Towards Tight Random Probing Security - extended version

Gaëtan Cassiers[1], Sebastian Faust[2],
Maximilian Orlt[2], François-Xavier Standaert[1]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
[2] TU Darmstadt, Darmstadt, Germany

Abstract. Proving the security of masked implementations in theoretical models that are relevant to practice and match the best known attacks of the side-channel literature is a notoriously hard problem. The random probing model is a promising candidate to contribute to this challenge, due to its ability to capture the continuous nature of physical leakage (contrary to the threshold probing model), while also being convenient to manipulate in proofs and to automate with verification tools. Yet, despite recent progress in the design of masked circuits with good asymptotic security guarantees in this model, existing results still fall short when it comes to analyze the security of concretely useful circuits under realistic noise levels and with low number of shares. In this paper, we contribute to this issue by introducing a new composability notion, the *Probe Distribution Table (PDT)*, and a new tool (called STRAPS, for the Sampled Testing of the RAndom Probing Security). Their combination allows us to significantly improve the tightness of existing analyses in the most practical (low noise, low number of shares) region of the design space. We illustrate these improvements by quantifying the random probing security of an AES S-box circuit, masked with the popular multiplication gadget of Ishai, Sahai and Wagner from Crypto 2003, with up to six shares.

1 Introduction

Context. Modern cryptography primarily analyzes the security of algorithms or protocols in a black-box model where the adversary has only access to their inputs and outputs. Since the late nineties, it is known that real-world implementations suffer from so-called side-channel leakage, which gives adversaries some information about intermediate computation states that are supposedly hidden. In this work, we focus on an important class of side-channel attacks against embedded devices, which exploits physical leakage such as their power consumption [26] or electro-magnetic radiation [22]. We are in particular concerned with the masking countermeasure [14], which is one of the most investigated solutions to mitigate side-channel attacks. In this context, the main scientific challenge we tackle is to find out security arguments that are at the same time practically relevant and theoretically sound.

Two separated worlds. In view of the difficulty to model side-channel attacks, their practical and theoretical investigations have first followed quite independent paths. On the practical side, the analysis of masked implementations as currently performed by evaluation laboratories is mostly based on statistical testing. Approaches for this purpose range from detection-based testing, which aims at identifying leakage independently of whether it can be exploited [32], to attack-based testing under various adversarial assumptions, which aims at approximating (if possible bounding) the concrete security level of the implementation with actual (profiled or non-profiled) attacks such as [15,11] and their numerous follow ups. On the theoretical side, the first model introduced to capture the security of masked implementations is the t-threshold probing model introduced by Ishai, Sahai and Wagner (ISW) [24]. In this model, leaky computation is captured as the evaluation of an arithmetic circuit, and the adversary may choose t wires of the circuit for which she receives the value they carry. The adversary succeeds if she recovers a secret input variable of the circuit.

The pros and cons of both approaches are easy to spot. On the one hand, statistical testing provides quantitative evaluations against concrete adversaries, but the guarantees it offers are inherently heuristic and limited to the specific setting used for the evaluations. On the other hand, theoretical models enable more general conclusions while also having a good potential for automation [5], but they may imperfectly abstract physical leakage. For some imperfections, tweaking the model appeared to be feasible. For example, ISW's threshold probing model initially failed to capture physical defaults such as glitches that can make masking ineffective [27,28]. Such glitches were then integrated in the model [21] and automated [10,4,6]. Yet, it remained that the threshold probing model is inherently unable to capture the continuous nature of physical leakage, and therefore the guarantees it provides can only be qualitative, as reflected by the notion of probing security order (i.e., the number of shares that the adversary can observe without learning any sensitive information). This also implies that so-called horizontal attacks taking advantage of multiple leakage points to reduce the noise of the implementations cannot be captured by this model [7].

An untight unifying approach. As a result of this limitation, the noisy leakage model was introduced by Prouff and Rivain [30]. In this model, each wire in the circuit leaks independently a noisy (i.e., partially randomized) value to the adversary. In an important piece of work, Duc et al. then proved that security in the threshold probing model implies security in the noisy leakage model, for some values of the model parameters [17]. This result created new bridges between the practical and theoretical analyzes of masked implementations. In particular, it made explicit that the security of this countermeasure depends both on a security order (which, under an independence assumption, depends on the number of shares) and on the noise level of the shares' leakage. So conceptually, it implies that it is sound to first evaluate the probing security order of an implementation, next to verify that this security order is maintained in concrete leakages (e.g., using detection-based statistical testing) and finally to assess the noise level. Yet, and as discussed in [18], such an analysis is still not tight: choosing security parameters based on this combination of models and the reductions connecting them would lead to overly expensive implementations compared to a choice based on the best known (profiled) side-channel attacks.

A tighter middle-ground. Incidentally, the reduction of Duc et al. also considered an intermediate level of abstraction denoted as the random probing model. In this model, each wire in the circuit independently leaks its value with probability p (and leaks no information with probability $1-p$). Technically, it turns out that the aforementioned tightness issue is mostly due to the reduction from the threshold probing model to the random probing model, while there is a closer relationship between the random probing model and the noisy leakage model [19,29]. Since the random probing model remains relatively easy to manipulate (and automate) in circuit-level proofs, it therefore appears as an interesting candidate to analyze masking schemes with tight security guarantees.

Like the noisy leakage model, the random probing model captures the concept of "noise rate", which specifies how the noise level of an implementation must evolve with the number of shares in order to remain secure against horizontal attacks. As a result, different papers focused on the design and analysis of gadgets with good (ideally constant) noise rate [1,3,2,23,20]. While these papers provide important steps in the direction of asymptotically efficient masking schemes, the actual number of shares they need to guarantee a given security level and/or the noise level they require to be secure remain far from practical. To the best of our knowledge, the most concrete contribution in this direction is the one of Belaïd et al. [8,9], which introduced a compiler that can generate random probing secure circuits from small gadgets satisfying a notion of "random probing expandability", together with a tool (called VRAPS) that quantifies the random probing security of a circuit from its leakage probability. With this tool, they reduce the level of noise required for security to practically acceptable values, but the number of shares required in order to reach a given security level for their (specialized) constructions is still significantly higher than expected from practical security evaluations – we give an example below.

Our contributions. In this paper, we improve the tightness of masking security proofs in the most practical (low noise, low number of shares) region of the design space, focusing on practical ISW-like multiplication gadgets, integrated in an AES S-box design for illustration purposes. More precisely:

We first introduce STRAPS, a tool for the Sampled Testing of the RAndom Probing Security of small circuits, which uses the Monte-Carlo technique for probability bounding and is released under an open source license.[1]

Since this tool is limited to the analysis of small circuits and/or small security orders due to computational reasons, we next combine it with a new compositional strategy that exploits a new security property for masked gadgets, the Probe Distribution Table (PDT), which gives tighter security bounds for composed circuits and is integrated in the STRAPS tool. This combination of tool and compositional strategy allows us analyzing significantly larger circuits and

[1] https://github.com/cassiersg/STRAPS

security orders than an exhaustive approach, while also being able to analyze any circuit (i.e., it does not rely on an expansion strategy [2]).

We finally confirm the practical relevance of our findings by applying them to a masked AES S-box using ISW gadgets. We show how to use them in order to discuss the trade-off between the security order and the noise level (i.e., leakage probability) of concrete masked implementations on formal bases. As an illustration, we use our tools to compare the impact of different refreshing strategies for the AES S-box (e.g., no refresh, simple refreshes or SNI refreshes) in function of the noise level. We can also claim provable security levels for useful circuits that are close to the worst-case attacks discussed in [18] which is in contrast to previous works. Precisely, we are able to prove the same statistical security order (i.e., the highest statistical moment of the leakage distribution that is independent of any sensitive information) as in this reference, for realistic leakage probabilities in the range $[10^{-1}; 10^{-4}]$. For example, our AES S-box with 6 shares and leakage probability of $\approx 10^{-3}$ ensures security against an adversary with up to one billion measurements. Belaïd et al. would need 27 shares to reach the same security.

