1

Masking Security Proofs

Sonia BELAÏD[1]

[1] *CryptoExperts*

1.1. Introduction

The security of cryptographic implementations can be assessed following two different methods. On the one hand, cryptographic implementations can be confronted with concrete side-channel attacks directly on embedded devices (as presented in previous chapters). While their security would thus be directly evaluated on the target device by exploiting the actual leakage that the attacker will have access to, this approach remains empirical, not portable and does not yield measurable security levels (*e.g.*, attacks might be missing). On the other hand, the security of cryptographic implementations can be formally proven based on abstract leakage models which aim to define the attacker's capabilities. This second approach advantageously makes it possible to measure concrete security levels. Although the leakage models may be too far removed from reality to yield practical security, the community works on improving them with respect to concrete devices in order to connect both sides. In this chapter, we will investigate the different leakage models and the related masking security proofs.

Several leakage models have been introduced to reason on the security of masked cryptographic implementations against side-channel attacks. In this chapter, we selected four of them that we believe are the most widely used. The *noisy leakage model*

Title of the book,
coordinated by Firstname LASTNAME.

is often considered to be the closest to the reality of embedded devices by assuming that the adversary gets a noisy function of each manipulated data. Given the difficulty of building security proofs in this model, most masking schemes are proven to be secure in the *probing model*, yet further from reality. Several leakage models stand in the middle. Among them, the *robust probing model* supplements the probing model by additionally considering the leakage of physical defaults and the *random probing model* is closer to the noisy leakage model (with a tight reduction) and makes it possible to build security proofs.

The next section introduces all the prerequisites that we need on circuits, sharings, gadgets, and compilers to accurately explain the security proofs in the subsequent parts. Section 1.3 intuitively and formally introduces the probing model. Its extension, the robust probing model, is discussed in Section 1.4. The more realistic noisy leakage and random probing models are then defined in Section 1.5. Finally, the last section is dedicated to the composition of secure gadgets in these models.

1.2. Preliminaries

In this chapter, $\mathbb{K}$ shall denote a finite field and $\mathbb{F}_q$ the finite field with q elements. Any two probability distributions D_1 and D_2 are said ε-close, denoted $D_1 \approx_\varepsilon D_2$, if their statistical distance is upper bounded by ε, that is

$$\mathtt{SD}(D_1; D_2) := \frac{1}{2} \sum_x |p_{D_1}(x) - p_{D_2}(x)| \leq \varepsilon,$$

where $p_{D_1}(\cdot)$ and $p_{D_2}(\cdot)$ denote the probability mass functions of D_1 and D_2.

1.2.1. *Circuits*

An *arithmetic circuit* over $\mathbb{K}$ is a labeled directed acyclic graph whose edges are *wires* and vertices are *arithmetic gates* processing operations over $\mathbb{K}$. A *randomized arithmetic circuit* is additionally equipped with random gates of fan-in (*i.e.*, number of inputs) 0 and fan-out (*i.e.*, number of outputs) 1 which output a fresh uniform random value in $\mathbb{K}$. A (randomized) arithmetic circuit is further formally composed of input gates of fan-in 0 and fan-out 1 and output gates of fan-in 1 and fan-out 0. During the evaluation of a ℓ-input circuit on an input $\boldsymbol{x} \in \mathbb{K}^\ell$, each wire is assigned with a value in $\mathbb{K}$. We denote `AssignWires` the probabilistic algorithm that given a randomized arithmetic circuit C, an input $\boldsymbol{x} \in \mathbb{K}^\ell$, and a subset $\mathcal{W}$ of wire labels outputs a set of $|\mathcal{W}|$ values corresponding the assignments of the wires of C with label in $\mathcal{W}$ for an evaluation on input $\boldsymbol{x}$.

1.2.2. *Additive Sharings and Gadgets*

In this chapter, the *n-additive decoding* mapping, denoted AddDec, refers to the function $\bigcup_n \mathbb{K}^n \to \mathbb{K}$ defined as

$$\mathsf{AddDec} : (x_1, \ldots, x_n) \mapsto x_1 + \cdots + x_n \ ,$$

for every $n \in \mathbb{N}$ and $(x_1, \ldots, x_n) \in \mathbb{K}^n$. We shall further consider that, for every $n, \ell \in \mathbb{N}$, on input $(\widehat{x}_1, \ldots, \widehat{x}_\ell) \in (\mathbb{K}^n)^\ell$ the n-additive decoding mapping acts as

$$\mathsf{AddDec} : (\widehat{x}_1, \ldots, \widehat{x}_\ell) \mapsto (\mathsf{AddDec}(\widehat{x}_1), \ldots, \mathsf{AddDec}(\widehat{x}_\ell)) \ .$$

For some tuple $\widehat{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n$ and for some set $I \subseteq [1; n]$, the tuple $(x_i)_{i \in I}$ is denoted $\widehat{x}|_I$.

DEFINITION 1.1.– [Additive Sharing] Let $n, \ell \in \mathbb{N}$. For any $x \in \mathbb{K}$, an *n-additive sharing* of x is a random vector $\widehat{x} \in \mathbb{K}^n$ such that $\mathsf{AddDec}(\widehat{x}) = x$. It is said to be *uniform* if for any set $I \subseteq [1; n]$ with $|I| < n$ the tuple $\widehat{x}|_I$ is uniformly distributed over $\mathbb{K}^{|I|}$. An *n-additive encoding* is a probabilistic algorithm AddEnc which on input a tuple $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{K}^\ell$ outputs a tuple $\widehat{\boldsymbol{x}} = (\widehat{x}_1, \ldots, \widehat{x}_\ell) \in (\mathbb{K}^n)^\ell$ such that $\widehat{x}_i$ is a uniform n-sharing of x_i for every $i \in [\ell]$.

We shall call an (*n-share*, *ℓ-to-m*) *gadget*, a randomized arithmetic circuit that maps an input $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$ to an output $\widehat{\boldsymbol{y}} \in (\mathbb{K}^n)^m$ such that $\boldsymbol{x} = \mathsf{AddDec}(\widehat{\boldsymbol{x}}) \in \mathbb{K}^\ell$ and $\boldsymbol{y} = \mathsf{AddDec}(\widehat{\boldsymbol{y}}) \in \mathbb{K}^m$ satisfy $\boldsymbol{y} = g(\boldsymbol{x})$ for some function g.

For an (n-share, ℓ-to-m) gadget, if we denote by $\boldsymbol{I}$ a collection of sets $\boldsymbol{I} = (I_1, \ldots, I_\ell)$ with $I_1 \subseteq [1; n]$, $\ldots$, $I_\ell \subseteq [1; n]$ where $n \in \mathbb{N}$ refers to the number of shares, for some $\widehat{\boldsymbol{x}} = (\widehat{x}_1, \ldots, \widehat{x}_\ell) \in (\mathbb{K}^n)^\ell$, we denote $\widehat{\boldsymbol{x}}|_{\boldsymbol{I}} = (\widehat{x}_1|_{I_1}, \ldots, \widehat{x}_\ell|_{I_\ell})$ where $\widehat{x}_i|_{I_i} \in \mathbb{K}^{|I_i|}$ is the tuple composed of the coordinates of the sharing $\widehat{x}_i$ of indexes included in I_i.

