recently standardized post-quantum schemes.

1 Introduction

Side-channel attacks target cryptographic implementations by exploiting physical phenomena such as the power consumption or running time of a device [Koc96, KJJ99]. They extract sensitive information about the internals of the computation, often leading to devastating attacks against cryptographic implementations. One of the most prominent countermeasures to defeat side-channel attacks is the *masking scheme* [CJRR99, ISW03]. The basic idea of masking is to hide sensitive information by randomizing all internal values that occur during the computation. This typically works by encoding all internal values $v \in \mathbb{F}$ via a randomized encoding scheme $(v_1, \ldots, v_n) \leftarrow \mathsf{Enc}(v)$ and designing algorithms that securely compute on these encodings. Since designing secure masking schemes is challenging, most state-of-the-art masking schemes are proven secure within a *security model*. The security model captures the power of the adversary, which in particular requires us to accurately model the side-channel *leakage* emitting from

*Work partially done while at Sapienza University of Rome and University of Warsaw.

a device. Over the last decade, there has been lots of interest in designing and analyzing masking schemes (see, e.g., [ISW03, PR13, DDF14, BRTV21, AIS18, CRZ18, CS19, DFS19, CGZ20] and many more), where one of the most fundamental challenges is to come up with an accurate model for real-world side-channel attacks. Below we briefly recap the history of leakage models that have been considered for masking schemes.

The t-probing model. The t-probing model was introduced in the seminal work of Ishai, Sahai, and Wagner [ISW03]. The authors model (cryptographic) computation as a Boolean circuit $\mathcal{C}$, where the wires of the circuit carry the sensitive values. In the t-probing model, the adversary obtains the values of up to t wires in $\mathcal{C}$, which should not reveal more information about the cryptographic computation than what can be learned by just black-box access to the device. As this is a highly relevant attack in practice, security analysis in the t-probing model is the de-facto standard when designing and analyzing masking schemes.

Noisy leakages. While the t-probing model is an excellent first step for verifying the security of a masking scheme, it has two shortcomings for modeling real-world leakages accurately. On the one hand, it is too restrictive as it requires that the leakage is quantitatively bounded. This is in contrast to real-world leakages, which rarely can be described by a small number of bits; e.g., a physical power measurement typically results in megabytes of data that the adversary has to store. On the other hand, the t-probing leakage model is too generous. It allows the adversary to learn the *exact* value on some of the wires of the computation. Non-invasive real-world leakages, however, are typically rather noisy, and that noise is precisely what makes a side-channel attack difficult to carry out in practice. To address this shortcoming, Prouff and Rivain introduce the important model of *noisy leakages* [PR13]. Here, the adversary obtains a noisy version of each value carried on a wire during the computation of the circuit (e.g., the value perturbed by a Gaussian distribution).

More formally, consider a uniform random variable $\mathbf{X}$ over some finite field $\mathbb{F}$. A leakage function $\boldsymbol{\nu}$ is said to be δ noisy if the statistical distance between the uniform distribution $\mathbf{X}$ and the conditional distribution $\mathbf{X}|\boldsymbol{\nu}(\mathbf{X})$ is upper bounded by δ. Hence, by choosing δ appropriately, we can cover a broad range of different noise distributions. Moreover, the noisy leakage model eliminates the quantitative bound on the amount of leakage that an adversary obtains, thus incorporating many realistic leakages such as the horizontal side-channel attacks [BCPZ16]. Hence, it is considered the practically most relevant leakage model for a security evaluation of masked circuits. Unfortunately, however, the noisy leakage model has a significant drawback. An analysis in this model is highly complex, which was one of the reasons why in their original work, Prouff and Rivain were only able to give a security proof assuming that some parts of the masked computation is leak free.

Random probing model. In a follow-up work, Duc, Dziembowski, and Faust [DDF14] observed that, somewhat surprisingly, the noisy leakage model is equivalent to the random probing model [ISW03]. In the random probing model, we consider a particular noise distribution, where each wire leaks with some probability ε. More concretely, for a ε-random probing leakage function $\boldsymbol{\rho}$, we have for all $x \in \mathbb{F}$ that $\Pr[\boldsymbol{\rho}(x) \neq \perp] = \epsilon$. The main result of Duc et al. is then to show that any δ-noisy leakage function can be simulated via some ε random probing function (for some ε related to δ). As analyzing security in the random probing model is much simpler than a security proof in the noisy leakage model, Duc et al. can show that the original ISW construction is δ noisy leakage resilient for some parameter δ without requiring any leak-free computation.

Because of the connection between noisy leakages and the random probing model, there has been significant interest in proposing new constructions and analyzing their security in the random probing model [ADF16, Ajt11, AIS18, BRT21, BRTV21, BMRT22, CFOS21]. Important goals are finding new constructions that achieve security for a (nearly) optimal noise parameter δ, optimizing overheads of the masked algorithms, and presenting new composition results and automated tools for easing the analysis of masking schemes in the random probing model.

Shortcoming of an analysis in the random probing model. While δ-noisy leakage and ε-random probing leakage functions are related via the reduction of Duc et al., there is a significant loss between these two models. Concretely, for some $\delta > 0$ to simulate the output of the δ-noisy leakage function $\boldsymbol{\nu}(\cdot)$ via an ε-random probing $\boldsymbol{\rho}(\cdot)$, we need that $\varepsilon = \delta \cdot |\mathbb{F}|$. Put differently: suppose that for some $\varepsilon > 0$ we prove the security of the masking scheme in the ε-random probing model. When we transfer this result via the reduction of Duc et al. [DDF14] to the more realistic δ-noisy leakage model, we lose a factor of $|\mathbb{F}|$. While such a loss is generally undesirable, it is particularly problematic when $|\mathbb{F}|$ is large. Examples of such cases include the masking of the AES, which works in $GF(2^8)$, or even worse when masking some of the recent post-quantum schemes that typically operate in much larger fields.

Average probing model. To address the loss in the reduction of [DDF14], Dziembowski, Faust and Skorski [DFS15] introduce the average probing model. The average probing model makes a subtle but important change to the random probing model. In particular, a function $\boldsymbol{\alpha}$ is said to be δ-average probing if for a uniformly random $\mathbf{X} \leftarrow \mathbb{F}$, we have that $\Pr[\boldsymbol{\alpha}(\mathbf{X}) \neq \perp] = \delta$. Here, the probability is taken over the randomness of $\boldsymbol{\alpha}$ and the choice of $\mathbf{X}$. As a main result, Dziembowski et al. present a tight reduction between average probing and noisy leakages. Concretely, for any field $\mathbb{F}$, any δ-noisy leakage function $\boldsymbol{\nu}$ can efficiently be simulated by some δ-average probing leakage function.

While at first sight average probing looks very similar to random probing leakage, it turns out that a security analysis in the average probing model is significantly more challenging. Concretely, in [DFS15] the authors present a masking compiler that while eliminating the loss of $|\mathbb{F}|$ again requires that certain parts of the computation are leak-free.

Related work. By changing the metric for the computation of the noisy leakage, Prest, Goudarzi, Martinelli and Passelègue [PGMP19] prove a tighter bound. Namely, the authors consider two *worst-case* metrics, Relative Error and Average Relative Error, to measure the noisy leakage. The advantage of using such metrics is that they allow for proofs that do not have a security loss in the size of the field; in particular, they are able to reduce ε-random probing to ε-ARE-noisy leakage. However, the ARE metric incurs in bigger overheads, compared to Statistical Distance, when measuring leakage of functions; as an example, they consider the hamming weight with gaussian noise, which has a overhead of $\sqrt{\log(|\mathbb{F}|)}$ in the ARE setting compared to the SD setting.

Other ways to get rid of the loss in the field size is to consider arithmetic *programs* [GJR18] instead of circuits or consider fields of characteristic 2 [BCG+22]. In the first case, in particular, even if arithmetic programs and circuits are equivalent, a program allows to split the computation in smaller logical instructions (i.e., the word size of the computer) and consider the noisy leakage from those instructions. This, in turns, allows for a security loss of just $\log(|\mathbb{F}|)$.

1.1 Our contribution

The main result of our work is to present a generic compiler that transforms any circuits $\mathcal{C}$ with security against random probing into a circuit $\hat{\mathcal{C}}$ with security in the average probing model. Our transformation does not require leak free computation. Thus, using the tight relation between average probing and noisy leakages, we can show that any circuit $\mathcal{C}$ working over an arbitrary field $\mathbb{F}$ which is ε-random probing secure, can be transformed into a circuit $\hat{\mathcal{C}}$ that is δ-noisy leakage resilient, where (a) $\hat{\mathcal{C}}$ does not require leak-free computation, and (b) δ is independent of $|\mathbb{F}|$. In particular, if ε is a constant, then $\delta \approx \varepsilon^2$ is a constant (independent of $|\mathbb{F}|$ and the security parameter λ).

High-level idea. The main idea of our compiler is to ensure that any value $x \in \mathbb{F}$ occurring in the original circuit $\mathcal{C}$ is "encoded" into a sharing $(x_1, x_2) \in \mathbb{F}^2$ where individually each share x_i is (almost) uniformly distributed over $\mathbb{F}$. Let us briefly discuss the main idea by taking a look at the most simple case of a circuit that consists only of a single wire x. In our transformed circuit $\hat{\mathcal{C}}$ the value x is represented by a secret sharing (x_1, x_2), where x_1, x_2 are uniformly random in $\mathbb{F}$ subject to the constrained that $x_1 + x_2 = x$. Our approach now works by using random probing leakage from x to simulate the average probing leakage from (x_1, x_2). More concretely, we show that for any δ-average probing leakage function $\boldsymbol{\alpha}$ the leakage $(\boldsymbol{\alpha}(x_1), \boldsymbol{\alpha}(x_2))$ can be simulated from $\varepsilon = 2\delta$-random probing leakage of x.

The basic idea to show the above statement is as follows. Since (x_1, x_2) is a secret sharing of x, each one of x_1, x_2 is uniformly random when considered independently, and therefore the marginal probability that any of them leaks is exactly δ. Then, by the union bound, the probability that any of x_1, x_2 leaks is bounded by 2δ. The strategy is then to construct a simulator Sim that outputs $(\perp, \perp)$ (i.e., no leakage occurred) when the random probing leakage is $\perp$, and is able to fully simulate the values on the wires (and, therefore, the corresponding leakage) when the random probing leakage is $x \in \mathbb{F}$. Clearly, upon input $x \in \mathbb{F}$, the simulator $\mathsf{Sim}(x)$ cannot naively output the real distribution, since otherwise $(\perp, \perp)$ would be output too often and the simulated average probing leakage would not be identically distributed to the real one; instead, $\mathsf{Sim}(x)$ adjusts the probabilities so that $(\perp, \perp)$ is output less often, in order to match the real distribution. This can be done efficiently, e.g., by rejection sampling.

In the previous paragraph, we only described the most simple case where we simulate average probing leakage from a single encoding $(x_1, x_2) \leftarrow \mathsf{Enc}(x)$. Our analysis gets much more involved, when we move to the setting where the adversary obtains leakage from the entire computation. For a high-level overview how we can simulate average random probing leakage of an encoded complex circuit from random probing leakage of an (unencoded) circuit, we refer the reader to the technical overview in Section 1.2.

Impact. We believe that our work shows a missing piece for the analysis of masking schemes. As discussed above, currently there is lots of interest in designing new masking countermeasures in the random probing model. However, to transform these results to the noisy leakage model, all these works will require a noise parameter δ that decreases by $1/|\mathbb{F}|$.[1] Our result shows that any of the existing constructions for random probing leakage can be transformed into a construction that is secure against noisy leakage *without* suffering this loss. One direct application of our result is to show that for any field $\mathbb{F}$ the ISW construction is noisy leakage resilient for $\delta \approx 1/n^2$ (where n is the number of shares used in the ISW-masked circuit). Earlier results required either leak-free gates [DFS15] or required a noise parameter of $\delta \approx 1/(n|\mathbb{F}|)$ [DDF14]. We leave

[1] Notice that means that we need an amount of noise that is at least proportional to $|\mathbb{F}|$.

it as an important open question to apply our result to some of the existing constructions for random probing security and further optimize their parameters.

1.2 Technical overview

In the previous section, we showed how to simulate average probing leakage of a single encoding $(x_1, x_2) \leftarrow \mathsf{Enc}(x)$ given random probing leakage from x. To extend our analysis to complex circuits, we follow a standard gate-by-gate approach. More precisely, we show that for each gate g in $\mathcal{C}$ there is an efficient gadget $\hat{g}$ in $\hat{\mathcal{C}}$ (in fact, the overhead is a small constant), where average probing leakage from $\hat{g}$ can be simulated by random probing leakage from the inputs of g. The simulation and analysis of this transformation is significantly more involved, and in particular requires us to take care of internal wires that are not uniformly distributed.[2] For instance, this is the case for the multiplication gadget, where internally we compute values $x \cdot y$, where x and y are uniform over $\mathbb{F}$.

Then, we also need to show that the gadgets we construct are composable in a "safe" way, meaning that one cannot break the simulatability of a gadget by looking at other parts of the circuit. This turns out to be quite involved, as in particular the simulation must be careful with sampling the outputs of the gadgets that are consistent with whatever an adversary has observed in the past. To explain the main technical challenge, let us consider a concrete toy example that highlights the technical difficulty of our analysis. Suppose that we are working in the finite field $\mathbb{F} = \mathbb{F}_5$ with 5 elements and that the leakage function $\boldsymbol{\alpha}$ leaks 0 with probability 1 and everything else with probability 0. Clearly, $\boldsymbol{\alpha}$ is a $\frac{1}{5}$-average probing function. We show two different ways to construct a uniform encoding of $4 \in \mathbb{F}_5$ which lead to two different leakage distributions when we consider leakage from the the whole circuit.

First, suppose the simplest way to generate an encoding $(x_1, x_2) \xleftarrow{\$} \mathsf{Enc}(4)$ by sampling from the distribution $\mathsf{Enc}(4)$. Assuming that the leakage from the encoding is $(\perp, \perp)$, then the possible encodings given the leakage are $(1,3), (2,2), (3,1)$. Notice that they also all appear with the same probability of $\frac{1}{3}$.

A second circuit that produces an encoding of 4 is through a sum of encodings of 3 and 1. Suppose again that nothing in the circuit leaks, then the possible encodings for 3 are $(4,4), (2,1), (1,2)$ and the possible encodings for 1 are $(4,2), (3,3), (2,4)$. By writing a table with all the possible sums, we can eliminate some of the outcomes as they are impossible (since they contain 0, which leaks with probability 1, and we assumed that nothing leaks); the remaining outcomes are, again, $(1,3), (2,2), (3,1)$. However, this time $(2,2)$ appears slightly less often as shown in the table below.