Open problems and related works. While providing tight results for a masked AES S-box implementation with up to 6 shares, therefore opening the way towards tight random probing security in general, we note that our composition results are not completely tight in certain contexts which (we discuss in the paper and) could pop up in other circuits than the AES S-box. Hence, generalizing our results to be tight for any circuit is an interesting open problem and the same holds for optimizing the complexity of our verification techniques in order to scale with even larger circuits and number of shares.

Besides, we illustrated our results with the popular ISW multiplications in order to show their applicability to non-specialized gadgets, which are concretely relevant for the number of shares and noise levels we consider. Yet, since one of the motivations to use the random probing model is to capture horizontal attacks, it would also be interesting to analyze multiplication algorithms that provide improved guarantees against such attacks thanks to a logarithmic or even constant noise rate and could not be proven so far (e.g., [7,13]).

2 Background

Notations. In this work, we consider Boolean or arithmetic circuits over finite fields $\mathbb{F}_{2^m}$ and refer to the underlying additive and multiplicative operations as $\oplus$ and $\odot$, respectively. For the sake of simplicity we also use these operations for a share-wise composition of vectors $(v_i)_{i\in[n]}$ and $(w_i)_{i\in[n]}$ with $[n] = \{0, 1, \ldots, n-1\}$ such that $(v_i)_{i\in[n]} \odot (w_i)_{i\in[n]} := (v_i \odot w_i)_{i\in[n]}$ and $(v_i)_{i\in[n]} \oplus (w_i)_{i\in[n]} := (v_i \oplus w_i)_{i\in[n]}$. Furthermore, we use the Kronecker product to compose two real matrices $A = (a_{i,j})_{i\in[m],j\in[n]}$, $B = (b_{i,j})_{i\in[k],j\in[l]}$ such that $A \otimes B = (a_{i,j}B)_{i\in[m],j\in[n]}$. We also denote $x \xleftarrow{\$} \mathcal{X}$ as choosing x uniformly at random from the set $\mathcal{X}$, and $\mathcal{X}^{(k)}$ as the set of subsets of $\mathcal{X}$ of size k.

Masking. Masking is a well known countermeasure against side-channel attacks. With an encoding scheme $(\mathsf{Enc}(\cdot), \mathsf{Dec}(\cdot))$, sensitive data x is split into n shares (represented as a vector) $(x_i)_{i\in[n]} \leftarrow \mathsf{Enc}(x)$, and the decoding function takes as input the n shares and recovers the unshared value x, i.e., $x \leftarrow \mathsf{Dec}((x_i)_{i\in[n]})$. For security we require that any subset of $n-1$ shares does not reveal any information about the sensitive data x. In this work, we focus on additive sharing $\mathsf{Dec}((x_i)_{i\in[n]}) = \bigoplus_{i=0}^{n-1} x_i$, which is the most studied scheme.

Circuit model. As common in masking scheme literature, we model computation as arithmetic circuits operating over a finite field $\mathbb{F}_{2^m}$. The circuit is represented by a directed acyclic graph, where each node is a gate that has a fixed number of input and output wires (incoming and outgoing edges) that carry arithmetic values. We consider the following types of gates in our circuits: addition $\oplus$ and multiplication $\odot$ gates have two input wires and one output wire, and perform the corresponding arithmetic operation. The copy gate $©$ has one input and two outputs, and is used to duplicate a value. Finally, the random gate $®$ has no input and one output, which carries a uniformly distributed value. The constant gate $ⓐ$ outputs a constant value a.

In a masked circuit the gates are represented by subcircuits called gadgets G. These gadgets operate on encoded inputs and produce encoded outputs. The gadgets contain: (1) A set of gates; (2) The set of wires that connect the inputs and outputs of those gates named internal wires ($\mathcal{W}$); (3) The set of wires only connected with those gates' input named input wires ($\mathcal{I}$); (4) The set of output gates $\hat{\mathcal{O}}$ (which is the subset of its gates that output wires that are not connected to another gate of the gadget). The gadgets, however, contain no output wires, such that each wire in a circuit composed of multiple gadgets belongs to only one of its composing gadgets. For convenience, we also write $\mathcal{O}$ for the set of output wires of the gates in $\hat{\mathcal{O}}$, although these wires are not part of the gadget but are the next gadgets input wires. We denote $\mathcal{A} = \mathcal{W} \cup \mathcal{I}$ the set of all wires in the gadget. The inputs and outputs of a gadget are partitioned in (ordered) sets of n elements named sharings (and each element is a share). A gadget G_f that implements the function $f : \mathbb{F}^l \mapsto \mathbb{F}^k$ with n shares has l input sharings and k output sharings. Let $(y_i^0)_{i\in[n]}, \ldots, (y_i^{k-1})_{i\in[n]}$ be the values of the output sharings when the input sharings have the values $(x_i^0)_{i\in[n]}, \ldots, (x_i^{l-1})_{i\in[n]}$. It must hold that

$$f(\mathsf{Dec}((x_i^0)_{i\in[n]}), \ldots, \mathsf{Dec}((x_i^{l-1})_{i\in[n]})) = (\mathsf{Dec}((y_i^0)_{i\in[n]}), \ldots, \mathsf{Dec}((y_i^{k-1})_{i\in[n]})).$$

In this work, we use various gadgets. First, gadgets that implement linear operations (addition $\mathsf{G}_{\oplus}$, copy $\mathsf{G}_{©}$, squaring $\mathsf{G}_{\cdot^2}$), which we implement share-wise. Next, we use the ISW multiplication gadget [24]. Finally, we use refresh gadgets $\mathsf{G}_{®}$ which re-randomize a sharing $(x_i)_{i\in[n]}$ to $(y_i)_{i\in[n]}$ such that $\mathsf{Dec}((x_i)_{i\in[n]}) = \mathsf{Dec}((y_i)_{i\in[n]})$. We consider two refresh gadget implementations: the simple refresh and the SNI, randomness-optimized refresh gadgets from [12]. Their algorithmic description is given in Appendix A .

Leakage model. In this work we consider the p-random probing model as originally introduced by Ishai, Sahai and Wagner [24]. This model defines the following random probing experiment. Let $\mathcal{W}$ be a set of wires in a circuit, $\mathcal{L}_p(\mathcal{W})$ is a random variable with $\mathcal{L}_p(\mathcal{W}) \subseteq \mathcal{W}$, such that each wire $w \in \mathcal{W}$ is in $\mathcal{L}_p(\mathcal{W})$ with probability p (independently for each wire). Following this notation, for a gadget G, we denote by $\mathcal{L}_p(\mathsf{G}) := \mathcal{L}_p(\mathcal{W}, \mathcal{I}) := (\mathcal{L}_p(\mathcal{W}), \mathcal{L}_p(\mathcal{I}))$, where $\mathcal{W}$ and $\mathcal{I}$ are the set of internal and input wires of G, respectively.

For a gadget G, a set of probes is a successful attack for an input sharing $(x_i)_{i\in[n]}$ if the joint distribution of the values carried by the probes depends on $\mathsf{Dec}((x_i)_{i\in[n]})$ (assuming that the other input sharings are public). The security level of G in the p-random probing model (or p-random probing security) with respect to an input sharing $(x_i)_{i\in[n]}$ is the probability (over the randomness in $\mathcal{L}_p$) that a set of probes $\mathcal{L}_p(\mathsf{G})$ is a successful attack. As a result, the security of a gadget in bits is worth $-\log_2$(security level). We omit to mention the attacked input sharing when the gadget has only one input sharing.

3 Random probing security of small circuits

In this section, we show how to efficiently compute an upper bound on the random probing security level of relatively small gadgets, and we illustrate the results on well-known masked gadgets. We also describe the high-level ideas that will lead to the STRAPS tool that we describe in Section 5.3.

3.1 Derivation of a random probing security bound

We first derive a way to compute the security level of a gadget for various values of p, using some computationally heavy pre-processing. Next, we explain a way to use statistical confidence intervals to reduce the cost of the pre-processing. Finally, we detail how these techniques are implemented in a practical algorithm.