1.2.3. *Compilers*

DEFINITION 1.2.– [Circuit Compiler] A *circuit compiler* is a triplet of algorithms $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:

– CC (circuit compilation) is a deterministic algorithm that takes as input an arithmetic circuit C and outputs a randomized arithmetic circuit $\widehat{C}$.

– Enc (input encoding) is a probabilistic algorithm that maps an input $\boldsymbol{x} \in \mathbb{K}^\ell$ to an encoded input $\widehat{\boldsymbol{x}} \in \mathbb{K}^{\ell'}$.

– Dec (output decoding) is a deterministic algorithm that maps an encoded output $\widehat{\boldsymbol{y}} \in \mathbb{K}^{m'}$ to a plain output $\boldsymbol{y} \in \mathbb{K}^m$.

These three algorithms satisfy the following properties:

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

$$\Pr\left(\mathsf{Dec}\big(\widehat{C}(\widehat{\boldsymbol{x}})\big) = C(\boldsymbol{x}) \;\middle|\; \widehat{\boldsymbol{x}} \leftarrow \mathsf{Enc}(\boldsymbol{x})\right) = 1\ , \text{ where } \widehat{C} = \mathsf{CC}(C).$$

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

A *standard circuit compiler* with sharing order n and arithmetic base gadgets is a compiler $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ which additionally satisfies the following properties:

– the input encoding Enc is an n-additive encoding

– the output encoding Dec is the n-additive decoding mapping AddDec

– the circuit compilation CC consists in replacing each gate in the original circuit by an n-share gadget with corresponding functionality, and each wire by a set of n wires carrying an n-additive sharing of the original wire. If the input circuit is a randomized arithmetic circuit, each of its random gates is replaced by n random gates, which duly produce an n-additive sharing of a random value.

For standard circuit compilers, the correctness and efficiency directly hold from the correctness and efficiency of its gadgets.

1.3. Probing Model

The *probing model* is one of the most broadly used leakage models. Informally, the t-probing model states that during the evaluation of a circuit C, at most t wires (chosen by the adversary) leak the value they carry. The circuit C is thus claimed to be t-probing secure if the exact values of any set of t intermediate variables, that are referred to as *observations* or *probes*, do not reveal any information about its inputs. As in reality, the leakage traces somehow reveal noisy functions of the manipulated data, this model is motivated by the difficulty of learning information from the combination of t variables from their noisy functions in masking schemes (as t grows).

1.3.1. *Formal Definition*

We first recall the formal definition of the t-probing security. Intuitively, if a simulator can perfectly simulate any set of t probes without any knowledge of the secret inputs, then an attacker won't learn any sensitive information from any set of t probes.

DEFINITION 1.3.– [t-Probing Security] A randomized arithmetic circuit $\widehat{C}$ equipped with an encoding `Enc` is t-probing secure if there exists a simulator `Sim` which, for

any input $\boldsymbol{x} \in \mathbb{K}^\ell$, for every set of wires $\mathcal{W}$ such that $|\mathcal{W}| \leq t$, satisfies

$$\mathtt{Sim}(\widehat{C}, \mathcal{W}) = \mathtt{AssignWires}(\widehat{C}, \mathcal{W}, \mathtt{Enc}(\boldsymbol{x})).$$

EXAMPLE.– Let us take a toy example to illustrate this notion. Consider a randomized arithmetic circuit $\widehat{C}$ which implements a second-order multiplication (as given in Algorithm 1). Basically, from the shared input $\widehat{\boldsymbol{x}} = (\widehat{a}, \widehat{b}) = ((a_0, a_1, a_2), (b_0, b_1, b_2)) \in \mathbb{F}_2^6$, it computes a shared output $\widehat{c} = (c_0, c_1, c_2) \in \mathbb{F}_2^3$ as follows:

$$\begin{aligned}
c_0 &\leftarrow a_0 \cdot b_0 + r_{0,1} + r_{0,2} \\
c_1 &\leftarrow a_1 \cdot b_1 + (r_{0,1} + a_0 \cdot b_1 + a_1 \cdot b_0) + r_{1,2} \\
c_2 &\leftarrow a_2 \cdot b_2 + (r_{0,2} + a_0 \cdot b_2 + a_2 \cdot b_0) + (r_{1,2} + a_1 \cdot b_2 + a_2 \cdot b_1)
\end{aligned}$$

such that $c = c_0 + c_1 + c_2 = a \cdot b = (a_0 + a_1 + a_2) \cdot (b_0 + b_1 + b_2)$. From the probing security definition, $\widehat{C}$ is 2-probing secure if and only if any pair of leaking wires can be perfectly simulated without the knowledge of inputs a or b. Ignoring the copy gates to replicate variables that are to be used more than once, Algorithm 1 manipulates 27 intermediate variables, therefore the dependency of $\binom{27}{2} = 351$ pairs of potential leaking wires with the secrets a and b must be determined to conclude.

Algorithm 1 Second-order multiplication

Require: (a_0, a_1, a_2) and (b_0, b_1, b_2)
Ensure: (c_0, c_1, c_2)

1: $c_0 \leftarrow a_0 \cdot b_0$
2: $c_0 \leftarrow c_0 + r_{0,1}$
3: $c_0 \leftarrow c_0 + r_{0,2}$
4: $t \leftarrow a_0 \cdot b_1$
5: $u \leftarrow a_1 \cdot b_0$
6: $v \leftarrow r_{0,1} + t$
7: $v \leftarrow v + u$
8: $c_1 \leftarrow a_1 \cdot b_1$
9: $c_1 \leftarrow c_1 + v$
10: $c_1 \leftarrow c_1 + r_{1,2}$
11: $t \leftarrow a_0 \cdot b_2$
12: $u \leftarrow a_2 \cdot b_0$
13: $v \leftarrow r_{0,2} + t$
14: $v \leftarrow v + u$
15: $c_2 \leftarrow a_2 \cdot b_2$
16: $c_2 \leftarrow c_2 + v$
17: $t \leftarrow a_1 \cdot b_2$
18: $u \leftarrow a_2 \cdot b_1$
19: $v \leftarrow r_{1,2} + t$
20: $v \leftarrow v + u$
21: $c_2 \leftarrow c_2 + v$

1.3.2. *Proofs for Small Gadgets*

This section describes two widely deployed methods to prove the t-probing security of a circuit, namely computing the distributions and simulating the values carried by leaking wires with input shares.

1.3.2.1. *Distribution-based Proofs*

Determining whether the values carried by t leaking wires are jointly independent from a secret in $\mathbb{K}^\ell$ can be done exactly by computing the distributions. Namely, given

a set $\mathcal{W}$ of t leaking wires and a $\mathbb{K}^\ell$-valued random variable $\mathcal{X}$, we must check if the following equality is satisfied:

$$\forall\, \boldsymbol{w} \in \mathbb{K}^t,\ \forall\, \boldsymbol{x} \in \mathbb{K}^\ell,\ \Pr[\texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\mathcal{X})) = \boldsymbol{w}] \\ = \Pr[\texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\mathcal{X})) = \boldsymbol{w} | \mathcal{X} = \boldsymbol{x}].$$

This can be done by computing the realizations of $\mathcal{W}$ and $\mathcal{X}$ for all the possible inputs (including all the possible random values involved). Such a method quickly becomes very expensive. Then, it remains to extend the verification to all the possible wire sets $\mathcal{W}$ whose number is exponential in t and in the total number of wires.