$+$	$(4,4)$	$(2,1)$	$(1,2)$
$(4,2)$	$(3,1)$	$(1,3)$	$(0,4)$
$(3,3)$	$(2,2)$	$(0,4)$	$(4,0)$
$(2,4)$	$(1,3)$	$(4,0)$	$(3,1)$

This, in turn, means that the probability of the outcome being $(2,2)$ is $\frac{1}{5}$, compared to $(1,3)$ and $(3,1)$ which appear with probability $\frac{2}{5}$. As can be seen from the above example, the simulation of the leakage from a gadget depends on the leakage that occured in other parts of the circuit. Notably, this is even true, when nothing leaked, i.e., when the leakage was only $\perp$. This is unlike in the random probing model, where leaking $\perp$ gives freedom to the gadget simulator and thus allows for much simpler composition results.

[2] Recall that in the case of the encoding each of the x_i is individual uniform over $\mathbb{F}$.

Approximation. In the above example, a naive simulation that does not take into account leakage from other parts of the circuit fails because even if the output distribution is uniform, it is not uniform anymore if we condition on some event such as "something leaks from the circuit computation" or "nothing leaks from the computation". However, the above example also shows a way to deal with such dependencies. Indeed, we can see that, even if the output distribution of the gadget (i.e., the encoding of 4) is not uniform conditioned on the leakage, it is still somewhat close to the simplest case in which we only consider leakage from the encoding and nothing else. We exploit this observation to construct a simulator that is able to approximate the output distribution of gadgets even when the simulator knows nothing about the input or the rest of the circuit. Then, we can follow a similar strategy as the one for simulating average probing leakage from the simple encoding. Concretely, as previously, when the simulator receives some additional information that allows to simulate the exact distributions, we can compensate for the above approximation error such that the final distribution output by the simulator is identical to the real distribution, even conditioned on the exact input encoding and leakage from the gadget.

In order to apply the above strategy, we need two additional properties from our gadgets. The first one is that all the wires on the gadget need to be uniform when considered independently, so that we can apply the same idea as in the simple encoding $(x_1, x_2) \leftarrow \mathsf{Enc}(x)$. Unfortunately, this is not possible due to the presence of multiplication gates (recall that, when $\mathbf{x}, \mathbf{y}$ are uniform over $\mathbb{F}$, the product $\mathbf{xy}$ takes the value 0 slightly more often); however, we are able to show that the strategy still holds if we relax this requirement and only ask for *close-to-uniform* wires. The second property, which we call *output-independence*, states that the values on the wires of the gadget are (close-to-)uniform even when the output of the gadget is known in full, and, additionally, the output of the gadget is identically distributed to the one of a fresh encoding. Looking ahead, this is needed so that the simulator is always able to approximate the output distribution.

Fortunately, it turns out that the above two properties are not hard to achieve. Indeed, the internal wires are close-to-uniform when all the input encodings are re-randomized and, additionally, re-randomizing the output of the gadget allows to achieve the output-independence property. Furthermore, some of the gadgets (i.e., addition, subtraction) already satisfy the output-independence property due to the properties of Enc (i.e., $\mathsf{Enc}(x) + \mathsf{Enc}(y)$ is identically distributed to $\mathsf{Enc}(x + y)$).

Gadget simulators. Now that we established the main strategy to construct the gadgets and the basic idea to simulate their leakage, we will provide the high-level idea of the composition. We start by describing the security guarantee that the gadget simulator has to satisfy and which will suffice for composition. For simplicity assume that the gate g only has 1 input and 1 output. A first attempt to define the gadget simulator $\mathsf{Sim}_{\widehat{g}}$ is to require for any input encoding $\widehat{x}$ to the gadget $\widehat{g}$ that the following holds:

$$\mathbf{\Lambda} \stackrel{?}{\equiv} \mathsf{Sim}_{\widehat{g}}\left(\boldsymbol{\rho}(x)\right),$$

where $\mathbf{\Lambda}$ is the random variable of the average-probing leakage from the gadget $\widehat{g}$ on input $\widehat{x}$ and $x \in \mathbb{F}$ is the input value on the unmasked circuit. Unfortunately, as discussed above, this is not sufficient, because the simulator needs to receive some additional information from the environment to produce the output encoding of the gadget as well:[3]

$$\mathsf{Real}_{\widehat{g}}(\widehat{x}) \equiv \mathsf{Sim}_{\widehat{g}}\left(\boldsymbol{\rho}(x), \mathsf{info}\right),$$

[3]Looking ahead, this information is needed for composition in order to produce a consistent simulation.

where $\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\mathbf{\Lambda}, \widehat{\mathbf{y}})$ is the joint distribution of the real leakage and the real output upon input $\widehat{x}$. Moreover, info denotes some auxiliary information that we will explain in a moment.

Unfortunately, the input $\boldsymbol{\rho}(x)$ does not suffice for the gadget simulator to produce $\mathsf{Real}_{\widehat{g}}(\widehat{x})$ and hence we slightly strengthen the power of the simulator by giving it as input $\mathsf{Blind}_{\boldsymbol{\rho}(x)}(\widehat{x})$ which outputs $\widehat{x}$ if $\boldsymbol{\rho}(x) = x$ and outputs $\perp$ otherwise. This allows us to let the simulator play "safe" and make it output everywhere $\perp$ when $\boldsymbol{\rho}(x) = \perp$ and compensate for this "overestimation" when getting $\widehat{x}$ as input. Hence, we get:

$$\mathsf{Real}_{\widehat{g}}(\widehat{x}) \equiv \mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}_{\boldsymbol{\rho}(x)}(\widehat{x}), \mathsf{info}\right),$$

where we explain the meaning of info next. Observe that the simulator needs to correctly sample $\widehat{\mathbf{y}}$ as discussed in the previous paragraph. However, this cannot be done if $\boldsymbol{\rho}(x) = \perp$. Looking ahead, the final observation here is that $\widehat{\mathbf{y}}$ is only needed by the simulator of the next gadget if the random probing of the output value $y = g(x)$ reveals y. When this is the case, we can give y to the simulator, which can then *approximate* the correct distribution $\widehat{\mathbf{y}}$ as we described above; the simulator will then compensate for this approximation error in the case in which it receives $\widehat{x}$ in full. Hence, the final notion of a composable gadget simulator is given by:

$$(\mathbf{\Lambda}, \mathsf{Blind}_{y_?}(\widehat{\mathbf{y}})) \equiv \mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}_{\boldsymbol{\rho}(x)}(\widehat{x}), y_?\right),$$

where $(\mathbf{\Lambda}, \widehat{\mathbf{y}}) = \mathsf{Real}_{\widehat{g}}(\widehat{x})$, $y_? = \boldsymbol{\rho}(y)$ denotes the outcome of random probing $y = g(x)$, and the above holds for all $\widehat{x}$ and all $\widehat{y} = \widehat{g}(\widehat{x})$. This notion states that the above simulator outputs a leakage distribution $\mathbf{\Lambda}$ and a "blinded" output distribution $\widehat{\mathbf{y}}$ that is identical to the real distribution even when considered jointly.

Sadly, this approach only works when all the inputs of the gate are given to the simulator, which, for gates with fan-in 2, causes the leakage parameter to be squared, i.e., $\delta \approx \varepsilon^2$ when starting from ϵ-random probing. We leave the problem of filling this gap open for future work.

Composition of gadget simulators. Now that we established the correct notion for the simulatable gadgets, we want to apply it to prove simulatability of the whole circuit. We plan to do so by hybrid argument. Namely, we define as many hybrid experiments as there are gates in the circuit, and we replace the real gadgets with the simulated gadgets one by one; then, we show that two consecutive hybrids are identically distributed, by a reduction to the simulatability of the gadgets. More in detail, the reduction would first run the real circuit until the challenge gadget, then receives some either real or simulated leakage/output pair and finally continues with the simulated gadgets. This is enabled by the above notion of simulatability, which states that the distribution of the output does not change when moving from the real to the simulated gadget. In turns, this means that we can completely replace the real gadget with the corresponding simulator and use the output of the simulator to feed the subsequent simulators in the circuit.

Structure of the paper. We state formally all the necessary notions for the gadgets in Section 3, in which we also show how to achieve such notions. Then, in Section 4 we state the simulatability notions for circuits, and we show how to use the simulatable gadgets to achieve the simulatable circuit. Finally, we conclude in Section 5 and pose some open problems for future research.

2 Preliminaries

Notation. For a number $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote sets by uppercase calligraphic letters $\mathcal{A}, \mathcal{B}, \mathcal{X}, \mathcal{Y}, \ldots$ and random variables by bold letters $\mathbf{A}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \ldots$;

similarly, we use bold letters for randomized functions, like $\mathbf{f}, \mathbf{g}, \boldsymbol{\alpha}, \boldsymbol{\rho}$. For a set $\mathcal{X}$, we denote by $x \xleftarrow{\$} \mathcal{X}$ the fact that x is uniformly sampled from $\mathcal{X}$. For two random variables $\mathbf{X}, \mathbf{Y}$ over the same set $\mathcal{X}$, the *statistical distance between* $\mathbf{X}$ *and* $\mathbf{Y}$ is denoted as $\Delta(\mathbf{X}, \mathbf{Y})$ and defined as

$$\Delta(\mathbf{X}, \mathbf{Y}) \coloneqq \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \Pr[\mathbf{X} = x] - \Pr[\mathbf{Y} = x] \right|.$$

Whenever $\Delta(\mathbf{X}, \mathbf{Y}) = 0$, we say that $\mathbf{X}$ and $\mathbf{Y}$ are *identically distributed*, and we denote this fact by writing $\mathbf{X} \equiv \mathbf{Y}$. We denote by $d(\mathbf{X}) \coloneqq \Delta(\mathbf{X}, \mathbf{U})$ the distance between $\mathbf{X}$ and the uniform random variable $\mathbf{U}$ over $\mathcal{X}$. In general, we will refer to $\mathbf{X}$ as *uniform* if $d(\mathbf{X}) = 0$, and, for $\gamma \in [0, 1]$, as *γ-close-to-uniform* if $d(\mathbf{X}) \leq \gamma$.

For a function $f : \mathcal{X} \to \mathcal{Y}$ and a vector $x = (x_1, \ldots, x_n) \in \mathcal{X}^n$, we overload the notation and write $f(x)$ for the function $(x_1, \ldots, x_n) \mapsto (f(x_1), \ldots, f(x_n))$; this applies to randomized functions $\mathbf{f}$ as well. Furthermore, for $x \in (\mathcal{X} \cup \{\perp\})^n$, we overload the notation of $\perp$ as well and write $x = \perp$ if $x = (\perp, \ldots, \perp)$.

Circuits. We model computation as an arithmetic circuit $\mathcal{C}$ carrying values from an (arbitrary) finite field $\mathbb{F}$ on their wires and using the following gates to carry out computation in $\mathbb{F}$:

- $\mathsf{ADD}, \mathsf{SUB}, \mathsf{MUL}$, which compute, respectively, the sum, the difference and the product in $\mathbb{F}$ of their inputs,
- IN, which has no input and models either some constant or the external input to the circuit,
- OUT, which has one input and no output, and models the output produced by the circuit,
- RND, which has no input and produces a uniformly random and independent element of $\mathbb{F}$,
- and CPY, which takes as input a single value and outputs two copies.

We say that the gates $\mathsf{ADD}, \mathsf{SUB}, \mathsf{MUL}, \mathsf{CPY}$ are *functional gadgets*, in that they compute a function (respectively, $(x, y) \mapsto x + y, (x, y) \mapsto x - y, (x, y) \mapsto xy, x \mapsto (x, x)$). Furthermore, we view the circuit as a directed acyclic graph $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ in which

- $\mathcal{G}$ is the set of the gates of the circuit, seen as a set of nodes of the graph,
- $\mathcal{W}$ is the set of the wires of the circuit, seen as a set of edges of the graph.

We also assume that $\mathcal{C}$ is connected (otherwise, it suffices to look at each connected component separately, since they act independently) and that $\mathcal{G}$ is topologically sorted. Namely, for every two gates $g_i, g_j \in \mathcal{G}$ with $i < j$, then g_i comes *before* g_j. Intuitively, this means that g_j is not needed to compute g_i.

A circuit $\mathcal{C}$ models computation on some (possibly randomized) input $x \in \mathbb{F}^{m^*}$ and produces some output $y \in \mathbb{F}^{n^*}$; we denote this by writing $y \leftarrow \mathcal{C}(x)$. The numbers m^* and n^* are respectively the *fan-in* and the *fan-out* of the circuit, and they correspond respectively to the number of input gates and the number of output gates of $\mathcal{C}$. Sometimes, we need more detail about the computation of the circuit. In this case, we denote by $\mathbf{W}(x)$ the list of all the values that the wires of the circuit take upon input x.

Circuit compiler. A circuit compiler Γ takes as input the original circuit $\mathcal{C}$ and produces a new circuit $\widehat{\mathcal{C}} = \Gamma(\mathcal{C})$. Unless stated otherwise, we denote by regular letters everything that belongs to the original circuit, like $\mathcal{C}, x, g \in \mathcal{G}, w \in \mathcal{W}$, and by letters with hats everything that belongs to the transformed circuit, like $\widehat{\mathcal{C}}, \widehat{x}, \widehat{g} \in \widehat{\mathcal{G}}, \widehat{w} \in \widehat{\mathcal{W}}$.

The compiler Γ is associated with an encoding scheme $\mathsf{Enc} : \mathbb{F} \to \mathbb{F}^\ell$, $\mathsf{Dec} : \mathbb{F}^\ell \to \mathbb{F}$ such that, for all $x \in \mathbb{F}$, $\mathsf{Dec}(\mathsf{Enc}(x)) = x$ (or, more formally, this happens with probability 1 over the randomness of the encoding). Then, the wires of the original circuit $\mathcal{C}$ are represented in the transformed circuit $\widehat{\mathcal{C}}$ as *wire bundles* that carry the value of the wire in encoded form. The input $x \in \mathbb{F}^{m^*}$ to $\mathcal{C}$ is then transformed into the encoded input $\mathsf{Enc}(x)$ to $\widehat{\mathcal{C}}$. The main challenge to compile $\mathcal{C}$ to $\widehat{\mathcal{C}}$ is to describe how to transform the gates. For each gate $g \in \mathcal{G}$, the compiler constructs a sub-circuit $\widehat{g}$, the so-called *gadget*, that represents the computation of $\widehat{g}$ in $\widehat{\mathcal{C}}$ and carries out the output of g in encoded form. We emphasize that the computation in the gadgets use the standard gates defined in the previous section. Notice also that, for simplicity, we focus in this work on *stateless* circuits, i.e., the circuits do not have memory gates. Hence, we require that compiled circuits $\widehat{\mathcal{C}}$ receive their inputs in encoded form.