A simple bound. We can obtain the security level of a small circuit by computing first the statistical distribution of $\mathcal{L}_p(\mathsf{G})$ (i.e., $\Pr[\mathcal{L}_p(\mathcal{A}) = \mathcal{A}']$ for each subset $\mathcal{A}' \subset \mathcal{A}$). Then, for each possible set of probes $\mathcal{A}'$, we do a dependency test in order to determine if the set is a successful attack, denoted as $\delta_{\mathcal{A}'} = 1$, while $\delta_{\mathcal{A}'} = 0$ otherwise [8]. There exist various tools that can be used to carry out such a dependency test, such as maskVerif [4] or SILVER [25] (while such tools are designed to prove threshold probing security, they perform dependency tests as a sub-routine). A first naive algorithm to compute the security level ϵ is thus given by the equation

$$\epsilon = \sum_{\substack{\mathcal{A}' \subset \mathcal{A} \\ \text{s.t. } \delta_{\mathcal{A}'}=1}} \Pr[\mathcal{L}_p(\mathcal{A}) = \mathcal{A}']. \tag{1}$$

The computational cost of iterating over all possible probe sets grows exponentially with $|\mathcal{A}|$: for a circuit with $|\mathcal{A}|$ internal wires, one has to do $2^{|\mathcal{A}|}$

dependency tests, for each value of p (e.g., we have $|\mathcal{A}| = 57$ for the ISW multiplication with three shares). To efficiently cover multiple values of p, we introduce a first improvement to the naive algorithm given by Equation (1). For each $i \in \{0, \ldots, |\mathcal{A}|\}$, we compute the number c_i of sets of probes of size i that are successful attacks $c_i = \left|\left\{\mathcal{A}' \in \mathcal{A}^{(i)} \text{ s.t. } \delta_{\mathcal{A}'} = 1\right\}\right|$. Then, we can compute

$$\epsilon = \sum_{i=0}^{|\mathcal{A}|} p^i (1-p)^{|\mathcal{A}|-i} c_i, \tag{2}$$

which gives us a more efficient algorithm to compute random probing security, since it re-uses the costly computation of c_i for multiple values of p.

The VRAPS tool [8] computes c_i for small values of i by computing $\delta_{\mathcal{A}'}$ for all $\mathcal{A}' \in \mathcal{A}^{(i)}$. This is however computationally intractable for larger i values, hence they use the bound $c_i \leq \binom{|\mathcal{A}|}{i}$ in such cases.

A statistical bound. Let us now show how to improve the bound $c_i \leq \binom{|\mathcal{A}|}{i}$ while keeping a practical computational cost. At a high level, we achieve this by using a Monte-Carlo method whose idea is as follows: instead of computing directly ϵ, we run a randomized computation that gives us information about ϵ (but not its exact value). More precisely, the result of our Monte-Carlo method is a random variable ϵ^U that satisfies $\epsilon^U \geq \epsilon$ with probability at least $1 - \alpha$ (the confidence level), where α is a parameter of the computation. That is, $\Pr_{\mathrm{MC}}\left[\epsilon^U \geq \epsilon\right] \geq 1 - \alpha$, where $\Pr_{\mathrm{MC}}$ means the probability over the randomness used in the Monte-Carlo method.[2] In the rest of this work, we use $\alpha = 10^{-6}$ since we consider that it corresponds to a sufficient confidence level.[3]

Let us now detail the method. First, let $r_i = c_i / \left|\mathcal{A}^{(i)}\right|$. We remark that r_i can be interpreted as a probability: $r_i = \Pr_{\mathcal{A}' \xleftarrow{\$} \mathcal{A}^{(i)}}[\delta_{\mathcal{A}'} = 1]$. The Monte-Carlo method actually computes r_i^U such that $r_i^U \geq r_i$ with probability at least $1 - \alpha / \left(|\mathcal{A}| + 1\right)$. Once the r_i^U are computed, the result is

$$\epsilon^U = \sum_{i=0}^{|\mathcal{A}|} p^i (1-p)^{|\mathcal{A}|-i} \binom{|\mathcal{A}|}{i} r_i^U, \tag{3}$$

which ensures that $\epsilon^U \geq \epsilon$ for any p with confidence level $1 - \alpha$, thanks to the union bound. Next, r_i^U is computed by running the following experiment: take t_i samples $\mathcal{A}' \xleftarrow{\$} \mathcal{A}^{(i)}$ uniformly at random (this sampling is the random part of the Monte-Carlo method) and compute the number s_i of samples for which $\delta_{\mathcal{A}'} = 1$. By definition, s_i is a random variable that follows a binomial distribution $B(t_i, r_i)$: the total number of samples is t_i and the "success" probability is r_i.

[2] In other words, $[0, \epsilon^U]$ is a conservative confidence interval for ϵ with nominal coverage probability of $1 - \alpha$.

[3] This parameter is not critical: we can obtain a similar value for ϵ^U with higher confidence level by increasing the amount of computation: requiring $\alpha = 10^{-12}$ would roughly double the computational cost of the Monte-Carlo method.

We can thus use the bound derived in [33]. If r_i^U satisfies $\mathrm{CDF}_{\mathrm{binom}}(s_i; t_i, r_i^U) = \alpha/(|\mathcal{A}|+1)$, then $\Pr[r_i^U \geq r_i] = 1 - \alpha/(|\mathcal{A}|+1)$, which gives

$$r_i^U = \begin{cases} 1 & \text{if } s_i = t_i, \\ x \quad \text{s.t.} \quad I_x(s_i+1, t_i - s_i) = 1 - \alpha/(|\mathcal{A}|+1) & \text{otherwise,} \end{cases} \tag{4}$$

where $I_x(a,b)$ is the regularized incomplete beta function. We can similarly compute a lower bound ϵ^L such that $\epsilon^L \leq \epsilon$ with confidence coefficient $1-\alpha$, which we compute by replacing r_i^U with r_i^L in Equation (3), where:

$$r_i^L = \begin{cases} 0 & \text{if } s_i = 0, \\ x \quad \text{s.t.} \quad I_x(s_i, t_i - s_i + 1) = \alpha/(|\mathcal{A}|+1) & \text{otherwise.} \end{cases} \tag{5}$$

A hybrid algorithm. Our Monte-Carlo method has a main limitation: when $r_i = 0$ the bound r_i^U will not be null (it will be proportional to $1/t_i$). This means that we cannot prove tightly the security of interesting gadgets when p is small. For instance, let us take a fourth-order secure gadget (that is, $r_0 = r_1 = r_2 = r_3 = r_4 = 0$). If $r_1^U \neq 1$, then ϵ^U scales like $r_1^U p$ as p becomes small (other, higher degree, terms become negligible). A solution to this problem would be to set t_i to a large number, such that, in our example, r_1^U would be small enough to guarantee that $r_1^U p \ll r_5 p^5$ for all considered values of p. If we care about $p = 10^{-3}$, this means $r_1^U \ll 10^{-12} \cdot r_5 \leq 10^{-12}$. This is however practically infeasible since the number of samples t_1 is of the order of magnitude $1/r_1^U > 10^{12}$.

There exist another solution, which we call the hybrid algorithm: perform a full exploration of $\mathcal{A}^{(i)}$ (i.e., use the algorithm based on Equation (2)) when it is not computationally too expensive (i.e., when $\left|\mathcal{A}^{(i)}\right|$ is below some limit N_{max}), and otherwise use the Monte-Carlo method. The goal of this hybrid algorithm is to perform a full exploration when $r_i = 0$ (in order to avoid the limitation discussed above), which can be achieved for gadgets with a small number n of shares. Indeed, r_i can be null only for $i < n$ (otherwise there can be probes on all the shares of the considered input sharing), and the number of cases for the full exploration is therefore $\left|\mathcal{A}^{(i)}\right| = \binom{|\mathcal{A}|}{i} \leq \binom{|\mathcal{A}|}{n-1}$, which is smaller than N_{max} if n and $|\mathcal{A}|$ are sufficiently small. The latter inequality holds if $|\mathcal{A}| \geq 2(n-1)$, which holds for all non-trivial gadgets.