EXAMPLE.– Let us take the example of the randomized arithmetic circuit implementing the second-order multiplication of Algorithm 1. Taking $\mathcal{W}$ as the wires carrying variable c_0 as defined at Step 2 (*i.e.*, $a_0 \cdot b_0 + r_{0,1}$) and variable u as defined at Step 12 (*i.e.*, $a_2 \cdot b_0$), the idea is to compute the values taken by $\mathcal{W}$ and $\mathcal{X}$ for all the possible values of the variables involved: a, b, a_0, a_2, b_0 and $r_{0,1}$. We get (*e.g.*, using a truth table) that, for all $\alpha \in \mathbb{F}_2$,

$$\Pr[\texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\mathcal{X})) = (\alpha, 0)] = 3/8$$
$$\Pr[\texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\mathcal{X})) = (\alpha, 1)] = 1/8$$

and

$$\forall\, \boldsymbol{x} \in \mathbb{F}_2^2,\ \Pr[\texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\mathcal{X})) = (\alpha, 0), \mathcal{X} = \boldsymbol{x}] = 3/32$$
$$\forall\, \boldsymbol{x} \in \mathbb{F}_2^2,\ \Pr[\texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\mathcal{X})) = (\alpha, 1), \mathcal{X} = \boldsymbol{x}] = 1/32.$$

The desired equality directly follows using the Bayes formula. The same computation is then to be performed for the 350 remaining pairs.

1.3.3. *Simulation-based Proofs*

A second method to circumvent the heavy computation of distributions relies on simulation-based properties to demonstrate the independence of the leaking wires from the input secrets. Informally, the idea is to perfectly simulate each possible set of leaking wires with the smallest set of input shares. If the latter is independent from the secret, then so is the set of leaking wires. If any set of t leaking wires can be perfectly simulated with at most t shares of each input, then the circuit is said to be t-*non-interferent* (or t-NI) and if the input sharing is uniform, it directly implies the t-probing security. The formal definition is recalled hereafter.

DEFINITION 1.4.– [t-Non-Interference] A randomized arithmetic circuit $\widehat{C}$ equipped with an encoding Enc is t-non-interferent (or t-NI) if there exists a deterministic simulator Sim_1 and a probabilistic simulator Sim_2, such that, for any input $\boldsymbol{x} \in \mathbb{K}^\ell$, for

every set of leaking wires $\mathcal{W}$ of size t,

$$(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell) \leftarrow \mathsf{Sim}_1(\widehat{C}, \mathcal{W}) \quad \text{with } |\mathcal{I}_1|, |\mathcal{I}_2|, \ldots, |\mathcal{I}_\ell| \leq t$$
$$\text{and} \quad \mathsf{Sim}_2(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell) = \texttt{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})).$$

Two different types of security proofs can be envisioned to demonstrate that a circuit is t-NI that we refer to as *exhaustive* proofs and *generic* proofs.

Exhaustive simulation-based proofs simply enumerate all the possible sets of t leaking wires in the circuit and check for each of them that the minimal number of input shares that are needed for a perfect simulation is at most equal to t.

EXAMPLE.– The set $\mathcal{W}$ of two leaking wires carrying the variables $a_0 \cdot b_0 + r_{0,1}$ and $a_2 \cdot b_0$ of our second-order multiplication circuit can be perfectly simulated from the set of input shares (a_2, b_0). Indeed, the second variable is perfectly simulated from the two input shares and the first variable can be generated uniformly as random as its distribution is independent from any input share with the addition of the random value $r_{0,1}$ which is used nowhere else in $\mathcal{W}$.

Such exhaustive simulation-based proofs are implemented in automatic verification tools that take the description of a randomized arithmetic circuit as input and produce either a security proof in the probing model or output a potential attack path. Some of these tools determine the smallest set of input shares required for the simulation from various methods that are more or less efficient and complete.

Generic NI-based proofs aim to demonstrate the t-non-interference of a circuit masked with n shares (*e.g.*, $n = t + 1$ shares), without instantiating n and t. The main idea is to prove that whatever the set of $t < n$ leaking wires, they can be simulated with at most t shares of each input.

EXAMPLE.– Consider a very simple sharewise addition gadget masked with n shares such that for any input sharings $\widehat{a} \in \mathbb{K}^n$ and $\widehat{b} \in \mathbb{K}^n$, the output sharing $\widehat{c}$ is defined as follows:

$$c_i \leftarrow a_i + b_i, \ \ \forall i \in [1; n].$$

Now let $\mathcal{W}$ be a set of probes such that $|\mathcal{W}| \leq t < n$. In our example, probes can take the form of a share of $\widehat{a}$, a share of $\widehat{b}$, or a share of $\widehat{c}$. We need to demonstrate that these probes can be perfectly simulated using at most t shares of $\widehat{a}$ and t shares of $\widehat{b}$. To do so, we define two sets I and J that are initially empty and that will represent the indices of the needed shares of $\widehat{a}$ and $\widehat{b}$ respectively. Namely, for each probe p in $\mathcal{W}$, we suggest to fill them as follows:

– we add the index i to I if p corresponds to a_i,

– we add the index i to J if p corresponds to b_i,

– we add the index i to I and the index i to J if p corresponds to c_i.

Doing so, we have covered all the possible probes and we can notice that $|I|$ and $|J|$ are both upper bounded by $|\mathcal{W}|$ (since each probe adds at most one index to each set). Now, it remains to show that the shares of $\widehat{a}$ whose indices are in I and the shares of $\widehat{b}$ whose indices are in J are enough to perfectly simulate all the probes in $\mathcal{W}$:

– for any probe p such that $p = a_i$, i is in I and so p can be perfectly simulated from a_i,

– for any probe p such that $p = b_i$, i is in J and so p can be perfectly simulated from b_i,

– for any probe p such that $p = c_i$, i is in $I \cap J$ and so p can be perfectly simulated from a_i and b_i.

We have thus demonstrated that any set $\mathcal{W}$ of at most t probes (with $t < n$) can be perfectly simulated from at most $|\mathcal{W}|$ shares of each input, hence the considered gadget is t-NI. While this example remains very simple, a very detailed example of such a proof of t non-interference for a more complex gadget is provided in Appendix C of Barthe et al.'s paper from CCS 2016.

1.3.4. *Limitations*

The probing model is quite convenient for security proofs as it manipulates the exact values carried by the leaking wires. Nevertheless, it fails in precisely reflecting the reality of embedded devices in at least two main aspects.

On the one hand, it does not natively consider physical defaults, like glitches or couplings which can yield leakage on secret values with less observations than expected from the probing security order.

EXAMPLE.– In order to illustrate the limitations of the probing model with respect to some physical effects, we take a simple example. On many micro-controllers, we observe that the power consumption could strongly depend on the number of bits that are actually modified when adding a new variable in a register. In the example of Algorithm 1, if output variables correspond to registers, then we can see that $a_0 \cdot b_2$ (line 11) and $a_1 \cdot b_2$ (line 17) are successively stored in the same register. These successive operations are likely to leak at once the Hamming Distance (i.e, the number of ones in the binary expression of the exclusive or) between $a_0 \cdot b_2$ and $a_1 \cdot b_2$. A single observation might reveal information on two shares of input a, and the corresponding gadget made of n-share variables would not be $(n-1)$-NI.