In what follows, we partition the wires of every gadget $\widehat{g}$ into three disjoint subsets:

- the *input* wires are the wires that carry the input encoding $\widehat{x}$;
- the *output* wires are the wires that carry the output encoding $\widehat{y} = \widehat{g}(\widehat{x})$;
- all the other wires belong to the set of the *internal* wires, which carry the computation inside $\widehat{g}$.

Notice that, whenever a gadget $\widehat{g_1}$ is connected to $\widehat{g_2}$, the wires from $\widehat{g_1}$ to $\widehat{g_2}$ are both input wires for $\widehat{g_2}$ and output wires for $\widehat{g_1}$. Since every value that is output by a gadget is then input into another gadget, we can ignore the input wires and consider every gadget to be just its internal and output wires.

Finally, we now establish some notation that is useful when reasoning about gadgets. For a gadget $\widehat{g}$, we denote by $\mathcal{I}\mathsf{n}_{\widehat{g}}$ the set of the possible inputs and by $\mathcal{O}\mathsf{ut}_{\widehat{g}}(\widehat{x})$ the set of the possible outputs upon input $\widehat{x}$.

Leakage. A *leakage* function is a (possibly randomized) function $\mathbf{f} : \mathbb{F} \to \Omega$ for some set Ω. As discussed in detail in the introduction, in this work we focus on *probing* functions, and we use two probing models. Let $\mathbb{F}$ be a finite field. A randomized function $\boldsymbol{\varphi} : \mathbb{F} \to \mathbb{F} \cup \{\perp\}$ is called a *(wire) probing function (over the field $\mathbb{F}$)* if for every $x \in \mathbb{F}$ we have that $\boldsymbol{\varphi}(x)$ is equal either to x or to $\perp$, where $\perp$ is a special symbol that denotes that the probing function failed to probe the wire. For $\varepsilon \in (0, 1)$, such a function is called:

- ε*-random* if for every $x \in \mathbb{F}$, $\Pr[\boldsymbol{\varphi}(x) = x] = \varepsilon$, and
- ε*-average* if for the uniform random variable $\mathbf{x}$ over $\mathbb{F}$, $\Pr[\boldsymbol{\varphi}(\mathbf{x}) = \mathbf{x}] = \varepsilon$.

In what follows, we use the letter $\boldsymbol{\rho}$ to denote random probing and the letter $\boldsymbol{\alpha}$ to denote average probing.

Sometimes we need to keep or discard values depending on the outcome of a probing function. Towards this, for any $x \in \mathbb{F} \cup \{\perp\}$ and any $\widehat{x} \in \mathbb{F}^\ell$, we define the function

$$\mathsf{Blind}_x(\widehat{x}) \coloneqq \begin{cases} \widehat{x} & \text{if } x \in \mathbb{F}, \\ \perp & \text{if } x = \perp. \end{cases}$$

We extend Blind_x to a function $\mathbb{F}^n \to \mathbb{F}^{\ell n}$ in the usual way, i.e., by applying it to every component of $x = (x_1, \ldots, x_n)$ and $\widehat{x} = (\widehat{x_1}, \ldots, \widehat{x_n})$:

$$\mathsf{Blind}_x(\widehat{x}) \coloneqq \left(\mathsf{Blind}_{x_1}\left(\widehat{x_1}\right), \ldots, \mathsf{Blind}_{x_n}\left(\widehat{x_n}\right)\right).$$

Furthermore, we consider an *all-or-nothing* function Blind^*_x defined for every $x \in (\mathbb{F} \cup \{\perp\})^n$ such that

$$\mathsf{Blind}^*_x(\widehat{x}) \coloneqq \begin{cases} \widehat{x} & \text{if } x \in \mathbb{F}^n, \\ \perp & \text{otherwise.} \end{cases}$$

2.1 Simple facts

In this section we list some results that we are going to use later in this work.

First of all, we state two general fact about products of uniform distributions.

Lemma 2.1. *Let $\mathbb{F}$ be a field and let $\mathbf{x}, \mathbf{y}$ be two independent and uniform random variables over $\mathbb{F}$. Then, the product $\mathbf{xy}$ is $\frac{1}{|\mathbb{F}|}$-close to uniform.*

The proof of this fact can be found in Appendix A.1.

Lemma 2.2 ([MPR07])**.** *Let $\mathbb{G}$ be a group $\mathbf{x}, \mathbf{y}$ be two independent random variables over $\mathbb{G}$. Then,*

$$d(\mathbf{x} + \mathbf{y}) \leq 2d(\mathbf{x})d(\mathbf{y}).$$

A consequence of the above is the following.

Lemma 2.3. *Let $\gamma \in \left[0, \frac{1}{2}\right]$ be a parameter, let $\mathbb{F}$ be a field and let $\mathbf{x}, \mathbf{y}$ be two independent and γ-close-to-uniform distributions over $\mathbb{F}$. Then, $\mathbf{x} + \mathbf{y}$ is γ-close-to-uniform.*

The following lemma states that, informally, if a distribution over a set of values is "uniform enough", then it is always possible that nothing leaks even in the stronger model of average probing.

Lemma 2.4. *Let $\gamma \in \left[0, \frac{1}{2}\right], \delta \in [0, 1]$ be parameters. Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ be a distribution over $\mathbb{F}^k$ and assume that there exist k' values $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i'_k}$ that are γ-close to uniform and that all the other $k - k'$ values $\mathbf{x}_i$ are uniform. Finally, let $\boldsymbol{\alpha}$ be any δ-average-probing function. Then,*

$$\Pr\left[\boldsymbol{\alpha}(\mathbf{x}) = \perp\right] \geq 1 - \left(k + k'\gamma|\mathbb{F}|\right)\delta$$

The proof of this fact can be found in Appendix A.2.

3 Composable gadgets against average probing

In this section, we design the gadgets that we are using in our circuit compiler. In Section 4 we will define a circuit simulator to prove that, intuitively, our compiler transforms random-probing-resilient circuits into average-probing-resilient circuits. To do so, the simulator only receives as input the random probing leakage $\boldsymbol{\rho}(\mathbf{W}(x))$ from the original circuit $\mathcal{C}$. Our strategy is to construct gadgets in a composable way so that we can reduce the simulatability of the circuit to the simulatability of every gadget. In particular, the circuit simulator will forward the random probing leakage (or part of it) to the gadget simulator, which will then produce leakage from the wires of the gadgets. The main difficulty here is that, for every gadget, the distribution of the values on the wires depends on the distribution of the input values, and such distribution is not always the same if we condition on all the leakage that happened before the

gadget. However, if we allow the gadget simulator to also receive, when available, the encoding of the input, then the simulation can be accurate. Intuitively, the notion of simulatabilty should look like

$$\forall \widehat{x} \in \mathbb{F}^{\ell m} : \mathbf{\Lambda} \equiv \mathsf{Sim}_{\widehat{g}} \left(\mathsf{Blind}^{*}_{\boldsymbol{\rho}(x)} \left(\widehat{x} \right) \right),$$

where $\mathbf{\Lambda}$ is the leakage from the wires of the real gadget and $x = \mathsf{Dec}(\widehat{x})$. Looking forward, when simulating two consecutive gadgets, the circuit simulator has no way to generate the input to the second gadget simulator, since it does not come from the real circuit. This means that every simulator is required to simulate the output of the gadget as well, unless it is not needed (i.e., when the random probing $\boldsymbol{\rho}(y)$ of the output value y results in $\perp$). The following definition captures exactly this property.

Definition 3.1 (Gadget simulatability)**.** *Let $\varepsilon, \delta \in [0,1], \ell \in \mathbb{N}$ be parameters. Let g be a functional gate with fan-in m and fan-out n. Let $\widehat{g}$ be the corresponding gadget for an encoding of size ℓ. We say that $\widehat{g}$ is ε-random to δ-average* leakage-simulatable *if for every δ-average probing function $\boldsymbol{\alpha}$ there exists a simulator $\mathsf{Sim}_{\widehat{g}}$ such that*

$$\forall \widehat{x} \in \mathbb{F}^{\ell m} : \mathsf{Real}_{\widehat{g}} \left(\widehat{x} \right) \equiv \mathsf{Sim}_{\widehat{g}} \left(\mathsf{Blind}^{*}_{\boldsymbol{\rho}(x)} \left(\widehat{x} \right), y \right)$$

and

$$\forall \widehat{x} \in \mathbb{F}^{\ell m} : (\mathbf{\Lambda}, \perp) \equiv \mathsf{Sim}_{\widehat{g}} \left(\mathsf{Blind}^{*}_{\boldsymbol{\rho}(x)} \left(\widehat{x} \right), \perp \right).$$

In the above, $x = \mathsf{Dec}(\widehat{x}), y = g(x)$, $\boldsymbol{\rho}$ is the ε-random probing function and $(\mathbf{\Lambda}, \widehat{\mathbf{y}}) = \mathsf{Real}_{\widehat{g}}$ is a sample from the real experiment, which computes $\widehat{\mathbf{y}} \equiv \widehat{g}(\widehat{x})$, obtains the average-probing leakage $\mathbf{\Lambda}$ from the wires and then outputs $(\mathbf{\Lambda}, \widehat{\mathbf{y}})$.

Additional properties. Now that we established the notion, we want to show that we are actually able to achieve it. Intuitively, we should construct every gadget and then prove that they meet the above definitions; however, the proofs are quite long and very similar. Instead, we proceed the other way around: first, we show a general technique to construct a simulator, and then we show that such general technique can be applied to simulate all the gadgets that we construct. In this way, we only have a general proof for the simulator and then one corollary for every gadget. However, to proceed in this way, we need two additional properties from the gadgets.

The first property states, informally, that every wire on the gadget carries a close-to-uniform value when considered independently of the rest of the gadget.

Definition 3.2 (Close-to-uniform gadget)**.** *Let g be any gate and let $\widehat{g}$ be the corresponding gadget. Let k be the number of wires in $\widehat{g}$ and let $\gamma \in \left[0, \frac{1}{2}\right], k' \in \mathbb{N}$ be parameters such that $k' \leq k$. Furthermore, let $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ be the random variable of the values on the wires of $\widehat{g}$. We say that $\widehat{g}$ is a (k, k', γ)-*close-to-uniform gadget *if $\mathbf{w}_1, \ldots, \mathbf{w}_k$ are all γ-close-to-uniform and, additionally, there are $k - k'$ indices $i \in [k]$ such that $\mathbf{w}_i$ is uniform. If $k' = 0$ (i.e., the gadget does not contain non-uniform wires), we simply say that the gadget is k-*uniform*.*

In the following corollary, which is a direct consequence of Lemma 2.4, we show that, intuitively, close-to-uniform gadgets have nice "hiding" properties.

Corollary 3.3. *Let $\delta \in (0,1), \gamma \in \left[0, \frac{1}{2}\right], k, k', \ell, m \in \mathbb{N}$ be parameters such that $k' \leq k$ and ℓ is the size of the encoding. Let g be a functional gate with fan-in m, and let $\widehat{g}$ be the corresponding gadget for an encoding of size ℓ. Assume that $\widehat{g}$ is (k, k', γ)-close-to-uniform. Then, for all $\widehat{x} \in \mathbb{F}^{\ell m}$ and all δ-average probing functions $\boldsymbol{\alpha}$,*

$$\Pr\left[\mathbf{\Lambda} = \perp\right] \geq 1 - k\delta - k'\gamma\delta|\mathbb{F}|,$$

where $\mathbf{\Lambda} \coloneqq \mathsf{Leak}_{\widehat{g}}(\boldsymbol{\alpha}, \widehat{x})$ *is the distribution of the leakage from the wires of* $\widehat{g}$ *upon input* $\widehat{x}$*.*

The second property states, informally, that every *internal* wire of the gadget carries a value that, when considered independently of the other internal wires, is also independent of the output of the gadget.

Definition 3.4 (Output independence)**.** *Let* g *be any gate with fan-out* n *and let* $\widehat{g}$ *be the corresponding gadget for an encoding of size* $\ell \in \mathbb{N}$*. Let* $k \in \mathbb{N}$ *be the number of wires in* $\widehat{g}$ *and let* $n \in \mathbb{N}$ *be the fan-out of* g*. Furthermore, let* $(\mathbf{w}_1, \ldots, \mathbf{w}_{k-\ell n})$ *be the random variable of the values on the* internal *wires of* $\widehat{g}$ *and let* $\widehat{\mathbf{y}}$ *be the random variable of the values on the* output *wires. We say that* $\widehat{g}$ *is* output-independent *if*

$$\forall i \in [k - \ell n], \forall w \in \mathbb{F} : \Pr\left[\mathbf{w}_i = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] = \Pr\left[\mathbf{w}_i = w\right]$$

and, additionally,

$$\widehat{\mathbf{y}} \equiv \mathsf{Enc}(\mathsf{Dec}(\widehat{\mathbf{y}})).$$

The following lemma shows a useful lower bound on the distribution of any non-leaking output of close-to-uniform gadgets with output-independence. Roughly speaking, this means that, when the gadget does not leak anything (i.e., $\mathbf{\Lambda} = \bot$), it is still possible to approximate from below the output distribution of the gadget with a distribution that does not depend on the input value $\widehat{x}$, namely, the distribution $\mathsf{Enc}(y)|\boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot$.

Lemma 3.5. *Let* $\gamma \in \left[0, \frac{1}{2}\right], \delta \in (0,1), k, k', \ell, m, n \in \mathbb{N}$ *be parameters such that* $k' \leq k$ *and* $(k + k'\gamma|\mathbb{F}|)\delta < 1$*. Let* g *be any gate with fan-in* m *and fan-out* n *and let* $\widehat{g}$ *be the corresponding gadget for an encoding of size* ℓ*. Assume that* $\widehat{g}$ *is* (k, k', γ)*-close-to-uniform and output-independent. Then, for all* $\widehat{x} \in \mathbb{F}^{\ell m}$ *and all* $\widehat{y} \in \mathcal{O}\mathsf{ut}_{\widehat{g}}(\widehat{x})$*,*

$$\Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot\right] \leq \frac{\Pr\left[\mathsf{Real}_g(\widehat{x}) = (\bot, \widehat{y})\right]}{1 - (k + k'\gamma|\mathbb{F}|)\delta}, \tag{1}$$

where $y = \mathsf{Dec}(\widehat{y})$*.*

The proof of this fact can be found in Appendix A.3.

The gadget simulator. Now we are ready to define the actual gadget simulator. Recall that the simulator receives two inputs which are possibly "blinded", namely the input of the gadget $\widehat{x}$ and the output of the original gate y. In what follows, we assume that, whenever $y \neq \bot$, $y \in \mathbb{F}^n$ or, in other words, y is given in full to the simulator;[4] it is easy to extend the simulator and the analysis to the general case.