Algorithm 1 describes how we choose between full enumeration and Monte-Carlo sampling, which is the basis of our STRAPS tool (see Section 5.3 for more details). The algorithm adds a refinement on top of the above explanation: if we can cheaply show that r_i is far from zero, we do not perform full exploration even if it would not be too expensive. It accelerates the tool, while keeping a good bound. This optimization is implemented by always starting with a Monte-Carlo sampling loop that takes at most N_{max} samples, with an early stop if s_i goes above the value of a parameter N_t (we typically use parameters such that $N_{max} \gg N_t$). The parameter N_t determines the relative accuracy of the bound we achieve when we do the early stop: in the final sampling, we will have $s_i \approx N_t$, which means that the uncertainty on r_i decreases as N_t increases. The parameter

Algorithm 1 Random probing security algorithm: compute r_i^U, r_i^L for a given $\mathcal{A}$ and i. The parameters are N_{max} and N_t.

```
Require N_t ≤ N_max
N_sets = (|A| choose i)
t_i ← 1, s_i ← 0                          ▷ t_i: total number of samples, s_i: successful attacks
while t_i ≤ N_max ∧ s_i < N_t do                       ▷ First Monte-Carlo sampling loop
    A' ←$ A^(i)
    if δ_A' = 1 then
        s_i ← s_i + 1.
    t_i ← t_i + 1
if N_sets ≤ t_i then                 ▷ Enumerate A^(i) if it is cheaper than Monte-Carlo.
    s_i ← 0
    for all A' ∈ A^(i) do
        if δ_A' = 1 then
            s_i ← s_i + 1
    r_i^U ← s_i/N_sets, r_i^L ← s_i/N_sets
else                         ▷ Re-run Monte-Carlo to avoid bias due to N_t early stopping.
    s_i ← 0
    Repeat t_i times
        A' ←$ A^(i)
        if δ_A' = 1 then
            s_i ← s_i + 1
    Compute r_i^U and r_i^L using Equations (4) and (5).
```

N_{max} has an impact when r_i is small and we do not reach N_t successful attacks: it limits both the maximum size of $\mathcal{A}^{(i)}$ for which full exploration is performed, and the number of samples used for the Monte-Carlo method.

Remark. The Monte-Carlo method is limited to the random probing model and cannot be used to prove security in the threshold probing model since proving security in this model means proving that $r_i = 0$, which it cannot do. Our hybrid algorithm, however, can prove threshold probing security for the numbers of probes i where it does full enumeration of $\mathcal{A}^{(j)}$ for all $j \in \{0, \ldots, i\}$.

Dependency test. We use the dependency test algorithm from maskVerif [4], as it offers two important characteristics: (i) it gives the set of input shares on which the probes depend, not only if there is a dependency to the unshared variable (the reason for this appears in Section 5.1), and (ii) it is quite efficient. One drawback of the maskVerif dependency test is that in some cases, it wrongly reports that the adversary succeeds, which implies that the statistical lower bound is not anymore a lower bound for the security level, and the statistical upper bound is not completely tight (but it is still an upper bound for the true security level). In this case, we refer to the statistical lower bound as the *stat-only* lower bound. While the stat-only lower bound is not indicative of the security level, it remains useful to quantify the statistical uncertainty and therefore to assess whether one could improve the tightness of the upper bound by increasing the number of samples in the Monte Carlo method.

3.2 Security of some simple gadgets

We now present the results of random probing security evaluations using the previously described tools. First, we discuss the sharewise XOR gadget and the ISW multiplication gadget with n shares. Next, we discuss the impact of the two parameters of our algorithm (N_{max} and N_t) on the tightness of the results and on the computational complexity (i.e., the execution time) of the tool.

In Figure 1 (left), we show the security level (with respect to one of the inputs) of the addition gadget for $n = 1, \ldots, 6$ shares. We can see that the security level of the gadget is proportional to p^n, which is expected. Indeed, the graph of this share-wise gadget is made of n connected components (so-called "circuit shares" [12]) such that each share of a given input sharing belongs to a distinct component, and the adversary needs at least one probe in each of them to succeed. This trend can also be linked with the security order in the threshold probing model. Since the gadget is $n-1$-threshold probing secure, a successful attack contains at least n probes, hence has probability proportional to p^n.

We can observe a similar trend for the ISW multiplication gadget (Figure 1, right). Since the gadget is $n-1$-threshold probing secure, the security level scales proportionally to p^n for small values of p. For larger values of p, the security level of this gadget is worse than p^n, which is due to the larger number of wires, and the increased connectivity compared to the addition gadgets. It implies that there are many sets of probes of sizes $n+1$, $n+2$, $\ldots$ that are successful attacks (which is usually referred to as horizontal attacks in the practical side-channel literature [7]). These sets make up for a large part of the success probability when $p > 0.05$ due to their large number, even though they individually have a lower probability of occurring than a set of size n (for $p < 0.5$).

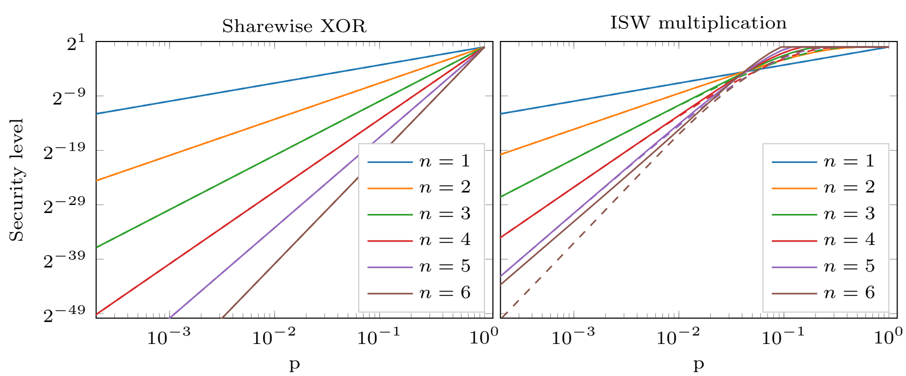


Fig. 1: Security of masked gadgets (with respect to the input sharing x, assuming the input sharing y is public). The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 10^7$, $N_t = 1000$.

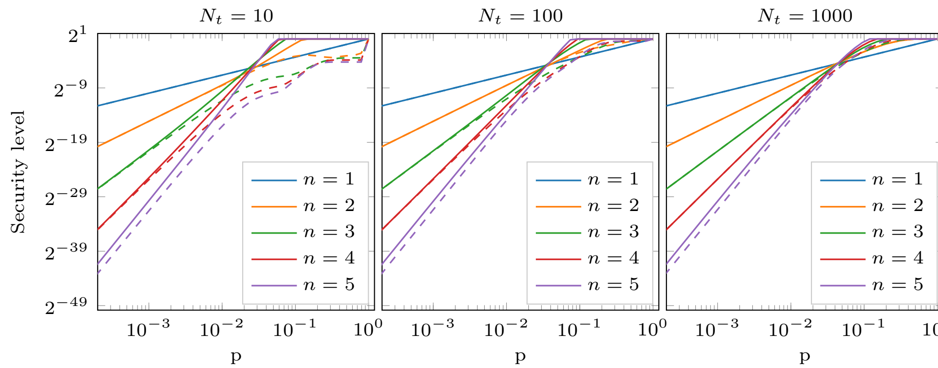


Fig. 2: Impact of the parameter N_t of Algorithm 1 on the security bounds of masked ISW multiplication gadgets (w.r.t. the input sharing x). $N_{max} = 10^7$.

Next, we discuss the impact of parameters N_{max} and N_t in Algorithm 1 on the tightness of the bounds we can compute. We first focus on the impact of N_t, which is shown on Figure 2. For $N_t = 10$, we have a significant distance between the statistical upper and lower bounds, while the gap becomes small for $N_t = 100$ and $N_t = 1000$. This gap appears as a bounded factor between the upper and lower bounds which, as discussed previously, is related to the accuracy of the estimate of a proportion when we have about N_t positive samples.

We also look at the impact of N_{max} on Figure 3. We observe a gap between the bounds for too low N_{max} values, which gets worse as the number of shares increases. Indeed, when N_{max} is too small, we cannot do an enumeration of all the sets of $n - 1$ probes, hence we cannot prove that the security order of the gadget is at least $n - 1$, which means that the upper bound is asymptotically proportional to $p^{n'}$, with $n' < n - 1$.