On the other hand, the probing model fails to capture the powerful horizontal attacks, *i.e.* it typically ignores the repeated manipulation of identical sensitive intermediate variables which would average the noise and hence remove uncertainty on secret variables. In practice, if we reasonably assume that each variable concretely leaks a deterministic function of its value plus an independent Gaussian noise with a constant standard deviation, then observing α occurrences of this variable will yield a reduction of the noise by a factor $\sqrt{\alpha}$. In the probing model, such repetitions have no effect on the targeted security order.

1.4. Robust Probing Model

In the t-probing model, t circuit wires are supposed to leak, independently from each other. However, in practice, physical defaults might generate a leakage of values carried by several wires at once. In that case, the lowest secret-dependent statistical moment of the leakage distribution could be lower that t. The most typical examples of physical defaults include transitions and glitches, and probes can be extended accordingly. For instance, in presence of glitches, probes (or equivalently observations) might reveal all the variables carried by coming wires up to the last synchronization point. Such probes thus include not only one, but a set of wires. Similarly, pairs of values that are successively stored in the same memory cell may leak at once (when the second variable replaces the first one), therefore in presence of transition-based leakage, probes are generally assumed to include specific pairs of wires.

Informally, a circuit is t-robust probing secure if any set of t so-called extended probes is independent from its inputs. The main idea remains similar to that of the probing model with the difficulty of combining several noisy values but the model is now refined by extending the probes that are likely to reveal more information than one single variable at once, following the practical observations on embedded devices.

1.4.1. *Formal Definition*

The formal definition of the t-robust probing model directly follows that of the probing model in which wires are replaced by extended probes. The latter may be reduced to sets of wires which simultaneously leak the values they carry. The definition of these sets completely depends on the physical defaults that are captured and on the architecture of the circuit.

DEFINITION 1.5.– [t-Robust Probing Security] A randomized arithmetic circuit $\widehat{C}$ equipped with an encoding $\mathtt{Enc}$ is t-robust probing secure if there exists a simulator $\mathtt{Sim}$ which, for any input $\boldsymbol{x} \in \mathbb{K}^{\ell}$, for every set of extended probes $\mathcal{P}$ such that $|\mathcal{P}| \leq t$ and pointing to the set of wires $\mathcal{W}$, satisfies

$$\mathtt{Sim}(\widehat{C}, \mathcal{W}) = \mathtt{AssignWires}(\widehat{C}, \mathcal{W}, \mathtt{Enc}(\boldsymbol{x})).$$

In this definition, a single extended probe in $\mathcal{P}$ potentially represents several wires in $\mathcal{W}$. That is why $|\mathcal{P}| \leq t$ but $\mathcal{W}$ can be larger in size than t.

EXAMPLE.– Let us consider a randomized arithmetic circuit implementing Algorithm 1 such that the three first steps are executed before the variable c_0 is stored in a register. In presence of glitches, an extended probe on c_0 after these three steps (*i.e.*, $\mathcal{P} = \{c_0\}$) could leak all the variables of coming wires up to the last synchronization point at once, namely a_0, b_0, $r_{0,1}$ and $r_{0,2}$ (*i.e.*, $\mathcal{W} = \{a_0, b_0, r_{0,1}, r_{0,2}\}$). Four wires could thus be targeted at the cost of a single probe.

Similarly, in case variable u at Step 5 and variable u at Step 12 are stored in the same memory cell consecutively, then a single probe at Step 12 might reveal information on the two variables $a_1 \cdot b_0$ and $a_2 \cdot b_0$ (corresponding to two different wires). In this case, two shares of a would be partly revealed at the cost of a single observation, which would reduce the robust security order to at most 1.

1.4.2. *Proofs for Small Gadgets*

As in the probing model, the security of small gadgets in the robust probing model can be demonstrated through distribution-based proofs or (exhaustive or generic) simulation-based proofs. The former behave the same as in the probing model but the leaking wire set $\mathcal{W}$ can be of size larger than t with extended probes. Exhaustive simulation-based proofs enumerate all the possible sets of leaking wires corresponding to t extended probes (*i.e.*, sets of leaking wires which are at least of size t) in the circuit and check for each of them that the minimal number of input shares that are needed for a perfect simulation is at most equal to t. Proving that a larger set only depends on t input shares may be more complex but the number of possible sets is very likely to be reduced.

EXAMPLE.– Let us get back to the example of the second-order multiplication. We re-write it on Algorithm 2 with a careful usage of registers. Namely, the notation `"[v]"` means that at this step, the output variable of the expression `v` is stored in a register. We then assume that each intermediate variable leaks all its inputs from their last storage in registers. For instance, the output c_1 (at Step 10) would leak a_1, b_1, $r_{1,2}$ and $v = ((r_{0,1} + a_0 \cdot b_1) + a_1 \cdot b_0)$, making it useless for an attacker to target these variables individually. The leaking sets are then larger but their number is much smaller: $\binom{9}{2} = 36$ compared to $\binom{27}{2} = 351$ in the probing model.

Algorithm 2 Second-order multiplication with careful storage in registers

Require: (a_0, a_1, a_2) and (b_0, b_1, b_2)
Ensure: (c_0, c_1, c_2)
1: $c_0 \leftarrow a_0 \cdot b_0$
2: $c_0 \leftarrow c_0 + r_{0,1}$
3: $c_0 \leftarrow c_0 + r_{0,2}$
4: $t \leftarrow a_0 \cdot b_1$
5: $u \leftarrow a_1 \cdot b_0$
6: $v \leftarrow [r_{0,1} + t]$
7: $v \leftarrow [v + u]$
8: $c_1 \leftarrow a_1 \cdot b_1$
9: $c_1 \leftarrow c_1 + v$
10: $c_1 \leftarrow c_1 + r_{1,2}$
11: $t \leftarrow a_0 \cdot b_2$
12: $u \leftarrow a_2 \cdot b_0$
13: $v \leftarrow [r_{0,2} + t]$
14: $v \leftarrow [v + u]$
15: $c_2 \leftarrow a_2 \cdot b_2$
16: $c_2 \leftarrow c_2 + v$
17: $t \leftarrow a_1 \cdot b_2$
18: $u \leftarrow a_2 \cdot b_1$
19: $v \leftarrow [r_{1,2} + t]$
20: $v \leftarrow [v + u]$
21: $c_2 \leftarrow c_2 + v$

Eventually, generic simulation-based proofs are also a bit different. Unlike in the probing model, where t input shares can be used to simulate t leaking wires, we now need t input shares to potentially simulate significantly more.

EXAMPLE.– Getting back to our example of the n-share sharewise addition gadget, each probe p on an output share c_i (for $1 \leq i \leq n$) would leak information on $c_i = a_i + b_i$, a_i and b_i (instead of only c_i). However, the gadget remains t-NI (for $t < n$) in the robust probing model since each (extended) probe can still be simulated using at most one share of each input. In our example, when $p = c_i$ (for some $1 \leq i \leq n$), all the corresponding leaking wires $c_i = a_i + b_i$, a_i and b_i can be simulated from one share of $\widehat{a}$ (*i.e.*, a_i) and one share of $\widehat{b}$ (*i.e.*, b_i).