For a gate g with fan-in m and fan-out n and its respective gadget $\widehat{g}$ for an encoding of size ℓ, we consider the following simulator $\mathsf{Sim}_{\widehat{g}}$.

- Upon input $(\bot, \bot)$, simply output $(\bot, \bot)$.
- Upon input $(\bot, y)$, sample $\widehat{y}$ with probability

  $$\Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot\right]$$

  and output $(\bot, \widehat{y})$.

[4] Notice that the only gate with $n > 1$ is the copy gate, for which all the components of y are the same.

- Upon input $(\widehat{x}, \perp)$, sample $\Lambda \neq \perp$ with probability

$$\frac{1}{\varepsilon^m} \Pr\left[\mathbf{\Lambda} = \Lambda\right],$$

where $\mathbf{\Lambda}$ is the leakage performed by the real experiment $\mathsf{Real}_{\widehat{g}}$, or set $\Lambda = \perp$ with probability

$$\frac{1}{\varepsilon^m} \Pr\left[\mathbf{\Lambda} = \perp\right] - \frac{1-\varepsilon^m}{\varepsilon^m}.$$

Then, output $(\Lambda, \perp)$.

- Upon input $(\widehat{x}, y)$, sample $(\Lambda, \widehat{y})$ for $\Lambda \neq \perp$ with probability

$$\frac{1}{\varepsilon^m} \Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\Lambda, \widehat{y})\right]$$

and sample $(\perp, \widehat{y})$ with probability

$$\frac{1}{\varepsilon^m} \Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\perp, \widehat{y})\right] - \frac{1-\varepsilon^m}{\varepsilon^m} \Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right].$$

Finally, output the sampled values.

The remainder of the section is dedicated to the proof of the following theorem stating that, informally, the output of the simulator is identically distributed to the output of the real experiment.

Theorem 3.6. *Let $\gamma \in \left[0, \frac{1}{2}\right], \varepsilon, \delta \in (0,1), k, k', \ell, m, n \in \mathbb{N}$ be parameters such that $k' \leq k$ and*

$$(k + k'\gamma|\mathbb{F}|)\delta < \varepsilon^m.$$

Let g be a functional gate with fan-in m and fan-out n and let $\widehat{g}$ be the corresponding gadget for an encoding of size ℓ. Assume that $\widehat{g}$ is (k, k', γ)-close-to-uniform and output-independent. Then, the above simulator is such that

$$\forall \widehat{x} \in \mathbb{F}^{\ell m} : \mathsf{Real}_{\widehat{g}}\left(\widehat{x}\right) \equiv \mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}^*_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right), y\right)$$

and

$$\forall \widehat{x} \in \mathbb{F}^{\ell m} : (\mathbf{\Lambda}, \perp) \equiv \mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}^*_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right), \perp\right).$$

In the above, $x = \mathsf{Dec}(\widehat{x}), y = g(x)$, $\boldsymbol{\rho}$ is the ε-random probing function and $(\mathbf{\Lambda}, \widehat{\mathbf{y}}) = \mathsf{Real}_{\widehat{g}}$ is a sample from the real experiment, which computes $\widehat{\mathbf{y}} \equiv \widehat{g}(\widehat{x})$, obtains the average-probing leakage $\mathbf{\Lambda}$ from the wires and then outputs $(\mathbf{\Lambda}, \widehat{\mathbf{y}})$.

Proof. First of all, we need to show that the simulator is well-defined. Namely, we need to show that the output of the simulator is actually a distribution, meaning that all the probabilities are non-negative and sum up to 1. This is trivial when the simulator receives $(\perp, \perp)$ or $(\perp, y)$ as input, therefore, in what follows, we focus on the case in which the simulator actually receives $\widehat{x}$.

- When the simulator receives $(\widehat{x}, \perp)$ as input, it samples Λ and outputs $(\Lambda, \perp)$. All the probabilities of the simulator outputting $\Lambda \neq \perp$ are trivially non-negative. When summing all of them, we get that the probability of outputting any $\Lambda \neq \perp$ is

$$\frac{1}{\varepsilon^m} \Pr\left[\mathbf{\Lambda} \neq \perp\right].$$

Since $\widehat{g}$ is close-to-uniform, we can apply Corollary 3.3 to obtain that

$$\frac{1}{\varepsilon^m}\Pr\left[\boldsymbol{\Lambda} \neq \perp\right] \leq \frac{1}{\varepsilon^m}(k + k'\gamma|\mathbb{F}|)\delta < 1, \tag{2}$$

where the last inequality holds by the hypothesis on the parameters. On the other side, the simulator outputs $\Lambda = \perp$ with probability

$$\frac{1}{\varepsilon^m}\Pr\left[\boldsymbol{\Lambda} = \perp\right] - \frac{1-\varepsilon^m}{\varepsilon^m} = 1 - \frac{1}{\varepsilon^m}\Pr\left[\boldsymbol{\Lambda} \neq \perp\right] > 0,$$

where the last inequality follows from Eq. (2). It follows that the probabilities are non-negative and that they sum up to 1.

- When the simulator receives $(\widehat{x}, y)$ as input, the proof is very similar to the above. Indeed, the simulator samples $(\Lambda, \widehat{y})$ for $\Lambda \neq \perp$ always with non-negative probability; on the other side, the probability of sampling $(\perp, \widehat{y})$ is non-negative if and only if

$$\Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\perp, \widehat{y})\right] - (1-\varepsilon^m)\Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right] \geq 0$$

or, by rearranging the terms, if and only if

$$\Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right] \leq \frac{\Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\perp, \widehat{y})\right]}{1-\varepsilon^m}. \tag{3}$$

However, since $\widehat{g}$ is close-to-uniform and output-independent by hypothesis, we can apply Lemma 3.5, which gives

$$\Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right] \leq \frac{\Pr\left[\mathsf{Real}_{g}(\widehat{x}) = (\perp, \widehat{y})\right]}{1-(k + k'\gamma|\mathbb{F}|)\delta}.$$

Then, Eq. (3) follows from the hypothesis $(k + k'\gamma|\mathbb{F}|)\delta < \varepsilon^m$. Finally, it is easy to see that all the terms sum up to 1.

Now it remains to show that the simulator perfectly simulates the real distribution. We start the analysis of the simulator from the simple case of $\Lambda \neq \perp$. Notice that $\mathsf{Sim}_{\widehat{g}}$ only outputs $\Lambda \neq \perp$ if it receives $\widehat{x}$ as input (otherwise, the simulator plays safe and outputs $\perp$). Therefore,

$$\begin{aligned}
&\Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}^*_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right), y\right) = (\Lambda, \widehat{y})\right] \\
&= \Pr\left[\mathsf{Blind}^*_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right) = \widehat{x}\right]\Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\widehat{x}, y\right) = (\Lambda, \widehat{y})\right] && (4)\\
&= \Pr\left[\boldsymbol{\rho}(x) = x\right]\Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\widehat{x}, y\right) = (\Lambda, \widehat{y})\right] && (5)\\
&= \varepsilon^m \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\widehat{x}, y\right) = (\Lambda, \widehat{y})\right] && (6)\\
&= \Pr\left[\mathsf{Real}_{\widehat{g}}\left(\widehat{x}\right) = (\Lambda, \widehat{y})\right], && (7)
\end{aligned}$$

where $\boldsymbol{\Lambda}$ is the leakage produced by the real experiment $\mathsf{Real}_{\widehat{g}}$. In the above derivation,

- Eq. (4) follows by definition of conditional probability;
- Eq. (5) holds because $\mathsf{Blind}^*_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right) = \widehat{x}$ if and only if $\boldsymbol{\rho}(x) = x$;
- Eq. (6) follows by definition of random probing and from the fact that $x \in \mathbb{F}^m$;
- and finally, Eq. (7) follows by how we defined the simulator to behave upon input $(\widehat{x}, y)$.

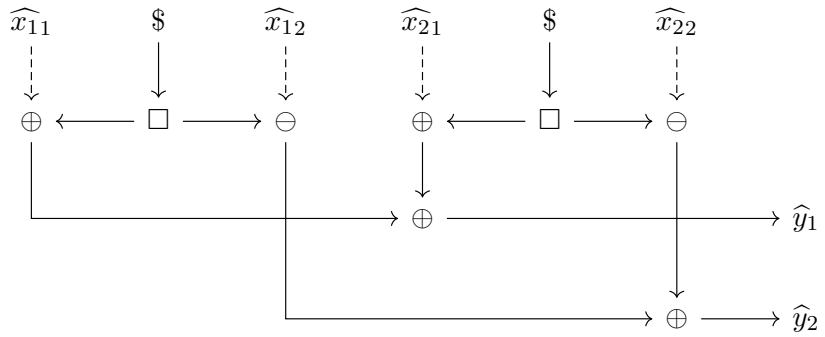

Figure 1: Scheme for the $\widehat{\mathsf{ADD}}$ gadget.

The proof for the case in which the simulator does not get y is analogous, therefore we now only focus on the case $\Lambda = \perp$. This case is a bit more involved, because $\mathsf{Sim}_{\widehat{g}}$ may output $\Lambda = \perp$ both when the input $\widehat{x}$ is given and when the simulator receives $\perp$. Namely,

$$
\begin{aligned}
&\Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}^{*}_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right), y\right) = (\Lambda, \widehat{y})\right] \\
&= \Pr\left[\mathsf{Blind}^{*}_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right) = \widehat{x}\right] \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\widehat{x}, y\right) = (\perp, \widehat{y})\right] && (8) \\
&\quad + \Pr\left[\mathsf{Blind}^{*}_{\boldsymbol{\rho}(x)}\left(\widehat{x}\right) = \perp\right] \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\perp, y\right) = (\perp, \widehat{y})\right] \\
&= \Pr\left[\boldsymbol{\rho}(x) = x\right] \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\widehat{x}, y\right) = (\perp, \widehat{y})\right] && (9) \\
&\quad + \Pr\left[\boldsymbol{\rho}(x) = \perp\right] \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\perp, y\right) = (\perp, \widehat{y})\right] \\
&= \varepsilon^{m} \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\widehat{x}, y\right) = (\perp, \widehat{y})\right] && (10) \\
&\quad + (1 - \varepsilon^{m}) \Pr\left[\mathsf{Sim}_{\widehat{g}}\left(\perp, y\right) = (\perp, \widehat{y})\right] \\
&= \Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\perp, \widehat{y})\right] - (1 - \varepsilon^{m}) \Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right] && (11) \\
&\quad + (1 - \varepsilon^{m}) \Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right] \\
&= \Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\perp, \widehat{y})\right]. && (12)
\end{aligned}
$$

In the above derivation,

- Eqs. (8) to (10) follow for the same reasoning as in Eqs. (4) to (6);
- Eq. (11) follows by how we defined the simulator to behave upon input $(\widehat{x}, y)$;
- and finally, Eq. (12) follows by simplifying the sum and subtraction of the same term.

This concludes the proof.

Now that we proved the main result of this section, we define the gadgets that we are going to use in the final construction.

3.1 Basic arithmetic gadgets

We start by constructing the $\widehat{\mathsf{ADD}}$ gadget, depicted in Fig. 1 and consisting of 2 RND gates, 2 CPY gates, 4 ADD gates and 2 SUB gates, for a total of 10 gates. Furthermore, the $\widehat{\mathsf{ADD}}$ gadget consists of 12 wires. The random gates output uniform random values $\mathbf{r}_1$ and $\mathbf{r}_2$, which are then copied and used to refresh the input values. Then, the refreshed values are summed

component-wise in order to obtain $\widehat{y}_1, \widehat{y}_2$. Overall, without taking into account wires carrying the same value, the wires in the gadget carry the following values:

$\mathbf{r}_1,\ \mathbf{r}_2,$	(From the random gates)
$\widehat{x_{11}} + \mathbf{r}_1,\ \widehat{x_{12}} - \mathbf{r}_1,$	(Refreshing of the first input)
$\widehat{x_{21}} + \mathbf{r}_2,\ \widehat{x_{22}} - \mathbf{r}_2,$	(Refreshing of the second input)
$\widehat{x_{11}} + \mathbf{r}_1 + \widehat{x_{21}} + \mathbf{r}_2,\ \widehat{x_{12}} - \mathbf{r}_1 + \widehat{x_{22}} - \mathbf{r}_2$	(Output encoding)

It is easy to see that every one of the above values is uniformly distributed when taken independently. Finally, ADD is a deterministic gate with fan-in 2 and fan-out 1, therefore it is a functional gate.

From the above, $\widehat{\mathsf{ADD}}$ is a 12-uniform gadget for a deterministic functional gate. We now show that the gadget is output-independent. Indeed, we have that

$$\widehat{\mathbf{y}}_1 = \widehat{x_{11}} + \mathbf{r}_1 + \widehat{x_{21}} + \mathbf{r}_2 \quad \text{and} \quad \widehat{\mathbf{y}}_2 = \widehat{x_{12}} - \mathbf{r}_1 + \widehat{x_{22}} - \mathbf{r}_2,$$

which means that, by uniformity of $\mathbf{r}_1$ and $\mathbf{r}_2$, the output is identically distributed to a fresh encoding. Furthermore, for the internal wires we have that, for every $r \in \mathbb{F}$, every $\widehat{x} \in \mathbb{F}^4$, and every $\widehat{y} \in \mathcal{O}\mathsf{ut}_{\widehat{\mathsf{ADD}}}(\widehat{x})$,

$$\begin{aligned}
\Pr\left[\mathbf{r}_1 = r \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] &= \Pr\left[\mathbf{r}_1 = r \,\middle|\, \begin{matrix} \widehat{x_{11}} + \mathbf{r}_1 + \widehat{x_{21}} + \mathbf{r}_2 = \widehat{y}_1 \\ \widehat{x_{12}} - \mathbf{r}_1 + \widehat{x_{22}} - \mathbf{r}_2 = \widehat{y}_2 \end{matrix}\right] \\
&= \Pr\left[\mathbf{r}_1 = r \,|\, \mathbf{r}_1 = c_1 - \mathbf{r}_2\right] \\
&= \Pr\left[\mathbf{r}_2 = c_1 - r\right] \\
&= \Pr\left[\mathbf{r}_1 = r\right],
\end{aligned}$$

where $c_1 = \widehat{y}_1 - \widehat{x_{11}} - \widehat{x_{21}}$, the first equality follows from making $\widehat{\mathbf{y}}$ explicit, the second equality follows by rearranging the terms and the fact that $\widehat{y} \in \mathcal{O}\mathsf{ut}_{\widehat{\mathsf{ADD}}}(\widehat{x})$, the third equality follows by replacing $\mathbf{r}_1$ with its value given by the condition, and the last equality follows because both $\mathbf{r}_1, \mathbf{r}_2$ are uniform samples from $\mathbb{F}$. A similar reasoning can be applied to the values carried by all the other internal wires, thus proving output-independence.