We finally observed that the computational cost is primarily dependent on N_{max} and the circuit size, while N_t has a lower impact (for the values considered). For instance, the execution time of the tool for the ISW multiplication with $n = 6$, $N_{max} = 10^8$ and $N_t = 100$ is about 33 h on a 24-core computer.

4 New composition results

In the previous section, it became clear that the tool is limited if it directly computes the security of complex circuits. This leads to the need to investigate composition properties. The existing definitions of random probing composability and random probing expandability in [8] are based on counting probes at the inputs and outputs of gadgets which are needed to simulate the leakage. We have recognized that ignoring the concrete random distribution over the needed input/output wires, and only counting the wires leads to a significant loss of tightness. Therefore we introduce our new security notion, the **PDT**. Before we define the **PDT** in Section 4.3 and present the composition results in Section 4.4, we recall the idea of simulatability in the leakage setting. Refining the dependency test of Section 3, we analyze the information a simulator needs to

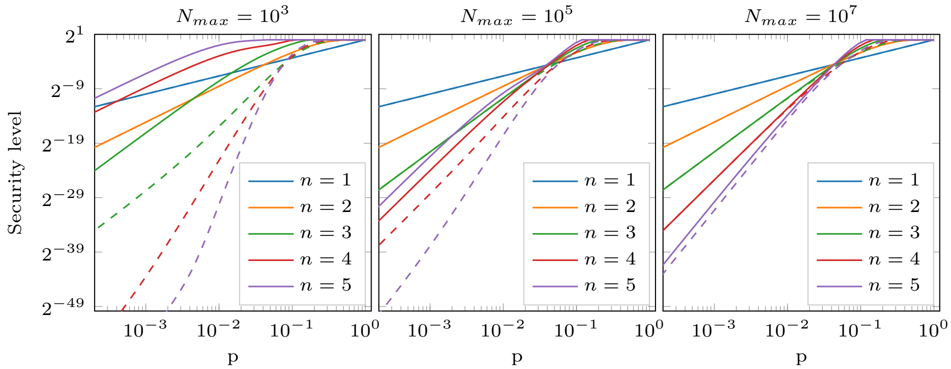


Fig. 3: Impact of the parameter N_{max} of Algorithm 1 on the security bounds of masked ISW multiplication gadgets (w.r.t. the input sharing x). $N_t = 1000$.

simulate a gadget's leakage in Section 4.2. In contrast to the previous section, we take into account the output gates, which is needed for composition. Further, we recall the definitions of parallel and sequential composition in Section 4.1, and present formal definitions adapted for our **PDT**s.

4.1 Definitions

Given two gadgets G_0 and G_1 with n shares, we define in this section the gadgets formed by their sequential composition written $\mathsf{G} = \mathsf{G}_1 \circ \mathsf{G}_0$ or their parallel composition written $\mathsf{G} = \mathsf{G}_1 || \mathsf{G}_0$.

We first introduce notations that allows us to keep track of input wires, output gates and internal wires in gadget compositions. We work with ordered finite sets. That is, given a finite set A (e.g., one of the sets $\mathcal{W}$, $\mathcal{I}$ or $\hat{\mathcal{O}}$ of a gadget G), we assign to each element of A a unique index in $[|A|] = \{0, 1, \ldots, |A|\}$. Then, given disjoint finite sets A and B, we denote by $C = A||_{(k)}B$ the union of A and B ordered such that a wire with index i in A has index i in C, and a wire with index i in B has index $k + i$ in B. The $||_{(\cdot)}$ operator is right-associative, which means that $A_2||_{(k_1)}A_1||_{(k_0)}A_0 = A_2||_{(k_1)}\left(A_1||_{(k_0)}A_0\right)$.

The sequential composition of gadgets allows implementing compositions of functions and is formally defined next.

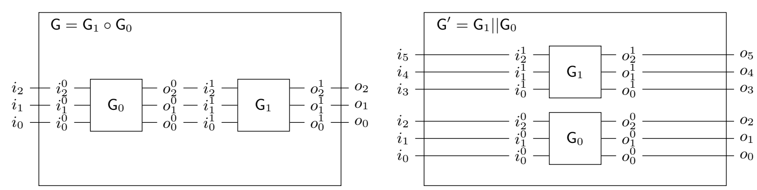
Definition 1 (Sequential composition). *Let G_0 and G_1 two gadgets with n shares, input wires $\mathcal{I}_i$, output gates $\hat{\mathcal{O}}_i$, and internal wires $\mathcal{W}_i$, respectively, such that $|\mathcal{I}_1| = |\hat{\mathcal{O}}_0|$. The sequential composition of G_0 and G_1 is the gadget G denoted as $\mathsf{G}_1 \circ \mathsf{G}_0$ whose set of input wires is $\mathcal{I} = \mathcal{I}_0$ and set of output gates is $\hat{\mathcal{O}} = \hat{\mathcal{O}}_1$. The set of internal wires of G is $\mathcal{W} = \mathcal{W}_1||_{(k_1)}\mathcal{I}_1||_{(k_0)}\mathcal{W}_0$ with $k_1 = |\mathcal{W}_0| + |\mathcal{I}_1|$ and $k_0 = |\mathcal{W}_0|$. The input wires of G_1 are connected to the output gates of G_0 such that for all i the input wire with index i is the output wire of the i^{th} output gate. If G_0 (resp. G_1) implements f_0 (resp. f_1), then G implements $f_1 \circ f_0$.*

The parallel composition of gadgets allows implementing a gadget for the function $f(x, y) = (f_0(x), f_1(y))$, using gadgets implementing f_0 and f_1.

Definition 2 (Parallel composition). *Let* G_0 *and* G_1 *two gadgets with* n *shares, input wires* $\mathcal{I}_i$*, output gates* $\hat{\mathcal{O}}_i$*, and internal wires* $\mathcal{W}_i$*, respectively. The parallel composition of* G_0 *and* G_1 *is the gadget* G *denoted as* $\mathsf{G}_1||\mathsf{G}_0$ *whose set of input wires is* $\mathcal{I} = \mathcal{I}_1||_{(|\mathcal{I}_0|)}\mathcal{I}_0$*, set of output gates is* $\hat{\mathcal{O}} = \hat{\mathcal{O}}_1||_{(|\hat{\mathcal{O}}_0|)}\hat{\mathcal{O}}_0$*, and set of internal wires is* $\mathcal{W} = \mathcal{W}_1||_{(|\mathcal{W}_0|)}\mathcal{W}_0$*.*

Figure 4 illustrates how to renumber the input wires and output gates in the case of gadgets with three inputs wires and three output gates. Figure 4a describes the sequential composition defined in Definition 1 and Figure 4b describes the parallel composition defined in Definition 2. For example, the input wire set of G' is $\mathcal{I} = \{i_5, i_4, \ldots, i_0\}$ which is the wire union $\mathcal{I} = \mathcal{I}_1||_{(|\mathcal{I}_0|)}\mathcal{I}_0$ of the input wires $\mathcal{I}_0 = \{i_2^0, i_1^0, i_0^0\}$ and $\mathcal{I}_1 = \{i_2^1, i_1^1, i_0^1\}$ of the gadgets G_0 and G_1.

We emphasize that both compositions are a basis for dividing a circuit into an arbitrary set of subcircuits. Therefore, if we have a masked gadget implementation of each gate type that appears in a circuit, we can build a masking compiler for that circuit: first decompose the circuit in sequential and parallel compositions down to subcircuits containing a single gate, then replace each gate with the corresponding masked gadget, and finally compose those gadgets according to the initial decomposition. As a case study, we depict a masked AES S-box implementation in Figure 6. The gadgets G_0-G_{10} are a parallel composition of the basis gadgets and $\mathsf{G}_{\text{S-box}}$ is a sequential composition of the gadgets G_0-G_{10}. The formal description of the S-box composition is given in Table 1.



(a) Sequential Composition (b) Parallel Composition

Fig. 4: Examples of sequential composition (4a) and parallel composition (4b).