1.4.3. *Limitations*

While the robust probing model nicely supplements the probing model by handling physical defaults, it suffers from the same limitations regarding horizontal attacks. Indeed, the robust probing model is similarly based on a fixed number of probes regardless of their number of occurrences (*i.e.*, usages).

1.5. Random Probing Model & Noisy Leakage Model

The *noisy leakage model* can be seen as a specialization of the *only computation leaks* model. Informally, a circuit is secure in the noisy leakage model if the adversary cannot recover non-negligible information on the secrets from a noisy function of each intermediate variable of the implementation. Such a noisy function f is said to be δ-noisy if it satisfies $\beta(\mathcal{X}, f(\mathcal{X})) \stackrel{\text{def}}{=} \Delta(\mathcal{X}; (\mathcal{X}|f(\mathcal{X}))) \leq \delta$ where Δ denotes the statistical distance between the laws of $\mathcal{X}$ and $(\mathcal{X}|f(\mathcal{X}))$ over the random distribution of $f(\mathcal{X})$ and for a uniform random variable $\mathcal{X}$. We stress that any power or electromagnetic leakage can be captured by this model.

1.5.1. *Formal Definition of the Noisy Leakage Model*

In the noisy leakage model, each wire is assumed to leak a noisy function of the value it carries. Let δ be the corresponding noise parameter. The formal definition follows. Informally, a circuit is secure against noisy leakage if the noisy leakage of each of its wires is not enough to recover non-negligible information on the input secrets.

DEFINITION 1.6.– [Security against δ-Noisy Leakage] Let $\mathcal{X}$ be a uniform random variable over $\mathbb{K}^{\ell}$. A randomized arithmetic circuit C with $\ell \cdot n \in \mathbb{N}$ input gates and made of a set $\mathcal{W}$ of wires is ε-secure against δ-*noisy leakage* with respect to encoding Enc if:

$$\beta(\mathcal{X}|f_1(W_1), \ldots, f_{|\mathcal{W}|}(W_{|\mathcal{W}|})) \leq \varepsilon$$

where

$$W_1, \ldots, W_{|\mathcal{W}|} \leftarrow \mathsf{AssignWires}(C, \mathcal{W}, \mathsf{Enc}(\mathcal{X}))$$

for any δ-noisy functions $f_1, \ldots, f_{|\mathcal{W}|}$.

Note that the statistical distance can be based on the L_2 (Euclideans) norm, on the L_1 norm, or even on the relative error.

1.5.2. *Limitations*

The security proofs remain far from convenient and masking schemes continue to be verified in the probing model.

1.5.3. *Reduction to the Probing Model*

The noisy leakage security can be reduced to the probing security. This reduction relies on an intermediate leakage model, called *random probing model*. Informally in the random probing model, each intermediate variable leaks with some constant leakage probability p. A circuit is secure if there is a negligible probability that these leaking wires actually reveal information about the secrets. The random probing model further encompasses the powerful horizontal attacks which exploit the repeated manipulations of variables in an implementation and also benefits from a tight reduction with the noisy leakage model which becomes independent of the size of the circuit.

The reduction of the noisy leakage security to the probing security first relies on the fact that, by applying the Chernoff bound, a t-probing secure computation is also p-random probing secure with $p = t/s$ where s denotes the number of gates in the

original circuit. The reduction is not tight in this respect (considering a constant number of probes) since the security level decreases as the size of the circuit increases. The second transition and key lemma of the reduction proves that p-random probing security implies δ-noisy leakage security with $\delta = p/|\mathbb{K}|$ where $|\mathbb{K}|$ denotes the cardinal of the base field $\mathbb{K}$. This is because any δ-noisy function f can be written as $f(\cdot) = g \circ \phi(\cdot)$ where g is a randomized function and ϕ is a p-random probing function (*i.e.* $\phi(x) = x$ with probability p and $\phi(x) = \perp$ otherwise) with $p = \delta \cdot |\mathbb{K}|$.

REMARQUE. Note that the definition of the noisy leakage model can be modified using other metrics than the statistical distance in order to tighten the reduction. For instance, using the average relative error (ARE) from pointwise mutual information would eliminate the multiplicative factor $|\mathbb{K}|$. This is because the statistical distance is a average case metric while the random probing is a worst case measurement of the leakage.

1.5.4. *Formal Definition of the Random Probing Model*

Let $p \in [0, 1]$ be some constant leakage probability parameter. This parameter is sometimes called *leakage rate* in the literature. Informally, the p-random probing model states that, during the evaluation of a circuit C, each wire leaks its value with probability p (and leaks nothing otherwise), where all the wire leakage events are mutually independent.

In order to formally define the random-probing leakage of a circuit, we shall define an additional *leaking-wires sampler*. This probabilistic algorithm takes as input a randomized arithmetic circuit C and a probability $p \in [0, 1]$, and outputs a set $\mathcal{W}$, denoted as

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

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

DEFINITION 1.7.– [(p, ϵ)-Random Probing Security] A randomized arithmetic circuit C with $\ell \cdot n \in \mathbb{N}$ input gates is (p, ϵ)-*random probing secure* with respect to encoding Enc if there exists a simulator Sim such that for every $\boldsymbol{x} \in \mathbb{K}^{\ell}$:

$$\mathsf{Sim}(C) \approx_{\varepsilon} \mathsf{AssignWires}(C, \mathsf{LeakingWires}(C, p), \mathsf{Enc}(\boldsymbol{x})) \ .$$

We can further consider a *simulation with abort*. In this approach, the simulator first calls the leaking-wires sampler to get a set $\mathcal{W}$ and then either aborts (or fails) with

probability ε or outputs the exact distribution of the wire assignment corresponding to $\mathcal{W}$. Formally, for any leakage probability $p \in [0, 1]$, the simulator Sim is defined as

$$\mathsf{Sim}(C) = \mathsf{SimAW}(C, \mathsf{LeakingWires}(C, p))$$

where SimAW, the *wire assignment simulator*, either returns $\perp$ (simulation failure, with probability ε) or a perfect simulation of the requested wires. It is not hard to see that if we can construct such a simulator SimAW for a compiled circuit $\widehat{C}$, then this circuit is (p, ε)-random probing secure.

1.5.5. *Proofs in the Random Probing Model*

In this section, we show how to compute the simulation failure probability ε as a function $f(p)$ of the leakage probability p.

We consider a compiled circuit $\widehat{C}$ composed of s wires labeled from 1 to s. The simulation failure probability ε can then be explicitly expressed as a function of p. Namely, we have $\varepsilon = f(p)$ with f defined for every $p \in [0, 1]$ by:

$$f(p) = \sum_{i=1}^{s} c_i \cdot p^i \cdot (1-p)^{s-i}$$

with c_i the number of subsets $\mathcal{W} \subseteq [s]$ of cardinality i for which the simulation fails. For any circuit $\widehat{C}$ achieving t-probing security, the values c_i with $i \leq t$ are equal to zero, which implies the following simplification:

$$f(p) = \sum_{i=t+1}^{s} c_i \cdot p^i \cdot (1-p)^{s-i}. \qquad [1.1]$$

EXAMPLE.– [Evaluating $f(p)$ for the second-order multiplication gadget] Getting back to our second-order multiplication gadget from Algorithm 1, we can compute the failure function (*e.g.* using automatic tools) and display its first coefficients:

$$f : p \mapsto 1\,297 \cdot p^3 + \mathcal{O}(p^4).$$

Notice that the three first coefficients are zero, which confirms the second-order probing security of this gadget. Then, 1 297 triplets of intermediate variables are leaking information on the secret inputs (on 2 925 triplets in total). Given the level of noise of a chosen underlying platform, the probability p can be defined to evaluate the random probing security parameter $\varepsilon = f(p)$ of this gadget.