Now that we proved that the gadget is 12-uniform and output-independent, we can apply Theorem 3.6 to obtain the following.

Corollary 3.7. *Let $\varepsilon, \delta \in (0, 1)$ be parameters such that*

$$12\delta < \varepsilon^2.$$

Then, $\widehat{\mathsf{ADD}}$ is ε-random to δ-average simulatable.

By replacing the last two ADD gates with SUB gates in $\widehat{\mathsf{ADD}}$, we obtain the new gadget $\widehat{\mathsf{SUB}}$, depicted in Fig. 2, which computes SUB. The analysis is completely analogous to the one for $\widehat{\mathsf{SUB}}$, hence we have the following.

Corollary 3.8. *Let $\varepsilon, \delta \in (0, 1)$ be parameters such that*

$$12\delta < \varepsilon^2.$$

Then, $\widehat{\mathsf{SUB}}$ is ε-random to δ-average simulatable.

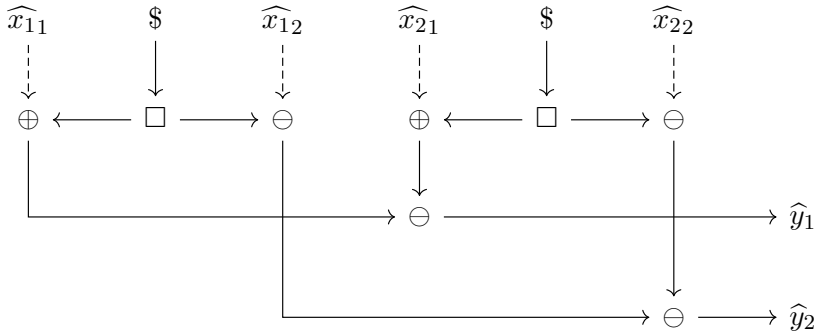

Figure 2: Scheme for the $\widehat{\mathsf{SUB}}$ gadget.

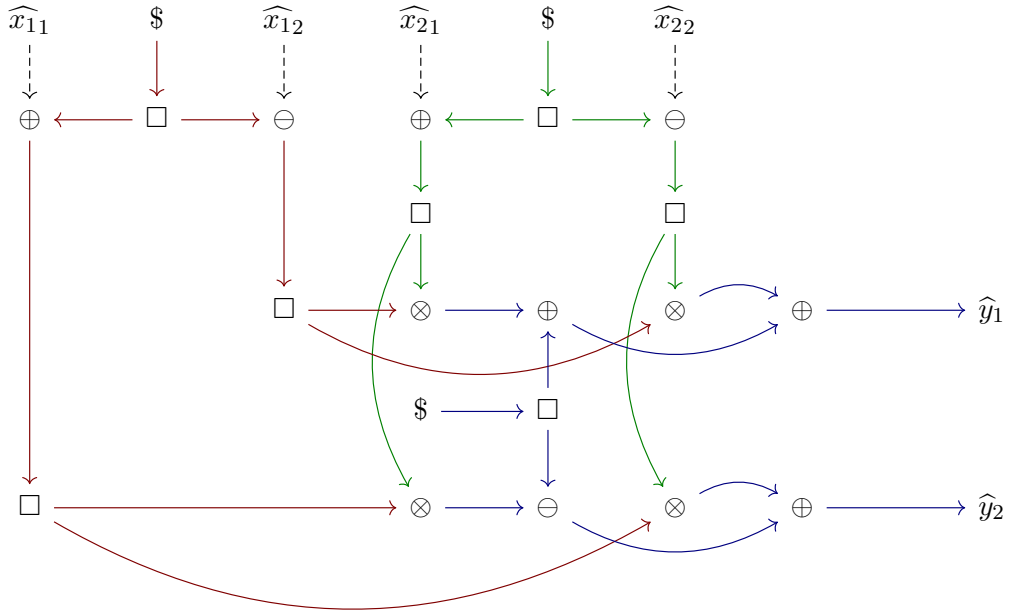

Figure 3: Scheme for the $\widehat{\mathsf{MUL}}$ gadget. To make the diagram easier to understand, the wires carrying an encoding of x_1 are marked in red, the wires carrying an encoding of x_2 are marked in green, and the wires after the multiplication are marked in blue.

3.2 Multiplication gadget

The $\widehat{\mathsf{MUL}}$ gadget, depicted in Fig. 3, is probably the most complex gadget due to the properties of multiplication. Indeed, it is the only non-uniform gadget, consisting of 3 RND gates, 7 CPY gates, 5 ADD gates, 3 SUB gates, and 4 MUL gates, for a total of 22 gates. Furthermore, the $\widehat{\mathsf{MUL}}$ gadget consists of 29 wires. The first two random gates output uniform random values $\mathbf{r}_1$ and $\mathbf{r}_2$, which are then copied and used to refresh the input values. Then, the refreshed values are (copied and) multiplied so to obtain the four terms of the product of two binomials:

$$(a_1 + a_2)(b_1 + b_2) = a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2.$$

Finally, the partial sums are computed, in order to have encodings of only two elements, and then a third random gate is used to refresh the output encoding. Overall, without taking into account wires carrying the same value, the wires in the gadget carry the following values:

$\mathbf{r}_1,\ \mathbf{r}_2,\ \mathbf{r}_3$	(From the random gates)
$\widehat{x_{11}} + \mathbf{r}_1,\ \widehat{x_{12}} - \mathbf{r}_1,$	(Refreshing of the first input)
$\widehat{x_{21}} + \mathbf{r}_2,\ \widehat{x_{22}} - \mathbf{r}_2,$	(Refreshing of the second input)
$(\widehat{x_{12}} - \mathbf{r}_1)(\widehat{x_{21}} + \mathbf{r}_2),$ $(\widehat{x_{11}} + \mathbf{r}_1)(\widehat{x_{21}} + \mathbf{r}_2)$	(First column of multiplications)
$(\widehat{x_{12}} - \mathbf{r}_1)(\widehat{x_{22}} - \mathbf{r}_2),$ $(\widehat{x_{11}} + \mathbf{r}_1)(\widehat{x_{22}} - \mathbf{r}_2)$	(Second column of multiplications)
$(\widehat{x_{12}} - \mathbf{r}_1)(\widehat{x_{21}} + \mathbf{r}_2) + \mathbf{r}_3,$ $(\widehat{x_{11}} + \mathbf{r}_1)(\widehat{x_{21}} + \mathbf{r}_2) - \mathbf{r}_3$	(First column rerandomized)
$(\widehat{x_{12}} - \mathbf{r}_1)(\widehat{x_{21}} + \mathbf{r}_2) + \mathbf{r}_3$ $+ (\widehat{x_{12}} - \mathbf{r}_1)(\widehat{x_{22}} - \mathbf{r}_2)$	(First output component)
$(\widehat{x_{11}} + \mathbf{r}_1)(\widehat{x_{21}} + \mathbf{r}_2) - \mathbf{r}_3$ $+ (\widehat{x_{11}} + \mathbf{r}_1)(\widehat{x_{22}} - \mathbf{r}_2).$	(Second output component)

It is easy to see that all the values on the red and green wires are uniform when considered independently, and so are the output wires and the wires carrying $\mathbf{r}_3$. The only wires that are not uniform are the outputs of the four MUL gates. However, by Lemma 2.1, the outputs of the MUL gates are $\frac{1}{|\mathbb{F}|}$-close-to-uniform. Finally, a reasoning similar to the one in Section 3.1 shows that multiplication gadgets are also output-independent. This suffices to apply Theorem 3.6 with parameters $k = 29$, $k' = 6$ and $\gamma = \frac{1}{|\mathbb{F}|}$, and obtain the following result.

Corollary 3.9. *Let $\varepsilon, \delta \in (0, 1)$ be parameters such that*

$$\left(29 + 6 \cdot \frac{1}{|\mathbb{F}|} \cdot |\mathbb{F}|\right)\delta = 35\delta < \varepsilon^2.$$

Then, $\widehat{\mathsf{MUL}}$ is ε-random to δ-average simulatable.

3.3 Copy gadget

The last functional gadget is the $\widehat{\mathsf{CPY}}$ gadget, depicted in Fig. 4, which is fairly simple. It consists of 3 RND gates, 5 CPY gates, 3 ADD gates and 3 SUB gates, for a total of 14 gates. Furthermore, the $\widehat{\mathsf{CPY}}$ gadget consists of 19 wires. There are no arithmetic operations except for the ones needed to refresh the encodings; on the other hand, the encodings are refreshed

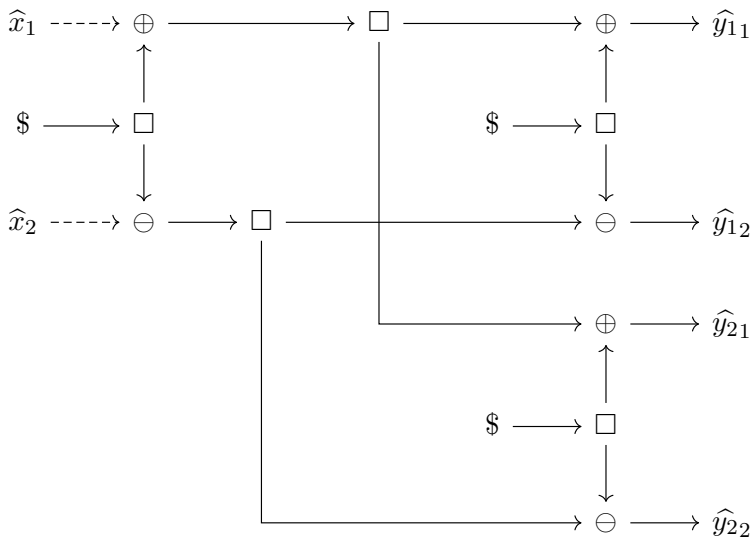

Figure 4: Scheme for the $\widehat{\mathsf{CPY}}$ gadget.

both on the input side, to ensure that the gadget is uniform, and on the output side, to ensure that the gadget is output-independent.

By applying Theorem 3.6, we obtain the following.

Corollary 3.10. *Let $\varepsilon, \delta \in (0,1)$ be parameters such that*

$$19\delta < \varepsilon^2.$$

Then, $\widehat{\mathsf{CPY}}$ is ε-random to δ-average simulatable.

3.4 Putting everything together

The only missing gadgets are the ones for $\mathsf{IN}, \mathsf{OUT}, \mathsf{RND}$. However, OUT only has input wires, therefore the corresponding gadget would be empty. On the other hand, IN and RND are very similar, therefore we focus on IN and then explain later the differences with RND.

Let g_0 be any input gate and let g_1 be the gate that receives the input from g_0; finally, let $w = (g_0, g_1)$ be the wire connecting the two gates. By assuming a leak-free encoder before the circuit, the encoding that we would place on the wire bundle $\widehat{w}$ is already a fresh encoding or, in other words, the value on every wire of the bundle is uniformly distributed when considered independently. This actually means two things:

- there is no need to re-randomize the values directly coming from the input in gadget $\widehat{g_1}$;
- if we consider $\widehat{g_0} \cup \widehat{w} \cup \widehat{g_1}$, we are only adding two uniform wires.

By applying the above, we can remove the re-randomization component (thus removing 3 wires) and consider the input wires as part of the gadget instead (thus adding 2 wires). As an example, Fig. 5 is what the $\widehat{\mathsf{ADD}}$ gadget looks like when we apply the above strategy.

Corollary 3.11. *Let $k, k' \in \mathbb{N}$, $\gamma \in \left[0, \frac{1}{2}\right]$, $\varepsilon, \delta \in (0,1)$ be parameters such that*

$$\left(k + k'\gamma|\mathbb{F}|\right)\delta \leq \varepsilon^2$$

and assume that $\widehat{g}$ is a (k, k', γ)-close-to-uniform and output-independent gadget. Then, the above operation of merging the input wires and removing the re-randomization component produces a new gadget $\widehat{g}'$ that is $(k-1, k', \gamma)$-close-to-uniform and output-independent.

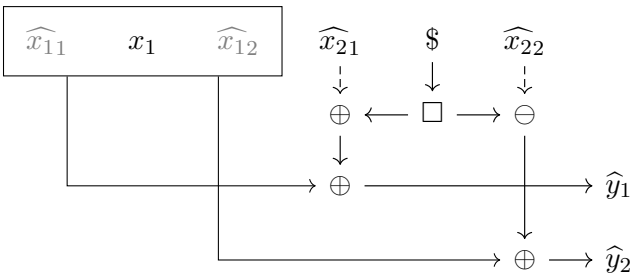

Figure 5: Scheme for the $\widehat{\mathsf{ADD}}$ gadget, in which we replaced the rerandomizing component with the input wires; the box represents the leak-free encoding circuit that takes as input x_1 and outputs the fresh encoding $(\widehat{x_{11}}, \widehat{x_{12}})$.

Proof. Notice that the above operation removes three uniform wires (namely, the ones from the random gate that are used to re-randomize the encoding) and adds two uniform wires (namely, the ones carrying the fresh encoding). Hence, this operation converts a (k, k', γ)-close-to-uniform gadget into a $(k-1, k', \gamma)$-close-to-uniform gadget. Finally, this operation does not modify any other wire, and it is easy to see that the output-independence property is preserved.

Since we addressed all gadgets, we can have the following general corollary that handles all of them.

Corollary 3.12. *Let $\varepsilon, \delta \in (0,1)$ be parameters such that*

$$35\delta \leq \varepsilon^2.$$

Then, all the gadgets defined so far are ε-random to δ-average simulatable.

4 The circuit compiler

The circuit compiler simply replaces every wire with the respective wire bundle and every gate with the respective gadget. Our goal is to achieve the following definition.

Definition 4.1 (Simulatability of circuits)**.** *Let $\mathcal{C}$ be a circuit with fan-in m^* and fan-out n^*. Let $\widehat{\mathcal{C}}$ be the corresponding transformed circuit. Let Φ, Ψ be two families of (possibly randomized) functions. Finally, let $\ell \in \mathbb{N}$ be the size of the encoding. We say that $\widehat{\mathcal{C}}$ is* Φ-to-Ψ-simulatable *(from $\mathcal{C}$) if there exists a simulator $\mathsf{Sim}_{\mathcal{C}}$ such that for every $x^* \in \mathbb{F}^{m^*}$ input to $\mathcal{C}$ and every function $\psi \in \Psi$ there exists a function $\varphi \in \Phi$:*

$$\mathsf{Leak}_{\mathcal{C}}\left(\mathsf{Enc}(x^*)\right) \equiv \mathsf{Sim}_{\mathcal{C}}\left(\varphi\left(\mathbf{W}(x^*)\right)\right). \tag{13}$$

In the above, $\mathbf{W}(x^*)$ is the distribution of the values on the wires of the original circuit $\mathcal{C}$ upon input x^*.