4.2 Simulatability

So far, we described how to measure the amount of information leaked by a circuit by analyzing it directly. As observed in previous works, the complexity of such an approach rapidly turns out to be unrealistic. We now formalize simulatability-based definitions following the ideas outlined in [5], which are useful to analyze large circuits thanks to compositional reasoning.

Definition 3 (Simulatability). *A set of wires* $\mathcal{W}$ *in a gadget* G *is simulatable by a subset* $\mathcal{I}' \subset \mathcal{I}$ *of its inputs if there exists a probabilistic simulator function taking as input the values of the inputs* $\mathcal{I}'$*, and outputs a distribution of values on wires. Conditioned on the values of the wires in* $\mathcal{I}$ *the distribution output by the simulator is identical to the leakage from wires in* $\mathcal{W}$ *when the gadget is evaluated (conditioned on* $\mathcal{I}$*).*

The simulatability of small circuits, and particularly gadgets, is well studied and can be proven with tools such as maskVerif [4] and SILVER [25]. In this work we use the distribution of the smallest set of input wires such that there exists a simulator whose output has the same distribution as the leakage. More precisely, let $\mathcal{W}'$ be a subset of input and internal wires of a gadget G and $\mathcal{O}'$ an arbitrary subset of output wires, then we write $\mathcal{I}' = \mathcal{S}^{\mathsf{G}}(\mathcal{W}', \mathcal{O}')$ to define the smallest subset $\mathcal{I}'$ of input wires of G by which $(\mathcal{W}', \mathcal{O}')$ is perfectly simulatable.

Definition 4 (Simulatability set). *Let G be a gadget with input wire, internal wire and output gate sets $\mathcal{I}$, $\mathcal{W}$, and $\hat{\mathcal{O}}$. Further, let $\mathcal{O}$ be the set of output wires of $\hat{\mathcal{O}}$. The* simulatability set *of a subset $\mathcal{W}' \subseteq (\mathcal{W}, \mathcal{I})$ and $\mathcal{O}' \subseteq \mathcal{O}$, denoted $\mathcal{S}^{\mathsf{G}}(\mathcal{W}', \mathcal{O}')$, is the smallest subset of $\mathcal{I}$ by which $\mathcal{W}'$ and $\mathcal{O}'$ can be simulated.*

In the random probing model, $\mathcal{W}' = \mathcal{L}_p(\mathsf{G})$ is a random variable, hence the simulatability set $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ is itself a random variable.

We now introduce rules for simulatability of parallel and sequential gadget compositions. Indeed, it is not enough to give a simulator for each gadget, but we also have to ensure that each individual simulator is consistent with the distribution generated by the other simulators, and that each simulator is provided with correct values for the input shares.

Claim 1 *For any parallel gadget composition $\mathsf{G} = \mathsf{G}_1 || \mathsf{G}_0$ with output gates $\hat{\mathcal{O}} = \hat{\mathcal{O}}_1 ||_{(|\hat{\mathcal{O}}_1|)} \hat{\mathcal{O}}_0$ an its output wires $\mathcal{O}$. It holds that*

$$\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) \;||_{(|\mathcal{I}_0|)}\; \mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0)$$

for any subset of output wires $\mathcal{O}' = \mathcal{O}'_1 ||_{(|\mathcal{O}_0|)} \mathcal{O}'_0 \subseteq \mathcal{O}$.

The proof is given in Appendix B.1.

Claim 2 *For any sequential gadget composition $\mathsf{G} = \mathsf{G}_1 \circ \mathsf{G}_0$ with output gates $\hat{\mathcal{O}}$ and its output wires $\mathcal{O}$, it holds that*

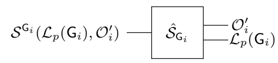
$$\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') \subseteq \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$$

for any subset of output wires $\mathcal{O}' \subseteq \mathcal{O}$.

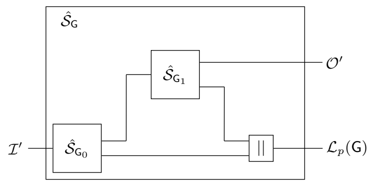
The proof is given in Appendix C.1.

4.3 Probe distributions

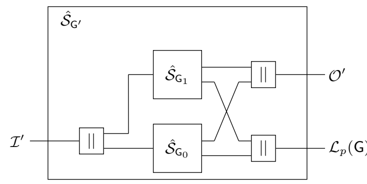
In this section, we introduce our new security properties, the **PD** (Probe Distribution) and the **PDT** (Probe Distribution Table). Intuitively, given a set of wires $\mathcal{W}$ and a leakage process $\mathcal{L}$ (hence $\mathcal{L}(\mathcal{W}) \subseteq \mathcal{W}$), the **PD** of $\mathcal{L}(\mathcal{W})$ is a vector of size $2^{|\mathcal{W}|}$ that represents the statistical distribution of $\mathcal{L}(\mathcal{W})$. In more detail, for each subset $\mathcal{W}' \subseteq \mathcal{W}$, there is a corresponding element of the **PD** with value $\Pr[\mathcal{L}(\mathcal{W}) = \mathcal{W}']$. The **PDT** notion extends the idea in a way that makes



(a) Tight Simulator for Gadget G_i used in the proof of Claim 1 and 2



(b) Simulator for a serial gadget compositions.



(c) Simulator for a parallel gadget compositions.

Fig. 5: Simulators for the gadgets depicted in Figure 4 to prove Claims 1 and 2.

it useful for analyzing gadget compositions: it links the set of output probes on the gadget to the distribution of the simulatability set of the gadget (i.e., to the inputs needed to simulate the leakage). More precisely, for a gadget G, the **PDT** is a matrix in $[0,1]^{|\mathcal{I}|\times|\mathcal{O}|}$, such that each column is associated to a subset of the outputs $\mathcal{O}' \subseteq \mathcal{O}$. Each column is a **PD** that represents the distribution of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}(\mathsf{G}), \mathcal{O}')$ (viewed as a subset of the set of inputs $\mathcal{I}$). The two main results (Theorems 1 and 2) of the next section relate the **PDT** of a sequential (resp., parallel) gadget composition to the matrix (resp., tensor) product of the **PDT**s of the composing gadgets. We first formalize the mapping between subsets of wires and indices in vectors/matrices.

Definition 5 (Index representation of subsets of wires). *For any set of wires $\mathcal{W}$ of which each element has a unique index in $[|\mathcal{W}|]$, we associate to each subset $\mathcal{W}'$ of $\mathcal{W}$ the index*

$$\tilde{\mathcal{W}'} = \sum_{i\in[|\mathcal{W}|]} b_i 2^i \quad \textit{with} \quad \begin{cases} b_i = 1 & \textit{if element } i \textit{ of } \mathcal{W} \textit{ belongs to } \mathcal{W}', \\ b_i = 0 & \textit{otherwise.} \end{cases}$$

For example, the wire set $\mathcal{W} = \{\omega_0, \omega_1\}$ has 4 subsets $\mathcal{W}'$, that we represent with their index below:

$$\begin{array}{c|cccc} \mathcal{W}' & \emptyset & \{\omega_0\} & \{\omega_1\} & \{\omega_0,\omega_1\} \\ \tilde{\mathcal{W}'} & 0 & 1 & 2 & 3 \end{array}$$

Let use now give the formal definition of the **PD**.

Definition 6 (Probe Distribution PD). *Let $\mathcal{L}$ be a probabilistic process that outputs subsets of a set of wires $\mathcal{W}$. The probe distribution (**PD**) of $\mathcal{L}$ with respect to $\mathcal{W}$ is $\mathbf{p} \in [0,1]^{2^{|\mathcal{W}|}}$ such that for all $\mathcal{W}' \subset \mathcal{W}$, $\mathbf{p}_{\tilde{\mathcal{W}'}} = Pr[\mathcal{L}(\mathcal{W}) = \mathcal{W}']$.*

The **PD** of $\mathcal{L}_p(\mathcal{W})$ in the previous example is $\mathbf{p} = \big((1-p)^2, p(1-p), p(1-p), p^2\big)$.

We next give the definition of the **PDT**, which can be seen as the **PD**s of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ conditioned on the set of output probes $\mathcal{O}'$.