1.5.6. *Extension to Handle Physical Defaults*

In a first attempt, the random probing model assumes that all the wires in a circuit are leaking independently with the same probability p. In practice, in presence of physical defaults, this assumption may be too strong. First, different wires may leak with different probabilities (*i.e.*, different signal-to-noise ratios). Second, dependencies might occur between the leakage of different wires.

In the first scenario, more accurate probabilities can be computed from the signal-to-noise ratios. From independent probabilities for each wire, it is easy (but much less efficient) to compute the failure function f. Unlike the formula from Equation 1.1, sets of wires cannot be jointly evaluated based on their size as they would not share the same probability to leak anymore. The probability that a tuple $(x_1, x_2, \ldots, x_\alpha)$ of *at least* α wires jointly leaks is then :

$$p_{x_1} \cdot p_{x_2} \cdot \cdots \cdot p_{x_\alpha}.$$

The probability that a tuple $(x_1, x_2, \ldots, x_\alpha)$ of *only* α wires among $\beta > \alpha$ wires leaks is then :

$$p_{x_1} \cdot p_{x_2} \cdot \cdots \cdot p_{x_\alpha} \cdot (1 - p_{x_{\alpha+1}}) \cdot (1 - p_{x_{\alpha+2}}) \cdot \cdots \cdot (1 - p_{x_\beta}).$$

If all the probabilities are different, then all the tuples must be individually considered in the computation of the failure function.

The presence of physical defaults like glitches breaks the independence between the leaking probabilities of different variables. In particular, wires that carry variables in the same computation between two synchronization points are likely to leak simultaneously. The computation of the failure function then depends on the dependencies between the leaking wires.

1.6. Composition

As explained in the previous sections, exhaustive or generic proofs can be built to demonstrate the security of small gadgets implementing atomic operations (*e.g.*, additions, multiplications). One step further, building security proofs for complex circuits was shown to be either error-prone for hand-made proofs or computationally impossible with automatic verification tools. Hence the need to split big circuits into smaller blocks that can be proven secure and then safely combine them with well-defined composition rules (see Figure 1.1 for an illustration).

In the following, we will recall the main composition rules for the aforementioned models and explain how to use them to safely compose small secure gadgets.

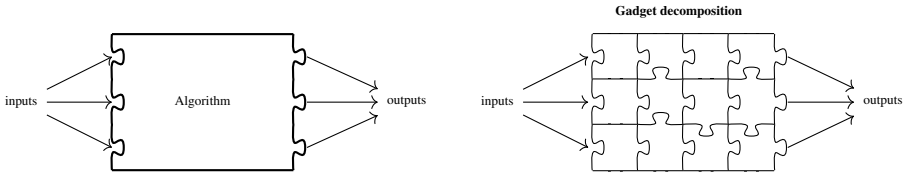


Figure 1.1. *Illustration of (de)-composition*

1.6.1. *Composition in the Probing Model*

Two main composition techniques to achieve global probing security are described hereafter.

1.6.1.1. *Doubling the Number of Shares*

A simple method to build secure masked schemes for Boolean circuits is based on the use of two specific gadgets and the doubling of the number of shares. Concretely, any Boolean circuit can be decomposed into `and` and `not` gates only:

– each `and` gate can be replaced by the $(n = 2t + 1)$-share ISW multiplication gadget,

– each `not` gate can be replaced by a $(n = 2t + 1)$-share gadget applying a `not` gate to one single share only.

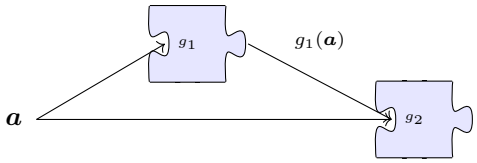
Then, it can be easily shown that any wire among these gadgets in a Boolean circuit can be perfectly simulated from at most the same two shares i and j at input/output of all the circuit's gadgets. Therefore, any set of t probes can trivially be simulated from at most $2 \cdot t$ shares of all the manipulated data in the circuit and the whole Boolean circuit is thus t-probing secure.

Although this method can be used to safely mask any Boolean circuit in the probing model, the doubling of the number of shares makes it quite expensive.

1.6.1.2. *Non-Interference-Based Proofs*

Relying on simulation-based properties, it is possible to achieve a tighter composition with gadgets with the optimal number of $n = t + 1$ shares, at the cost of stronger security properties. Indeed, directly composing any gadget of $n = t + 1$ shares does not always yield t-probing security, due to the dependency between some gadgets' inputs.

EXAMPLE.– See for instance the masking scheme that takes a $(n = t + 1)$-sharing $\boldsymbol{a}$ and that computes $g_2(\boldsymbol{a}, g_1(\boldsymbol{a}))$ using two t-NI gadgets g_1 and g_2 (see Definition 1.4), illustrated below.



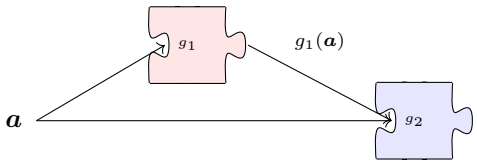
A t-probing adversary can observe t_0 input shares of $\boldsymbol{a}$, t_1 intermediate variables on g_1 and t_2 intermediate variables on g_2 as soon as $t_0 + t_1 + t_2 \leq t$. From the t-NI property of g_2, all its t_2 probes can be perfectly simulated from t_2 shares of $\boldsymbol{a}$ and t_2 output shares of g_1. The t_1 probes of g_1 together with the t_2 output shares required to simulate g_2's probes can themselves be perfectly simulated from $t_1 + t_2$ input shares of g_1, *i.e.* shares of $\boldsymbol{a}$. At the end, all the probes can be simulated from $t_0 + t_1 + 2 \cdot t_2$ shares of $\boldsymbol{a}$ which can be higher than t. The proof therefore cannot be completed, hence the need for stronger properties [1].

An example of a stronger property is the *strong non-interference*. It benefits from stopping the propagation of the probes between the output and the input shares and additionally trivially implies t-NI. Intuitively, a circuit is t-strong non-interferent (or t-SNI) if and only if any set of at most t intermediate variables whose t_{int} on internal variables and t_{out} on the output shares can be perfectly simulated with at most t_{int} shares of each input.