In what follows, we show the following.

Theorem 4.2. *Let $\varepsilon, \delta \in (0,1)$ be parameters such that $35\delta < \varepsilon^2$. Let $\mathcal{C}$ be a circuit and let $\widehat{\mathcal{C}}$ be the circuit transformed in the following way:*

- *for every functional gate $g \in \mathcal{C}$, we place the corresponding gadget, as described in Section 3, in $\widehat{\mathcal{C}}$;*
- *for every wire $w \in \mathcal{C}$, we place the corresponding wire bundle in $\widehat{\mathcal{C}}$, consisting of 2 wires;*

- *if $w = (g_1, g_2)$, the corresponding wire bundle $\widehat{w}$ will connect $\widehat{g_1}$ to $\widehat{g_2}$.*

Then, the transformed circuit $\widehat{\mathcal{C}}$ is ε-random to δ-average simulatable.

The proof proceeds by hybrid argument. Namely, let k^* be the number of functional gates in $\mathcal{C}$. Then, the real experiment $\mathsf{Leak}_{\mathcal{C}}(\mathsf{Enc}(x^*))$ can be rewritten as follows.

1. Sample $\widehat{x^*} = \mathsf{Enc}(x^*)$.
2. For every $j \in [k^*]$, compute $\Lambda_j, \widehat{y_j}$ as follows.
   (a) For every input bundle of gadget $\widehat{g_j}$ that comes from an input of the circuit, place the corresponding value taken from the vector $\widehat{x^*}$.
   (b) For every input bundle of gadget $\widehat{g_j}$ that comes from a random gate of the circuit, sample uniform $r \in \mathbb{F}$ and place $\mathsf{Enc}(r)$ on the wires.
   (c) Since $\mathcal{C}$ is topologically sorted, every other input bundle of gadget $\widehat{g_j}$ already has a value, which has been computed in a previous iteration as $\widehat{y_{j'}}$ for some $j' < j$.
   (d) Let $\widehat{W_j} \in \mathbb{F}^{k_j}$ be the list of values on all the k_j wires of $\widehat{g_j}$.
   (e) The leakage Λ_j is set to be $\Lambda_j \coloneqq \boldsymbol{\alpha}(\widehat{W_j})$, while the output $\widehat{y_j}$ is taken from the output values stored in $\widehat{W_j}$.
3. Output $\Lambda = \Lambda_1 || \ldots || \Lambda_{k^*}$.

The above experiment, which we denote as $\mathsf{Hyb}_{k^*+1}(x^*)$, only differs from the original experiment $\mathsf{Leak}_{\mathcal{C}}(\mathsf{Enc}(x^*))$ in when the leakage occurs. Namely, in $\mathsf{Leak}_{\mathcal{C}}$ first the circuit is computed entirely and then the leakage is computed, while Hyb_{k^*+1} computes the leakage immediately after the values on the wires are available. Therefore,

$$\mathsf{Leak}_{\mathcal{C}}(\mathsf{Enc}(x^*)) \equiv \mathsf{Hyb}_{k^*+1}(x^*).$$

Now we define, for $i \in [k^*]$, the following hybrid experiment $\mathsf{Hyb}_i(x^*)$.

1. Sample $\widehat{x^*} = \mathsf{Enc}(x^*)$ and $\widehat{x^*_?} = \mathsf{Blind}_{\boldsymbol{\rho}(x^*)}(\widehat{x^*})$.
2. For every $j \in [k^*]$, compute $\Lambda_j, \widehat{y_j}$ as follows.
   - If $j < i$, do the following.
     (a) For every input bundle of gadget $\widehat{g_j}$ that comes from an input of the circuit, place the corresponding value taken from the vector $\widehat{x^*}$.
     (b) For every input bundle of gadget $\widehat{g_j}$ that comes from a random gate of the circuit, sample uniform $r \in \mathbb{F}$ and place $\mathsf{Enc}(r)$ on the wires.
     (c) Since $\mathcal{C}$ is topologically sorted, every other input bundle of gadget $\widehat{g_j}$ already has a value, which has been computed in a previous iteration as $\widehat{y_{j'}}$ for some $j' < j$.
     (d) Let $\widehat{W_j} \in \mathbb{F}^{k_j}$ be the list of values on all the k_j wires of $\widehat{g_j}$.
     (e) The leakage Λ_j is set to be $\Lambda_j \coloneqq \boldsymbol{\alpha}(\widehat{W_j})$, while the output $\widehat{y_j}$ is taken from the output values stored in $\widehat{W_j}$.
     (f) Compute $y_j = \mathsf{Dec}(\widehat{y_j})$ and sample $\widehat{y_{j?}} = \mathsf{Blind}_{\boldsymbol{\rho}(y_j)}(\widehat{y_j})$.

     If $j \geq i$, the leakage and the output will be generated by the corresponding simulator $\mathsf{Sim}_{\widehat{g_j}}$. Namely, do the following.

(a) Let $\widehat{x_?}$ be possibly blinded the values on the input bundles of gadget $\widehat{g_j}$. Notice that, for every input bundle, there are only four possibilities:
   - the input bundle comes from input gates, in which case $\widehat{x_?}$ has already been computed among the circuit inputs $\widehat{x_?^*} = \mathsf{Blind}_{\boldsymbol{\rho}(x^*)}(\widehat{x^*})$;
   - the input bundle comes from random gates, in which case just sample uniform $r \in \mathbb{F}$ and sample $\widehat{x_?} = \mathsf{Blind}_{\boldsymbol{\rho}(r)}(\mathsf{Enc}(r))$;
   - the input bundle is an output bundle of a functional gadget $\widehat{g_{j'}}$ for $j' < i$, in which case we already computed $\widehat{x_?}$ as $\widehat{y_{j?}} = \mathsf{Blind}_{\boldsymbol{\rho}(y_j)}(\widehat{y_j})$ in a previous step;
   - the input bundle is an output bundle of a functional gadget $\widehat{g_{j'}}$ for $j' \geq i$ and $j' < j$, in which case $\widehat{x_?}$ has already been output by a previous simulator as $\widehat{y_{j?}}$.

   In any case, $\widehat{x_?}$ is always available.

(b) If some of the values of $\widehat{x_?}$ are $\perp$, set $\widehat{x_?} \leftarrow \perp$; otherwise, leave it unchanged. This is equivalent to convert the output of Blind into the output of Blind^*.

(c) Let $y = g(x)$, where $x = \mathsf{Dec}(\widehat{x})$ and $\widehat{x}$ is the collection of the input values computed as above. Notice that, even if $\widehat{x}$ is not available to the experiment, x is always available: indeed, the experiment knows the input x^* and the random coins sampled so far, therefore is able to deterministically reconstruct x.

(d) Probe $y_? \leftarrow \boldsymbol{\rho}(y)$.

(e) Run the simulator $(\Lambda_j, \widehat{y_?}) \leftarrow \mathsf{Sim}_{\widehat{g_i}}(\widehat{x_?}, y_?)$.

3. Output $\Lambda = \Lambda_1 || \ldots || \Lambda_{k^*}$.

Notice that, if we instantiate the above algorithm with $i = k^* + 1$, the part $j \geq i$ is never executed, and the part $j < i$ is exactly the same as in $\mathsf{Hyb}_{k^*+1}(x^*)$.

The following lemma says that the changes that we are introducing do not affect the final distribution of the leakage.

Lemma 4.3. *Let x^* be any input to $\mathcal{C}$. Then, for every $i \in [k^*]$,*

$$\mathsf{Hyb}_i(x^*) \equiv \mathsf{Hyb}_{i+1}(x^*)$$

The proof of this fact is a simple but long reduction, which can be found in Appendix A.4.

For the next step of the proof, we take a closer look at $\mathsf{Hyb}_1(x^*)$. Namely, since the branch $j < 1$ is never executed and all the real functional gadgets have been replaced with the respective simulators, the description of $\mathsf{Hyb}_1(x^*)$ looks as follows.

1. Sample $\widehat{x^*} = \mathsf{Enc}(x^*)$ and $\widehat{x_?^*} = \mathsf{Blind}_{\boldsymbol{\rho}(x^*)}(\widehat{x^*})$.

2. For every $j \in [k^*]$, compute $\Lambda_j, \widehat{y_j}$ as follows.

   (a) Let $\widehat{x_?}$ be the possibly blinded values on the input bundles of gadget $\widehat{g_j}$. Notice that, for every input bundle, there are only three possibilities:
   - the input bundle comes from input gates, in which case $\widehat{x_?}$ has already been computed among the circuit inputs $\widehat{x_?^*} = \mathsf{Blind}_{\boldsymbol{\rho}(x^*)}(\widehat{x^*})$;
   - the input bundle comes from random gates, in which case just sample uniform $r \in \mathbb{F}$ and sample $\widehat{x_?} = \mathsf{Blind}_{\boldsymbol{\rho}(r)}(\mathsf{Enc}(r))$;
   - the input bundle is an output bundle of a functional gadget $\widehat{g_{j'}}$ for $j' < j$, in which case $\widehat{x_?}$ has already been output by a previous simulator as $\widehat{y_{j?}}$.

In any case, $\widehat{x_?}$ is always available.

(b) If some of the values of $\widehat{x_?}$ are $\perp$, set $\widehat{x_?} \leftarrow \perp$; otherwise, leave it unchanged. This is equivalent to convert the output of Blind into the output of Blind^*.

(c) Let $y = g(x)$, where $x = \mathsf{Dec}(\widehat{x})$ and $\widehat{x}$ is the collection of the input values computed as above. Notice that, even if $\widehat{x}$ is not available to the experiment, x is always available: indeed, the experiment knows the input x^* and the random coins sampled so far, therefore is able to deterministically reconstruct x.

(d) Probe $y_? \leftarrow \boldsymbol{\rho}(y)$.

(e) Run the simulator $(\Lambda_j, \widehat{y_?}) \leftarrow \mathsf{Sim}_{\widehat{g_i}}(\widehat{x_?}, y_?)$.

3. Output $\Lambda = \Lambda_1 || \ldots || \Lambda_{k^*}$.

We define the last hybrid experiment $\mathsf{Hyb}_0(x^*)$ as follows. Here we underline the differences between Hyb_1 and Hyb_0.

1. Run $\mathcal{C}(x^*)$ and sample $W_? := \boldsymbol{\rho}(\mathbf{W}(x^*))$.

2. For every $j \in [k^*]$, compute $\Lambda_j, \widehat{y_j}$ as follows.

   (a) Let $\widehat{x_?}$ be possibly blinded the values on the input bundles of gadget $\widehat{g_j}$. Notice that, for every input bundle, there are only three possibilities:

   - the input bundle comes from input gates, in which case we set $\widehat{x_?} = \mathsf{Enc}(x)$ if the random probing on the original input x was successful and $\widehat{x_?} = \perp$ otherwise.
   - the input bundle comes from random gates, in which case just sample uniform $r \in \mathbb{F}$ and sample $\widehat{x_?} = \mathsf{Blind}_{\boldsymbol{\rho}(r)}(\mathsf{Enc}(r))$;
   - the input bundle is an output bundle of a functional gadget $\widehat{g_{j'}}$ for $j' < j$, in which case $\widehat{x_?}$ has already been output by a previous simulator as $\widehat{y_{j?}}$.

   In any case, $\widehat{x_?}$ is always available.

   (b) If some of the values of $\widehat{x_?}$ are $\perp$, set $\widehat{x_?} \leftarrow \perp$; otherwise, leave it unchanged. This is equivalent to convert the output of Blind into the output of Blind^*.

   (c) Let $y_?$ be the random probing of the output y of the original gate g_j, as already sampled in $W_?$.

   (d) Run the simulator $(\Lambda_j, \widehat{y_?}) \leftarrow \mathsf{Sim}_{\widehat{g_j}}(\widehat{x_?}, y_?)$.

3. Output $\Lambda = \Lambda_1 || \ldots || \Lambda_{k^*}$.

Lemma 4.4. *Let x^* be any input to $\mathcal{C}$. Then, for every $i \in [k^*]$,*

$$\mathsf{Hyb}_0(x^*) \equiv \mathsf{Hyb}_1(x^*)$$

Proof. The only differences between the two hybrid experiments are when and how the random probing is sampled. In particular, Hyb_1 runs the transformed circuit $\widehat{\mathcal{C}}$, decodes the values on the wires and then samples random and average probing leakage. On the other hand, Hyb_0 runs the original circuit $\mathcal{C}$, computes the random probing and then only samples the average probing when needed, i.e., for the input bundles that are part of the input of the circuit or for the input bundles that come from random gates. Finally, the average probing samples in Hyb_0 are identically distributed to the ones of Hyb_1, since both are identically distributed to the samples in the real case. The lemma follows.

Finally, notice that the only use that $\mathsf{Hyb}_0(x^*)$ makes of x^* is to compute the values on the wires of the original circuit, which is then only used to sample the random probing $W_? := \boldsymbol{\rho}(\mathbf{W}(x^*))$. We can now extract this procedure outside the hybrid experiment and directly give the value $W_?$ to the experiment; the simulator is then defined as receiving the random probing $W_?$ from the wires and then running exactly as $\mathsf{Hyb}_0(x^*)$ from the second step on. Therefore, the following holds.

Corollary 4.5. *Let x^* be any input to $\mathcal{C}$. Then, for every $i \in [k^*]$,*

$$\mathsf{Hyb}_0(x^*) \equiv \mathsf{Sim}_{\mathcal{C}}(\boldsymbol{\rho}(\mathbf{W}(x^*))).$$

By putting everything together, we are finally able to prove Theorem 4.2.

Proof. As observed at the beginning, $\mathsf{Hyb}_{k^*+1}(x^*)$ is just a syntactic change from $\mathsf{Leak}_{\mathcal{C}}(\mathsf{Enc}(x^*))$ and is, otherwise, identical. Hence:

$$\begin{aligned}
\mathsf{Leak}_{\mathcal{C}}(\mathsf{Enc}(x^*)) &\equiv \mathsf{Hyb}_{k^*+1}(x^*) \\
&\equiv \mathsf{Hyb}_{k^*}(x^*) && \text{(By applying Lemma 4.3)} \\
&\equiv \mathsf{Hyb}_1(x^*) && \text{(By repeatedly applying Lemma 4.3)} \\
&\equiv \mathsf{Hyb}_0(x^*) && \text{(By applying Lemma 4.4)} \\
&\equiv \mathsf{Sim}_{\mathcal{C}}(\boldsymbol{\rho}(\mathbf{W}(x^*))). && \text{(By applying Corollary 4.5)}
\end{aligned}$$

This concludes the proof.