Definition 7 (Probe Distribution Table (PDT)). *Let* G *be a gadget with input wires* $\mathcal{I}$ *and output wires* $\mathcal{O}$*. For any* $\mathcal{O}' \subseteq \mathcal{O}$*, let* $\mathbf{p}_{\tilde{\mathcal{O}}'}$ *be the* ***PD*** *of* $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$*. The* ***PDT*** *of* G *(*$\boldsymbol{PDT}_{\mathsf{G}}$*) is a* $[0,1]^{2^{|\mathcal{I}|}\times 2^{|\mathcal{O}|}}$ *matrix with all the* $\mathbf{p}_{\tilde{\mathcal{O}}'}$ *as columns, that is*

$$\boldsymbol{PDT}_{\mathsf{G}} = (\mathbf{p}_j)_{j\in[2^{|\mathcal{O}|}]},$$

with $j = \tilde{\mathcal{O}}'$ *for all subsets* $\mathcal{O}' \subseteq \mathcal{O}$*. The notation* $\boldsymbol{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ *refers to the element of* $\mathbf{p}_{\tilde{\mathcal{O}}'}$ *associated to* $\mathcal{I}'$*.*

$\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') = \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right]$. Furthermore, the **PDT** of a gadget is independent of its environment (i.e., of the **PD** of its output wires).

A first example of **PDT** is the one of the $\oplus$ and $\odot$ gates (when viewed as gadgets with one share). In the first column, no output has to be simulated, and thus the only leakage comes from the two input wires. For the second column, knowledge of both inputs is needed to simulate the output. This gives:

$\mathbf{PDT}_{\oplus} = \mathbf{PDT}_{\odot} =$

PDT	$\mathcal{O}' = \emptyset$	$\mathcal{O}' = \{0\}$
$\mathcal{I}' = \emptyset$	$(1-p^2)$	0
$\mathcal{I}' = \{0\}$	$p(1-p)$	0
$\mathcal{I}' = \{1\}$	$p(1-p)$	0
$\mathcal{I}' = \{0,1\}$	p^2	1

The second example is the simple refresh gadget G_r with two shares where a random value is added to two different wires. The random value leaks three times with probability p (one time in the copy gate and two times in the $\oplus$). Thus the leakage probability of the random value is $q = 1 - (1-p)^3$, and we get:

$\mathbf{PDT}_{\mathsf{G}_r} =$

PDT	$\mathcal{O}' = \emptyset$	$\mathcal{O}' = \{0\}$	$\mathcal{O}' = \{1\}$	$\mathcal{O}' = \{1,0\}$
$\mathcal{I}' = \emptyset$	$(1-p)^2$	$(1-q)(1-p)^2$	$(1-q)(1-p)^2$	0
$\mathcal{I}' = \{0\}$	$p(1-p)$	$(q+qp)(1-p)$	$(1-q)p(1-p)$	0
$\mathcal{I}' = \{1\}$	$p(1-p)$	$(1-q)p(1-p)$	$(q+(1-q)p)(1-p)$	0
$\mathcal{I}' = \{0,1\}$	p^2	$qp+(1-q)p^2$	$qp+(1-q)p^2$	1

The **PDT** is related to the security level in the random probing model.

Claim 3 (Security level from PDT) *Let* G *be a gadget and* $\boldsymbol{PDT}_{\mathsf{G}}$ *its Probe Distribution Table. Let* s *be the the security level of* G *with respect to an input sharing. If the set of shares of the considered input sharing is* $\mathcal{I}'$*, then*

$$\mathbf{e}^T \cdot \boldsymbol{PDT}_{\mathsf{G}} \cdot \mathbf{p}_{\emptyset} = \sum_{\mathcal{I}'' \supseteq \mathcal{I}'} \boldsymbol{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}'', 0) \geq s,$$

where $\mathbf{p}_{\emptyset} = (1, 0, \ldots, 0)$ *is the* ***PD*** *corresponding to no output leakage and* $e_i = 1$ *for all* $i = \tilde{\mathcal{I}}''$ *with* $\mathcal{I}'' \supseteq \mathcal{I}'$*, while* $e_i = 0$ *otherwise.*

Proof. Let $\mathcal{A}'$ be a set of wires that is an attack, that is, that depends on the considered unshared value which we denote Simulating $\mathcal{A}'$ therefore requires at least all the shares in $\mathcal{I}'$, hence

$$s \leq \Pr_{\mathcal{A}' \leftarrow \mathcal{L}_p(\mathsf{G})}\left[\mathcal{S}^{\mathsf{G}}(\mathcal{A}', \emptyset) \subseteq \mathcal{I}'\right].$$

Then, by definition of $\mathcal{L}_p(\mathsf{G})$ and of the **PDT**,

$$s \leq \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \emptyset) \subseteq \mathcal{I}'\right] = \sum_{\mathcal{I}'' \supseteq \mathcal{I}'} \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \emptyset) = \mathcal{I}''\right] = \sum_{\mathcal{I}'' \supseteq \mathcal{I}'} \mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}''}, 0).$$

This proves the inequality. The equality claim holds by construction of $\mathbf{e}$. □

We now give a few results that constitute the basis for the composition theorems of the next section. A first result links the **PD** of the input wires needed to simulate the leakage of the gadget and some of its outputs to the **PDT** of the gadget and the **PD** of its outputs. This claim is the foundation for the analysis of sequential gadget composition.

Claim 4 (PDT and PD) *Let G be a gadget with output wire set $\mathcal{O}$ and input wire set $\mathcal{I}$. If a probabilistic process $\mathcal{L}'(\mathcal{O})$ has a **PD** $\mathbf{p}$ with respect to $\mathcal{O}$, then $\boldsymbol{PDT}_{\mathsf{G}} \cdot \mathbf{p}$ is the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{L}'(\mathcal{O}))$ with respect to input wires $\mathcal{I}$.*

Proof. The solution can be directly derived from the definitions: Let $(v_i)_{i \in 2^{|\mathcal{I}|}} = \mathbf{PDT}_{\mathsf{G}} \cdot \mathbf{p}$. For any $\mathcal{I}' \subseteq \mathcal{I}$, it holds that

$$\begin{aligned}
v_{\tilde{\mathcal{I}}'} &= \sum_{\mathcal{O}' \subseteq \mathcal{O}} \mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') \cdot p_{\tilde{\mathcal{O}}'} \\
&= \sum_{\mathcal{O}' \subseteq \mathcal{O}} \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right] \cdot \Pr\left[\mathcal{L}'(\mathcal{O}) = \mathcal{O}'\right] \\
&= \sum_{\mathcal{O}' \subseteq \mathcal{O}} \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}', \mathcal{L}'(\mathcal{O}) = \mathcal{O}'\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{L}'(\mathcal{O})) = \mathcal{I}'\right].
\end{aligned}$$

The final equation gives the claim since it is exactly the i^{th} entry of the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{L}'(\mathcal{O}))$ with $i = \tilde{\mathcal{I}}'$. □

We next want to compare two probe distributions $\mathbf{p}$, $\mathbf{p}'$ to describe a partial order for distributions "$\dot{\leq}$". The high-level idea is that $\mathbf{p}$ is "larger" than $\mathbf{p}'$ (denoted $\mathbf{p} \dot{\geq} \mathbf{p}'$) if $\mathcal{L}$ gives more information than $\mathcal{L}'$. In other words, $\mathbf{p}$ is "larger" than $\mathbf{p}'$ if we can simulate $\mathcal{L}'(\mathcal{W})$ with $\mathcal{L}(\mathcal{W})$, where $\mathcal{L}$ (resp., $\mathcal{L}'$) is the probabilistic process associated to $\mathbf{p}$ (resp., $\mathbf{p}'$).