DEFINITION 1.8.– [t-Strong Non-Interference] A randomized arithmetic circuit $\widehat{C}$ equipped with an encoding Enc is t-strong non-interferent (or t-SNI) if there exists a deterministic simulator Sim_1 and a probabilistic simulator Sim_2, such that, for any input $\boldsymbol{x} \in \mathbb{K}^\ell$, for every sets of internal leaking wires $\mathcal{W}_i$ of size t_i and output leaking wires $\mathcal{W}_o$ of size t_o such that $t_i + t_o \leq t$,

$$(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell) \leftarrow \mathsf{Sim}_1(\widehat{C}, \mathcal{W}_i, \mathcal{W}_o) \quad \text{with } |\mathcal{I}_1|, |\mathcal{I}_2|, \ldots, |\mathcal{I}_\ell| \leq t_i$$
$$\text{and} \quad \mathsf{Sim}_2(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell) = \mathtt{AssignWires}(\widehat{C}, \mathcal{W}_i \cup \mathcal{W}_o, \mathsf{Enc}(\boldsymbol{x})).$$

EXAMPLE.– Getting back to our example, let us assume now that g_1 is t-SNI. A t-probing adversary can still observe t_0 input shares of $\boldsymbol{a}$, t_1 intermediate variables on g_1 and t_2 intermediate variables on g_2 (which is t-NI) as soon as $t_0 + t_1 + t_2 \leq t$.



1. In practice, such a structure indeed yields concrete probing attacks for specific choices of gadgets g_1 and g_2.

From the t-NI property of g_2, all its t_2 probes can be perfectly simulated from t_2 shares of $\boldsymbol{a}$ and t_2 output shares of g_1. Then, the t_1 probes of g_1 together with the t_2 output shares required to simulate g_2's probes can themselves be perfectly simulated from only t_1 input shares of g_1 thanks to the t-SNI property (and because $t_1 + t_2 \leq t$). At the end, all the probes of the circuit can be simulated from $t_0 + t_1 + t_2$ shares of $\boldsymbol{a}$ which is always lower or equal to t, which is enough to conclude that the circuit is t-NI, hence t-probing secure.

REMARQUE. Many other properties based on the non-interference notions can be used to build tighter compositions with lower complexity and/or better security levels. In particular, the Probe Isolating Non-Interference property (or PINI) reasons on the dependency between a probe and the index of the share of input it can be simulated from, rather than the number of probes and input shares.

In case a circuit misses gadgets with strong properties to achieve composition, a common practice is to add so-called *refresh gadgets*. The latter are functionally equivalent to the identity function but refresh the sharing with new random values to break the dependencies between the old input sharing and the new one. Note that inserting t-SNI refresh gadgets at careful locations is enough to make a randomized arithmetic circuit made of t-NI secure gadgets globally t-probing secure.

1.6.1.3. *Extension in the Glitch Robust Probing Model*

Intuitively, the simulation strategy used in the probing model to securely compose probing gadgets can directly be applied to the glitch robust probing model when there are no glitches on the inputs. And it is actually proven to be also true in the latter scenario. Namely, we can build a global simulator from glitch-robust probing simulators on each individual gadget that take and produce extended probes.

1.6.2. *Composition in the Random Probing Model*

Probing-secure schemes are also secure in the random probing model, but the tolerated leakage probability might not be constant, which is not satisfactory from a practical viewpoint. Indeed, in practice, the side-channel noise may not be customizable by the implementer.

In this section, we describe two methods to safely compose gadgets and build schemes that are secure in the random probing model.

1.6.2.1. *Simulation Based on the Number of Shares*

The first method relies on the *random probing composability* notion for a gadget. Informally and similarly to the non-interference property in the probing model, this

notion relies on the capability for a gadget to tolerate a certain number of probes on its output and a certain number of probes on its intermediate variables. All these probes must additionally be (almost perfectly) simulated by a fixed set of input shares to ensure the composition with prior gadgets.

DEFINITION 1.9.– [Random Probing Composability] Let $n, \ell, m \in \mathbb{N}$. An ($n$-share, ℓ-to-m) gadget $G : (\mathbb{K}^n)^\ell \to (\mathbb{K}^n)^m$ is (t, p, ε)-*random probing composable (RPC)* for some $t \in \mathbb{N}$ and $p, \varepsilon \in [0, 1]$ if there exists a deterministic algorithm Sim_1^G and a probabilistic algorithm Sim_2^G such that for every input $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$ and for every set collection $J_1 \subseteq [1; n]$, ..., $J_m \subseteq [1; n]$ of cardinals $|J_1| \leq t$, ..., $|J_m| \leq t$, the random experiment

$$\mathcal{W} \leftarrow \mathsf{LeakingWires}(G, p)$$
$$\boldsymbol{I} \leftarrow \mathsf{Sim}_1^G(\mathcal{W}, \boldsymbol{J})$$
$$out \leftarrow \mathsf{Sim}_2^G\left(\widehat{\boldsymbol{x}}|_{\boldsymbol{I}}\right)$$

yields

$$\Pr\left((|I_1| > t) \vee \ldots \vee (|I_\ell| > t)\right) \leq \varepsilon \qquad [1.2]$$

and

$$out \overset{\text{id}}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}, \widehat{\boldsymbol{x}})\ ,\ \widehat{\boldsymbol{y}}|_{\boldsymbol{J}}\right)$$

where $\boldsymbol{J} = (J_1, \ldots, J_m)$ and $\widehat{\boldsymbol{y}} = G(\widehat{\boldsymbol{x}})$. Let $f : \mathbb{R} \to \mathbb{R}$. The gadget G is (t, f)-RPC if it is $(t, p, f(p))$-RPC for every $p \in [0, 1]$.

In the above definition, the first-pass simulator Sim_1^G determines the necessary input shares (through the returned collection of sets $\boldsymbol{I}$) for the second-pass simulator Sim_2^G to produce a perfect simulation of the leaking wires defined by the set $\mathcal{W}$ together with the output shares defined by the collection of sets $\boldsymbol{J}$. Note that there always exists such a collection of sets $\boldsymbol{I}$ since $\boldsymbol{I} = ([1; n], \ldots, [1; n])$ trivially allows a perfect simulation whatever $\mathcal{W}$ and $\boldsymbol{J}$. However, the goal of Sim_1^G is to return a collection of sets $\boldsymbol{I}$ with cardinals at most t. The idea behind this constraint is to keep the following composition invariant: for each gadget we can achieve a perfect simulation of the leaking wires plus t shares of each output sharing from t shares of each input sharing. We shall call *failure event* the event that at least one of the sets I_1, ..., I_ℓ output of Sim_1^G has cardinality greater than t. When (t, p, ε)-RPC is achieved, the failure event probability is upper bounded by ε according to [1.2]. A failure event occurs whenever Sim_2^G requires more than t shares of one input sharing to be able to produce a perfect simulation of the leaking wires (*i.e.*the wires with label in $\mathcal{W}$) together with the output shares in $\widehat{\boldsymbol{y}}|_{\boldsymbol{J}}$. Whenever such a failure occurs, the composition invariant is broken. In the absence of failure event, the RPC notion implies that a perfect simulation can be

achieved for the full circuit composed of RPC gadgets. This is formally stated in the next theorem.

THEOREM.– [Composition] Let $t \in \mathbb{N}$, $p, \varepsilon \in [0,1]$, and CC be a standard circuit compiler with (t, p, ε)-RPC base gadgets. For every (randomized) arithmetic circuit C composed of $|C|$ gadgets, the compiled circuit $\mathsf{CC}(C)$ is $(p, |C| \cdot \varepsilon)$-random probing secure.

1.6.2.2. *Simulation Based on the Set of Shares*

The first method relies on the number of leaking wires at the input and output of each composable gadget. By preserving the same number of leaking wires at the input and output, the security of each gadget can be easily considered separately and the composition can be evaluated efficiently. Nevertheless, the security of the composition can be tighter by further considering specific sets of leaking inputs and outputs rather than their number.