5 Conclusions and open problems

In this work, we presented the first generic compiler that compiles any ε-random probing resilient circuit $\mathcal{C}$ into a δ-average probing resilient circuit $\widehat{\mathcal{C}}$, as long as

$$35\delta < \varepsilon^2.$$

Our compiler takes a circuit $\mathcal{C}$ with W wires and G gates and produces a new circuit $\widehat{\mathcal{C}}$ with $W' \leq 2W + 27G$ wires and $G' \leq 22G$ gates. Notice that W' and G' are maximum when $\mathcal{C}$ only contains multiplication gates, and in practical settings, we will usually be below such bounds.

As an immediate application, we are able to achieve the first random-probing to noisy-leakage compiler where the loss in the noise parameter does not depend on the size of the underlying field. We briefly recall below the definition of noisy-leakage from [PR13] and the main result of [DFS15].

Definition 5.1 ([PR13])**.** *Let $\varepsilon \in (0,1)$ be a parameter. A (randomized) function $\nu : \mathbb{F} \to \Omega$ is* ε-noisy leakage *if*

$$\Delta\left((\boldsymbol{\nu}(\mathbf{x}), \mathbf{x}), (\boldsymbol{\nu}(\mathbf{x}), \mathbf{x}')\right) \leq \varepsilon,$$

where $\mathbf{x}, \mathbf{x}'$ are uniform over $\mathbb{F}$.

Theorem 5.2 ([DFS15])**.** *For every $\varepsilon \in (0,1)$, and every circuit $\mathcal{C}$, $\mathcal{C}$ is ε-average-probing to ε-noisy-leakage simulatable.*

In the above, we formalized the result of [DFS15] using Definition 4.1. Notice that [DFS15] does not use a compiler to transform the circuit $\mathcal{C}$, therefore the circuit is simulatable from $\mathcal{C}$ itself. The following corollary is an immediate consequence of the above and Theorem 4.2. It implies that circuits transformed by our compiler from Section 4 are δ-noisy secure, if the original circuit is secure against ϵ-random probing leakage for parameters ϵ and δ as given in the corollary below.

Corollary 5.3. *Let $\varepsilon, \delta \in (0,1)$ be parameters such that*

$$35\delta < \varepsilon^2.$$

Let $\mathcal{C}$ be any circuit and let $\widehat{\mathcal{C}}$ be the circuit transformed according to the compiler in Section 4. If $\mathcal{C}$ is ε-random-probing resilient, then $\widehat{\mathcal{C}}$ is δ-average-probing to δ-noisy-leakage simulatable.

One limitation of our current analysis is a tightness loss in the noise parameters. Concretely, if the circuit C is ϵ-random probing secure, then $\widehat{\mathcal{C}}$ is δ-noisy leakage resilient where $35\delta < \varepsilon^2$. The quadratic loss is a consequence of the fact that our gadget simulator in Section 3 only simulates the leakage when random-probing is successful on *all* the inputs of the gate in the original circuit. This happens with probability ε^2 for gates with 2 inputs. With our current techniques, this is somewhat inherent due to the nature of average-probing. In particular, the simulator cannot simulate the leakage in a consistent way by just receiving partial leakage from one of the inputs (e.g., because the simulator may incorrectly assume some values being on the wires that contradict previous leakage from the circuit). We leave it as an important open problem to further improve our result and eliminate the quadratic loss.

Acknowledgements

Stefan Dziembowski and Gianluca Brian were supported by an NCN Grant 2019/35/B/ST6/04138 and by the Copernicus Awards (agreement no. COP/01/ 2020). This work has been partially funded by the German Research Foundation (DFG) CRC 1119 CROSSING (project S7), the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, by the ERC Grant 101044770 (CRYPTOLAYER) and the Copernicus Award (INST 18989/419-1).

References

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. pages 586–615, 2016.

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. pages 427–455, 2018.

[Ajt11] Miklós Ajtai. Secure computation with information leaking to an adversary. pages 715–724, 2011.

[BCG+22] 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. Cryptology ePrint Archive, Report 2022/1738, 2022. https://eprint.iacr.org/2022/1738.

[BCPZ16] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. pages 23–39, 2016.

[BMRT22] Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. IronMask: Versatile verification of masking security. pages 142–160, 2022.

[BRT21] Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. pages 313–343, 2021.

[BRTV21] Sonia Belaïd, Matthieu Rivain, Abdul Rahman Taleb, and Damien Vergnaud. Dynamic random probing expansion with quasi linear asymptotic complexity. pages 157–188, 2021.

[CFOS21] Gaëtan Cassiers, Sebastian Faust, Maximilian Orlt, and François-Xavier Standaert. Towards tight random probing security. pages 185–214, 2021.

[CGZ20] Jean-Sébastien Coron, Aurélien Greuet, and Rina Zeitoun. Side-channel masking with pseudo-random generator. pages 342–375, 2020.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. pages 398–412, 1999.

[CRZ18] Jean-Sébastien Coron, Franck Rondepierre, and Rina Zeitoun. High order masking of look-up tables with common shares. 2018(1):40–72, 2018. https://tches.iacr.org/index.php/TCHES/article/view/832.

[CS19] Gaëtan Cassiers and François-Xavier Standaert. Towards globally optimized masking: From low randomness to low noise rate. 2019(2):162–198, 2019. https://tches.iacr.org/index.php/TCHES/article/view/7389.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. pages 423–440, 2014.

[DFS15] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. pages 159–188, 2015.

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

[GJR18] Dahmun Goudarzi, Antoine Joux, and Matthieu Rivain. How to securely compute with noisy leakage in quasilinear complexity. pages 547–574, 2018.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. pages 463–481, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. pages 388–397, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. pages 104–113, 1996.

[MPR07] Ueli M. Maurer, Krzysztof Pietrzak, and Renato Renner. Indistinguishability amplification. pages 130–149, 2007.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a Rényi day. pages 683–712, 2019.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. pages 142–159, 2013.

A Missing proofs

In this appendix, we put all the proofs that we omitted for space reasons.

A.1 Proof of Lemma 2.1

Proof. We have that, for all $z \in \mathbb{F} \setminus \{0\}$,

$$\begin{aligned}
\Pr[\mathbf{xy} = z] &= \sum_{x \in \mathbb{F} \setminus \{0\}} \Pr[\mathbf{xy} = z \wedge \mathbf{x} = x] \\
&= \sum_{x \in \mathbb{F} \setminus \{0\}} \Pr[\mathbf{x} = x] \Pr\left[\mathbf{y} = \frac{z}{x}\right] \\
&= \sum_{x \in \mathbb{F} \setminus \{0\}} \frac{1}{|\mathbb{F}|} \frac{1}{|\mathbb{F}|} \\
&= \frac{1}{|\mathbb{F}|} - \frac{1}{|\mathbb{F}|^2},
\end{aligned}$$

while, for $z = 0$,

$$\begin{aligned}
\Pr[\mathbf{xy} = 0] &= \sum_{x \in \mathbb{F} \setminus \{0\}} \Pr[\mathbf{xy} = 0 \wedge \mathbf{x} = x] + \Pr[\mathbf{xy} = 0 \wedge \mathbf{x} = 0] \\
&= \sum_{x \in \mathbb{F} \setminus \{0\}} \Pr[\mathbf{x} = x] \Pr[\mathbf{y} = 0] + \Pr[\mathbf{x} = 0] \\
&= \sum_{x \in \mathbb{F} \setminus \{0\}} \frac{1}{|\mathbb{F}|} \frac{1}{|\mathbb{F}|} + \frac{1}{|\mathbb{F}|} \\
&= \frac{2}{|\mathbb{F}|} - \frac{1}{|\mathbb{F}|^2}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Delta(\mathbf{xy}, \mathbf{U}) &= \frac{1}{2} \sum_{z \in \mathbb{F}} |\Pr[\mathbf{xy} = z] - \Pr[\mathbf{U} = z]| \\
&= \frac{1}{2} \sum_{z \in \mathbb{F} \setminus \{0\}} |\Pr[\mathbf{xy} = z] - \Pr[\mathbf{U} = z]| \\
&\quad + \frac{1}{2} |\Pr[\mathbf{xy} = 0] - \Pr[\mathbf{U} = 0]| \\
&= \frac{1}{2} \sum_{z \in \mathbb{F} \setminus \{0\}} \left| \frac{1}{|\mathbb{F}|} - \frac{1}{|\mathbb{F}|^2} - \frac{1}{|\mathbb{F}|} \right| \\
&\quad + \frac{1}{2} \left| \frac{2}{|\mathbb{F}|} - \frac{1}{|\mathbb{F}|^2} - \frac{1}{|\mathbb{F}|} \right| \\
&= \frac{1}{2} \sum_{z \in \mathbb{F} \setminus \{0\}} \frac{1}{|\mathbb{F}|^2} + \frac{1}{2} \left| \frac{1}{|\mathbb{F}|} - \frac{1}{|\mathbb{F}|^2} \right| \\
&= \frac{|\mathbb{F}| - 1}{2|\mathbb{F}|^2} + \frac{|\mathbb{F}| - 1}{2|\mathbb{F}|^2} \leq \frac{1}{|\mathbb{F}|},
\end{aligned}$$

where the first equality holds by definition of Δ, in the second equality we just split the sum in $z = 0$ and $z \neq 0$, the third equality follows from the above derivations, and the remaining equalities and inequalities are just calculations.

A.2 Proof of Lemma 2.4

Proof. Without loss of generality, we can consider the first values $\mathbf{x}_1, \ldots, \mathbf{x}_{k'}$ to be the γ-close-to-uniform ones and all the other ones to be the uniform ones. Then, the proof proceeds by applying the union bound to the complementary event $\boldsymbol{\alpha}(\mathbf{x}) \neq \perp$. Indeed,

$$\begin{aligned}
\Pr\left[\boldsymbol{\alpha}(\mathbf{x}) \neq \perp\right] &= \Pr\left[\bigvee_{i\in[k]} \boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i\right] && (14)\\
&\leq \sum_{i\in[k]} \Pr\left[\boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i\right] \\
&= \sum_{i\in[k']} \Pr\left[\boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i\right] + \sum_{i\in[k]\setminus[k']} \Pr\left[\boldsymbol{\alpha}(\mathbf{U}) = \mathbf{U}\right] \\
&= \sum_{i\in[k']} \Pr\left[\boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i\right] + \left(k - k'\right)\delta. && (15)
\end{aligned}$$

In the above, the first inequality comes from the union bound, the subsequent equality comes from the partition in γ-close-to-uniform values and uniform values and the last equality comes from the definition of δ-average probing. Now we apply the definition of close-to-uniform to obtain a similar bound on the left part. Namely, for every $i \in [k']$,

$$\begin{aligned}
\Pr\left[\boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i\right] &= \sum_{x\in\mathbb{F}} \Pr\left[\mathbf{x}_i = x\right] \Pr\left[\boldsymbol{\alpha}(x) = x\right] \\
&\leq \sum_{x\in\mathbb{F}} \left(\Pr\left[\mathbf{U} = x\right] + \gamma\right) \Pr\left[\boldsymbol{\alpha}(x) = x\right] \\
&= \sum_{x\in\mathbb{F}} \left(1 + \gamma|\mathbb{F}|\right) \Pr\left[\mathbf{U} = x\right] \Pr\left[\boldsymbol{\alpha}(x) = x\right] \\
&= \left(1 + \gamma|\mathbb{F}|\right) \Pr\left[\boldsymbol{\alpha}(\mathbf{U}) = \mathbf{U}\right] \\
&= \left(1 + \gamma|\mathbb{F}|\right)\delta.
\end{aligned}$$

In the above, the first equality comes from splitting the event $\boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i$ into disjoint events and applying the conditional probability (recall that $\boldsymbol{\alpha}(x)$ is a random variable that only depends on the value x), the inequality comes from the definition of being γ-close-to-uniform, and the subsequent equalities come from the fact that $\Pr\left[\mathbf{U} = x\right] = \frac{1}{|\mathbb{F}|}$, from reversing the conditional probability and from the definition of average probing. By applying the above to Eq. (15), we can continue the derivation of Eq. (14).

$$\begin{aligned}
\Pr\left[\boldsymbol{\alpha}(\mathbf{x}) \neq \perp\right] &\leq \sum_{i\in[k']} \Pr\left[\boldsymbol{\alpha}(\mathbf{x}_i) = \mathbf{x}_i\right] + \left(k - k'\right)\delta \\
&\leq k' \cdot \left(1 + \gamma|\mathbb{F}|\right)\delta + \left(k - k'\right)\delta \\
&= \left(k + k'\gamma|\mathbb{F}|\right)\delta.
\end{aligned}$$

The lemma follows by applying again the rules for complementary events:

$$\begin{aligned}
\Pr\left[\boldsymbol{\alpha}(\mathbf{x}) = \perp\right] &= 1 - \Pr\left[\boldsymbol{\alpha}(\mathbf{x}) \neq \perp\right] \\
&\geq 1 - \left(k + k'\gamma|\mathbb{F}|\right)\delta.
\end{aligned}$$

A.3 Proof of Lemma 3.5

Proof. We start by lower-bounding the right-hand term of Eq. (1). Namely, we can write $\mathsf{Real}_{\widehat{g}}(\widehat{x})$ as $(\boldsymbol{\Lambda}, \widehat{\mathbf{y}})$, and we can further split $\boldsymbol{\Lambda}$ into the leakage from the internal wires $\boldsymbol{\Lambda}_{\mathsf{int}}$ and the leakage from the output wires $\boldsymbol{\alpha}(\widehat{\mathbf{y}})$:

$$\begin{aligned}\Pr\left[\mathsf{Real}_{\widehat{g}}(\widehat{x}) = (\perp, \widehat{y})\right] &= \Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\Lambda}_{\mathsf{int}} = \perp, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right] \\ &= \Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right] \Pr\left[\boldsymbol{\Lambda}_{\mathsf{int}} = \perp \,|\, \widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right].\end{aligned} \tag{16}$$

Notice that, in the last probability of Eq. (16), we are conditioning on two events, namely $\widehat{\mathbf{y}} = \widehat{y}$ and $\boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp$; however, the first event allows to replace $\widehat{\mathbf{y}}$ with $\widehat{y}$ in the second event, which becomes $\boldsymbol{\alpha}(\widehat{y}) = \perp$. Furthermore, notice that, once $\widehat{y}$ is fixed, the probability of $\boldsymbol{\alpha}(\widehat{y}) = \perp$ only depends on $\widehat{y}$ and is independent of anything else; in particular, it is independent of $\boldsymbol{\Lambda}_{\mathsf{int}}$, and hence can be removed from the condition:

$$\Pr\left[\boldsymbol{\Lambda}_{\mathsf{int}} = \perp \,|\, \widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{y}) = \perp\right] = \Pr\left[\boldsymbol{\Lambda}_{\mathsf{int}} = \perp \,|\, \widehat{\mathbf{y}} = \widehat{y}\right].$$