This second method therefore relies on so-called (two-entry) *probe distribution tables* (PDT). Informally, for some gadget $\mathcal{G}$, its PDT provides, for a set of input shares I and a set of output shares O, the probability that a simulator exactly needs I to simulate the leaking wires in $\mathcal{G}$ (*i.e.*, from $\mathsf{LeakingWires}(G, p)$) and the output shares O. Contrary to the first method which records the number of required input and output shares, the PDT records the exact input and output sets of shares.

DEFINITION 1.10.– [Probe Distribution Table] Let $p \in [0,1]$ be some constant leakage probability parameter. Let G be an (n-share, ℓ-to-m) gadget with a set of input wires $\boldsymbol{I} = (I_1, \ldots, I_\ell)$ and a set of output wires $\boldsymbol{O} = (O_1, \ldots, O_m)$. The PDT of G is a $[0,1]^{2^{\ell \cdot n} \times 2^{m \cdot n}}$ matrix such that $\mathtt{PDT}_G[\boldsymbol{I'}, \boldsymbol{O'}]$ is the probability for $\boldsymbol{I'}$ to be the smaller set that is enough to simulate the leaking wires from $\mathsf{LeakingWires}(G, p)$ together with $\boldsymbol{O'}$.

EXAMPLE.– Let us take the very simple example of a (1-share, 2-to-1) gadget as an `xor` gate, as represented in Figure 1.2.

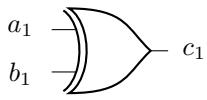


Figure 1.2. `xor` *gate with a single share for each input*

If we assume that each internal wire leaks with probability p, then its PDT table is given by:

	$O' = \emptyset$	$O' = \{c_1\}$
$\boldsymbol{I'} = \emptyset$	$1 - p^2$	0
$\boldsymbol{I'} = \{\{a_1\}\}$	$p \cdot (1-p)$	0
$\boldsymbol{I'} = \{\{b_1\}\}$	$p \cdot (1-p)$	0
$\boldsymbol{I'} = \{\{a_1\}, \{b_1\}\}$	p^2	1

For instance, the probability for $\boldsymbol{I'} = \{\{a_1\}\}$ to be the smallest set that is enough to simulate the leaking wires from $\mathsf{LeakingWires}(G, p)$ is computed from the probability that exactly a_1 is leaking (and not b_1, otherwise we would need b_1 as well), which is $p \cdot (1-p)$. Furthermore, if O' is not empty, *i.e.* $O' = \{c_1\}$, then we need both input shares to simulate it, *i.e.* $\boldsymbol{I'} = \{\{a_1\}, \{b_1\}\}$, hence the second column.

Note that $\mathtt{PDT}_G[\boldsymbol{I'}, \boldsymbol{O'}] = \sum_{i=1}^{s} c_i \cdot p^i \cdot (1-p)^{s-i}$ is similar to the failure function from [1.1] but where c_i denotes the number of leaking sets of size i that exactly requires the input shares $\boldsymbol{I'}$, to be simulated together with $\boldsymbol{O'}$.

One step further, there is a direct relation between the probe distribution table and the random probing security: a gadget G with a single input (whose sharing is indexed by I) and a PDT $\mathtt{PDT}_G$ is $(p, \mathtt{PDT}_G[I, \emptyset])$-random probing secure. The relation can be extended to multiple-input gadgets by considering the sum $\sum_{I'} \mathtt{PDT}_G[I', \emptyset]$ for all I' that cover at least one input. A partial order can be defined on PDT so that for two gadgets G_1 and G_2,

$$\mathtt{PDT}_{G_1} \leq \mathtt{PDT}_{G_2}$$

means that the amount of information leaked in G_1 is less than or equal to the information leaked in G_2. Using this partial order, composition of gadgets can be built directly from their PDTs. We recall two main composition theorems to handle the two scenarios of parallel composition and sequential composition (see Figure 1.3).

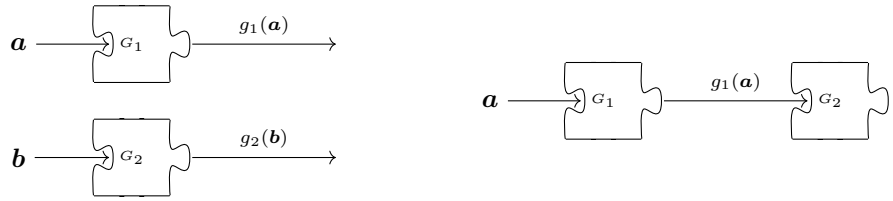


Figure 1.3. *Parallel (left) and sequential (right) composition of two gadgets*

THEOREM.– [Parallel Composition] Let G_1 and G_2 be two gadgets and let $\mathtt{PDT}_{G_1}$ and $\mathtt{PDT}_{G_2}$ be their respective probe distribution tables. The probe distribution table $\mathtt{PDT}_G$

of gadget $G = G_1 \| G_2$ can be computed as follows:

$$\mathtt{PDT}_G = \mathtt{PDT}_{G_1} \otimes \mathtt{PDT}_{G_2}$$

where $\otimes$ denotes the Kronecker product between matrices.

THEOREM.– [Sequential Composition] Let G_1 be an (ℓ_1-to-m) gadget and G_2 be an (m-to-ℓ_2) gadget. Let $\mathtt{PDT}_{G_1}$ and $\mathtt{PDT}_{G_2}$ be their respective probe distribution tables. The probe distribution table $\mathtt{PDT}_G$ of gadget $G = G_2 \circ G_1$ is bounded as follows:

$$\mathtt{PDT}_G \leq \mathtt{PDT}_{G_1} \cdot \mathtt{PDT}_{G_2}$$

where $\cdot$ denotes the standard matrix multiplication.

The global composition theorem is stated below.

THEOREM.– [Composition] Let $p, \varepsilon \in [0, 1]$. Let C be a (n-share, ℓ-input) randomized arithmetic circuit of input sharings indexed by $\boldsymbol{I}$ made of gadgets $G_1, \ldots, G_m$. Then, the PDT of C is upper bounded by a composition f of the PDTs of $G_1, \ldots, G_m$ which directly depends on the circuit's structure:

$$\mathtt{PDT}_C \leq f(\mathtt{PDT}_{G_1}, \ldots, \mathtt{PDT}_{G_m}).$$

C is then $(p, \sum_{\boldsymbol{I'}} \mathtt{PDT}_C[\boldsymbol{I'}, \emptyset])$-random probing secure for all the sets $\boldsymbol{I'}$ that cover at least one input of C. While this method yields tighter security parameters, it is also more expensive to evaluate. Hybrid methods can be exhibited with different trade-offs in terms of complexity and security evaluation.

1.7. Conclusion

This chapter has introduced several leakage models that are commonly used to model the attacker's observations on masking schemes during a side-channel attack. Different methods have been discussed depending on these leakage models to prove the security of the underlying masking schemes. Some of them apply directly to individual gadgets and can then be extended with composition techniques to prove the security of larger circuits.

The next chapter will present verification techniques to automatically prove the security of individual gadgets in the different leakage models, following the security notions introduced in this chapter.