Now we make $\boldsymbol{\Lambda}_{\mathsf{int}}$ even more explicit. Let $(\mathbf{w}_1, \ldots, \mathbf{w}_{k-\ell n})$ be the distribution of the values on the internal wires of $\widehat{g}$ upon input $\widehat{x}$. Then,

$$\begin{aligned}&\Pr\left[\boldsymbol{\Lambda}_{\mathsf{int}} = \perp \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] && \\ &= \Pr\left[\bigwedge_{i\in[k-\ell n]} \boldsymbol{\alpha}(\mathbf{w}_i) = \perp \,\middle|\, \widehat{\mathbf{y}} = \widehat{y}\right] && \text{(By making } \boldsymbol{\Lambda}_{\mathsf{int}} \text{ explicit)} \\ &= 1 - \Pr\left[\bigvee_{i\in[k-\ell n]} \boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i \,\middle|\, \widehat{\mathbf{y}} = \widehat{y}\right] && \text{(By complementing the event)} \\ &\geq 1 - \sum_{i\in[k-\ell n]} \Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i \,|\, \widehat{\mathbf{y}} = \widehat{y}\right]. && \text{(By the union bound)}\end{aligned} \tag{17}$$

Thanks to the properties of $\boldsymbol{\alpha}$ and the gadget, we can simplify the term inside the sum. Indeed,

$$\begin{aligned}\Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] &= \sum_{w\in\mathbb{F}} \Pr\left[\mathbf{w}_i = w \wedge \boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] \\ &= \sum_{w\in\mathbb{F}} \Pr\left[\mathbf{w}_i = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] \Pr\left[\boldsymbol{\alpha}(w) = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right],\end{aligned} \tag{18}$$

respectively by splitting into disjoint events and then by applying conditional probability. Then we can apply the output-independence property, which states that, for every internal wire of the gadget,

$$\Pr\left[\mathbf{w}_i = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] = \Pr\left[\mathbf{w}_i = w\right],$$

and we can also apply, again, the fact that the output of $\boldsymbol{\alpha}$ only depends on its input:

$$\Pr\left[\boldsymbol{\alpha}(w) = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] = \Pr\left[\boldsymbol{\alpha}(w) = w\right].$$

These two facts allow to rewrite the last sum in Eq. (18) as

$$\begin{aligned}\sum_{w\in\mathbb{F}} \Pr\left[\mathbf{w}_i = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] \Pr\left[\boldsymbol{\alpha}(w) = w \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] &= \sum_{w\in\mathbb{F}} \Pr\left[\mathbf{w}_i = w\right] \Pr\left[\boldsymbol{\alpha}(w) = w\right] \\ &= \sum_{w\in\mathbb{F}} \Pr\left[\mathbf{w}_i = w \wedge \boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i\right] \\ &= \Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i\right],\end{aligned}$$

which, together with the beginning of Eq. (18), allows to conclude that

$$\Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] = \Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i\right]. \tag{19}$$

Now we can continue the derivation in Eq. (17):

$$\begin{aligned}
&\Pr\left[\boldsymbol{\Lambda}_{\mathsf{int}} = \perp \,|\, \widehat{\mathbf{y}} = \widehat{y}\right] && (20)\\
&\geq 1 - \sum_{i \in [k-\ell n]} \Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i \,|\, \widehat{\mathbf{y}} = \widehat{y}\right], && \text{(From Eq. (17))}\\
&= 1 - \sum_{i \in [k-\ell n]} \Pr\left[\boldsymbol{\alpha}(\mathbf{w}_i) = \mathbf{w}_i\right], && \text{(From Eq. (19))}\\
&\geq 1 - \left(k - \ell n + k'\gamma|\mathbb{F}|\right)\delta, && \text{(By Lemma 2.4)}
\end{aligned}$$

where the last step is possible because $\widehat{g}$ is (k, k', γ)-close-to-uniform and has fan-out ℓn, therefore the set of the internal wires is $(k - \ell n, k', \gamma)$-close-to-uniform. Finally, we can lower-bound the right-hand term of Eq. (1):

$$\begin{aligned}
&\frac{\Pr\left[\mathsf{Real}_g(\widehat{x}) = (\perp, \widehat{y})\right]}{1 - (k + k'\gamma|\mathbb{F}|)\delta} && (21)\\
&= \frac{\Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right] \Pr\left[\boldsymbol{\Lambda}_{\mathsf{int}} = \perp \,|\, \widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right]}{1 - (k + k'\gamma|\mathbb{F}|)\delta} && \text{(From Eq. (16))}\\
&\geq \frac{\Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right] \left(1 - \left(k - \ell n + k'\gamma|\mathbb{F}|\right)\delta\right)}{1 - (k + k'\gamma|\mathbb{F}|)\delta} && \text{(From Eq. (20))}\\
&\geq \frac{\Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \perp\right]}{1 - \ell n\delta},
\end{aligned}$$

where the last step comes from the fact that

$$\begin{aligned}
&1 - \left(k + k'\gamma|\mathbb{F}|\right)\delta\\
&= (1 - \ell n\delta)\left(1 - \left(k - \ell n + k'\gamma|\mathbb{F}|\right)\delta\right) - \ell n\delta \cdot \left(k - \ell n + k'\gamma|\mathbb{F}|\right)\delta\\
&\leq (1 - \ell n\delta)\left(1 - \left(k - \ell n + k'\gamma|\mathbb{F}|\right)\delta\right)
\end{aligned}$$

and therefore, rearranging the terms,

$$\frac{1 - \left(k - \ell n + k'\gamma|\mathbb{F}|\right)\delta}{1 - \left(k + k'\gamma|\mathbb{F}|\right)\delta} \geq \frac{1}{1 - \ell n\delta}.$$

The last step of the proof comes from Lemma 2.4, from which we get

$$\Pr\left[\boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right] \geq 1 - \ell n\delta,$$

or, rearranging the terms,

$$\frac{1}{1 - \ell n\delta} \geq \frac{1}{\Pr\left[\boldsymbol{\alpha}(\mathsf{Enc}(y)) = \perp\right]}.$$

By plugging the above into Eq. (21), we get

$$
\begin{aligned}
&\frac{\Pr\left[\mathsf{Real}_g(\widehat{x}) = (\bot, \widehat{y})\right]}{1 - (k + k'\gamma|\mathbb{F}|)\delta} \\
&\geq \frac{\Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \bot\right]}{1 - \ell n \delta} && \text{(From Eq. (21))} \\
&\geq \frac{\Pr\left[\widehat{\mathbf{y}} = \widehat{y}, \boldsymbol{\alpha}(\widehat{\mathbf{y}}) = \bot\right]}{\Pr\left[\boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot\right]} && \text{(From the above)} \\
&= \frac{\Pr\left[\mathsf{Enc}(y) = \widehat{y}, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot\right]}{\Pr\left[\boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot\right]} && \text{(By definition of output-independence)} \\
&= \Pr\left[\mathsf{Enc}(y) = \widehat{y} \,|\, \boldsymbol{\alpha}(\mathsf{Enc}(y)) = \bot\right], && \text{(By definition of conditional probability)}
\end{aligned}
$$

which concludes the proof.

A.4 Proof of Lemma 4.3

Proof. By reduction to the simulatability of gadget $\widehat{g}_i$. Assume that $\mathsf{Hyb}_i(x^*)$ and $\mathsf{Hyb}_{i+1}(x^*)$ are not identically distributed. Consider the following reduction, consisting of two algorithms $\mathsf{R}_1, \mathsf{R}_2$. Algorithm R_1 takes as input x^* and computes the following.

1. Sample $\widehat{x^*} = \mathsf{Enc}(x^*)$ and $\widehat{x^*_?} = \mathsf{Blind}_{\boldsymbol{\rho}(x^*)}(\widehat{x^*})$.
2. For every $j \in [i-1]$, compute $\Lambda_j, \widehat{y_j}$ as follows.
   (a) For every input bundle of gadget $\widehat{g_j}$ that comes from an input of the circuit, place the corresponding value taken from the vector $\widehat{x^*}$.
   (b) For every input bundle of gadget $\widehat{g_j}$ that comes from a random gate of the circuit, sample uniform $r \in \mathbb{F}$ and place $\mathsf{Enc}(r)$ on the wires.
   (c) Since $\mathcal{C}$ is topologically sorted, every other input of gadget $\widehat{g_j}$ has already been computed in a previous iteration.
   (d) Let $\widehat{W_j} \in \mathbb{F}^{k_j}$ be the list of values on all the k_j wires of $\widehat{g_j}$.
   (e) The leakage Λ_j is set to be $\Lambda_j \coloneqq \boldsymbol{\alpha}(\widehat{W_j})$, while the output $\widehat{y_j}$ is taken from the output values stored in $\widehat{W_j}$.
   (f) Compute $y_j = \mathsf{Dec}(\widehat{y_j})$ and sample $\widehat{y_{j?}} = \mathsf{Blind}_{\boldsymbol{\rho}(y_j)}(\widehat{y_j})$.
3. Let st be the internal state of R containing everything that has been computed so far.
4. Let $\widehat{x}$ be the input to gadget $\widehat{g_i}$, as computed in the previous steps, and let $x = \mathsf{Dec}(\widehat{x})$ and $y = g_i(x)$ (recall that g_i is a deterministic gadget, hence y is uniquely defined).
5. Let $\Lambda = \Lambda_1 || \ldots || \Lambda_{i-1}$.
6. Output $(\mathsf{st}, \widehat{x}, x, y_?, \Lambda)$.

Let $(\Lambda^*, \widehat{y_?}^*)$ be sampled either from the real distribution $(\boldsymbol{\Lambda}, \mathsf{Blind}_{y_?}(\widehat{\mathbf{y}}))$, where $(\boldsymbol{\Lambda}, \mathbf{y}) = \mathsf{Real}_{\widehat{g}}(\widehat{x})$, or from the simulator $\mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}^*_{x_?}(\widehat{x}), y_?\right)$, where $x_? \xleftarrow{\$} \boldsymbol{\rho}(x)$; then, let $\Lambda \leftarrow \Lambda || \Lambda^*$.

Algorithm R_2 takes as input $\mathsf{st}, \Lambda, \widehat{y_?}^*$ and computes the following.

1. For every $j \in [k^*] \setminus [i]$, compute $\Lambda_j, \widehat{y_j}$ as follows.

(a) Let $\widehat{x_?}$ be the possibly blinded values on the input bundles of gadget $\widehat{g_j}$. Notice that, for every input bundle, there are five possibilities:

- the input bundle comes from input gates, in which case $\widehat{x_?}$ has already been computed among the circuit inputs $\widehat{x_?^*} = \mathsf{Blind}_{\boldsymbol{\rho}(x^*)}(\widehat{x^*})$ by R_1;
- the input bundle comes from random gates, in which case just sample uniform $r \in \mathbb{F}$ and sample $\widehat{x_?} = \mathsf{Blind}_{\boldsymbol{\rho}(r)}(\mathsf{Enc}(r))$;
- the input bundle is an output bundle of a functional gadget $\widehat{g_{j'}}$ for $j' < i$, in which case we already computed $\widehat{x_?}$ as $\widehat{y_{j?}} = \mathsf{Blind}_{\boldsymbol{\rho}(y_j)}(\widehat{y_j})$ in a previous step;
- the input bundle is an output bundle of a functional gadget $\widehat{g_{j'}}$ for $j' > i$ and $j' < j$, in which case $\widehat{x_?}$ has already been output by a previous simulator as $\widehat{y_{j?}}$.
- the input bundle is an output bundle of the functional gadget $\widehat{g_i}$, in which case $\widehat{x_?}$ has been received as input as $\widehat{y_?}^*$.

In any case, $\widehat{x_?}$ is always available.

(b) If some of the values of $\widehat{x_?}$ are $\perp$, set $\widehat{x_?} \leftarrow \perp$; otherwise, leave it unchanged. This is equivalent to convert the output of Blind into the output of Blind^*.

(c) Let $y = g(x)$, where $x = \mathsf{Dec}(\widehat{x})$ and $\widehat{x}$ is the collection of the input values computed as above. Notice that, even if $\widehat{x}$ is not available to the experiment, x is always available: indeed, the experiment knows the input x^* and the random coins sampled so far, therefore is able to deterministically reconstruct x.

(d) Probe $y_? \leftarrow \boldsymbol{\rho}(y)$.

(e) Run the simulator $(\Lambda_j, \widehat{y_?}) \leftarrow \mathsf{Sim}_{\widehat{g_i}}(\widehat{x_?}, y_?)$.

2. Output $\Lambda \leftarrow \Lambda || \Lambda_{i+1} || \dots || \Lambda_{k^*}$.

Notice that the reduction is perfect. In particular, if $(\Lambda^*, \widehat{y_?}^*)$ comes from the real distribution, the reduction gets the real Λ^* and the value $\widehat{y_?}^*$ is computed starting from the real $\widehat{y}$ and applying the function $\mathsf{Blind}_{\boldsymbol{\rho}(y)}$, which is exactly how Λ_i and $\widehat{y_?}^*$ would be computed in hybrid $\mathsf{Hyb}_{i+1}(x^*)$. On the other hand, if $(\Lambda^*, \widehat{y_?}^*)$ comes from the simulator, the step $(\widehat{x_?}, y_?) \leftarrow \left(\mathsf{Blind}^*_{\boldsymbol{\rho}(x)}(\widehat{x}), y_?\right)$ that would appear in hybrid $\mathsf{Hyb}_i(x^*)$ appears instead in the argument passed to the simulator, and the simulator outputs Λ_i and $\widehat{y_?}^*$ exactly as in hybrid $\mathsf{Hyb}_{i+1}(x^*)$. Since everything else is the same, the assumption that $\mathsf{Hyb}_i(x^*)$ and $\mathsf{Hyb}_{i+1}(x^*)$ are not identically distributed implies that the real distribution $(\boldsymbol{\Lambda}(\widehat{x}), \mathsf{Blind}_{y_?}(\widehat{\mathbf{y}}(\widehat{x})))$ and the simulator $\mathsf{Sim}_{\widehat{g}}\left(\mathsf{Blind}^*_{y_?}(\widehat{x}), \boldsymbol{\rho}(y)\right)$ are not identically distributed, which contradicts Theorem 3.6. The lemma follows.