Definition 8 (Partial order for distributions). *For a set of wires $\mathcal{W}$, let $\mathcal{L}$ and $\mathcal{L}'$ be probabilistic processes with **PD**s $\mathbf{p}$ and $\mathbf{p}'$. We say that $\mathbf{p}$ is larger than $\mathbf{p}'$ and write $\mathbf{p} \dot{\geq} \mathbf{p}'$ iff the $\mathcal{L}'$ is simulatable by $\mathcal{L}$, that is, if there exists a probabilistic algorithm S that satisfies $S(\mathcal{X}) \subset \mathcal{X}$ such that the distribution of $\mathcal{L}'(\mathcal{W})$ and $S(\mathcal{L}(\mathcal{W}))$ are equal.*

On the one hand, it is clear that the definition is reflexive, antisymmetric, and transitive. Let $\mathbf{p}$, $\mathbf{p}'$, $\mathbf{p}''$ three **PD**s, it holds:

- $\mathbf{p} \dot{\geq} \mathbf{p}$, since we can always use the identity as simulator.

– If we know $\mathbf{p} \dot{\geq} \mathbf{p}'$ and $\mathbf{p} \dot{\leq} \mathbf{p}'$, both **PD**s describe processes with the same distribution, and we know $\mathbf{p} = \mathbf{p}'$.
– If it holds that $\mathbf{p} \dot{\geq} \mathbf{p}'$ and $\mathbf{p}' \dot{\geq} \mathbf{p}''$, it exists a simulator S' that simulates the process defined by $\mathbf{p}'$ with the process defined by $\mathbf{p}$, and a simulator S'' that does the same for $\mathbf{p}''$ and $\mathbf{p}'$. Hence, $S := S'(S''(\cdot))$ simulates the process defined by $\mathbf{p}''$ with the process of $\mathbf{p}$ and it follows $\mathbf{p} \dot{\geq} \mathbf{p}''$.

On the other hand, the order is only partial since it can happen that we have two probabilistic processes such that for both processes there exist no simulator to simulate the other.

The partial order for **PD**s is respected by linear combinations:

Claim 5 *Let $(\mathbf{p}_i)_{i \in [k]}$, $(\mathbf{p}'_i)_{i \in [k]}$ be **PD**s such that $\mathbf{p}_i \dot{\geq} \mathbf{p}'_i$ for all i. let $(\alpha_i)_{i \in [k]}$ be such that $0 \leq \alpha_i \leq 1$ for all i and $\sum_{i \in [k]} \alpha_i = 1$. If we denote $\mathbf{p} = \sum_{i \in [k]} \alpha_i \mathbf{p}_i$ and $\mathbf{p}' = \sum_{i \in [k]} \alpha_i \mathbf{p}'_i$, then $\mathbf{p}$ and $\mathbf{p}'$ are **PD**s and furthermore, $\mathbf{p} \dot{\geq} \mathbf{p}'$.*

Proof. Let $\mathcal{W}$ be a set of wires such that the random processes $(\mathcal{L}_i)_{i \in [k]}$ (resp. $(\mathcal{L}'_i)_{i \in [k]}$) have $(\mathbf{p}_i)_{i \in [k]}$ (resp. $(\mathbf{p}'_i)_{i \in [k]}$) as **PD**s. Further, let S^i be such that $S^i(\mathcal{L}_i(\mathcal{W}))$ has the same distribution as $\mathcal{L}'_i$. Let $\mathcal{L}$ be such that

$$\Pr[\mathcal{L}(\mathcal{W}) = \mathcal{W}'] = \sum_{i \in [k]} \alpha_i \Pr[\mathcal{L}_i(\mathcal{W}) = \mathcal{W}'],$$

and similarly for $\mathcal{L}'$. Firstly, $\mathcal{L}$ and $\mathcal{L}'$ are well-defined: the probabilities given above are non-negative and sum to 1. Next, the **PD** of $\mathcal{L}$ (resp. $\mathcal{L}'$) is $\mathbf{p}$ (resp. $\mathbf{p}'$). Finally, we build the simulator S. Let $\mathcal{L}''$ be a random process that, on input $\mathcal{W}$, selects randomly $i \in [k]$ (such that the probability of taking the value i is α_i), and outputs $S^i(\mathcal{L}_i(\mathcal{W}))$. Then, let S be a random process such that $\Pr[S(\mathcal{W}'') = \mathcal{W}'] = \Pr[\mathcal{L}'' = \mathcal{W}' | \mathcal{L} = \mathcal{W}'']$ for all $\mathcal{W}', \mathcal{W}'' \subseteq \mathcal{W}$. We observe that for all $\mathcal{W}' \subseteq \mathcal{W}$,

$$\begin{aligned} \Pr[S(\mathcal{L}) = \mathcal{W}'] &= \sum_{\mathcal{W}'' \subseteq \mathcal{W}} \Pr[S(\mathcal{W}'') = \mathcal{W}'] * \Pr[\mathcal{L} = \mathcal{W}''] \\ &= \sum_{\mathcal{W}'' \subseteq \mathcal{W}} \Pr[\mathcal{L}'' = \mathcal{W}' | \mathcal{L} = \mathcal{W}''] * \Pr[\mathcal{L} = \mathcal{W}''] \\ &= \Pr[\mathcal{L}'' = \mathcal{W}']. \end{aligned}$$

Since $\mathcal{L}''$ has the same distribution as $\mathcal{L}'$, this means that $\Pr[S(\mathcal{L}) = \mathcal{W}'] = \Pr[\mathcal{L}' = \mathcal{W}']$. □

The **PDT** has a partial structure. As described above each column i of the **PDT** is the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ with $\tilde{\mathcal{O}}' = i$. Since we know that the input set required by a leakage simulator can only grow (or stay constant) if it has to simulate additional (output) leakage, we get:

Claim 6 *For any gadget with output wires $\mathcal{O}$, the columns $\mathbf{p}_{\cdot}$ of the **PDT** have the following property: $\mathbf{p}_{\tilde{\mathcal{O}}'} \dot{\geq} \mathbf{p}_{\tilde{\mathcal{O}}''}$ for all $\mathcal{O}'' \subseteq \mathcal{O}' \subseteq \mathcal{O}$.*

Proof. It follows directly from claim 4. It holds that $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}),\mathcal{O}'') \subseteq \mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}),\mathcal{O}')$ and thus $\Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}),\mathcal{O}'') \subseteq \mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}),\mathcal{O}')\right] = 1$. The last equation is the claim $p_{\tilde{\mathcal{O}}'} \dot{\geq} p_{\tilde{\mathcal{O}}''}$. □

Finally, we want to extend the partial order of **PD**s to the whole **PDT**, with the same meaning: if $\mathbf{PDT}_{\mathsf{G}_0} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_1}$, the amount of information leaked in G_0 is less than the information leaked in G_1:

Definition 9 (Partial order for PDT's). *Let $\mathbf{A},\mathbf{B} \in [0,1]^{2^{|\mathcal{I}|}\times 2^{|\mathcal{O}|}}$ be two **PDT**s, we write*

$$\mathbf{A} \dot{\leq} \mathbf{B}$$

*if for any **PD** $\mathbf{p} \in [0,1]^{2^{|\mathcal{O}|}}$ it holds $\mathbf{A}\cdot\mathbf{p} \dot{\leq} \mathbf{B}\cdot\mathbf{p}$.*

As shown in Claim 4, $\mathbf{A}\cdot\mathbf{p}$ and $\mathbf{B}\cdot\mathbf{p}$ are **PD**s, therefore the partial order of **PDT**s is well defined.

Corollary 1 (PDT order is column-wise). *Let **PDT** and **PDT'** be **PDT**s, with columns $(\mathbf{p}_i)_{i\in[|\mathcal{O}|]}$ and $(\mathbf{p}'_i)_{i\in[|\mathcal{O}|]}$ respectively. Then, $\mathbf{PDT} \dot{\geq} \mathbf{PDT}'$ iff $\mathbf{p}_i \dot{\geq} \mathbf{p}'_i$ for all $i \in [|\mathcal{O}|$.*

Proof. If $\mathbf{PDT} \dot{\geq} \mathbf{PDT}'$, then for any $i \in [|\mathcal{O}|$, let $\mathbf{e}$ be such that $e_j = 1$ if $i = j$ and $e_j = 0$ otherwise. Since $\mathbf{e}$ is a **PD**, we have $\mathbf{p}_i = \mathbf{PDT}\cdot\mathbf{e} \dot{\geq} \mathbf{PDT}'\cdot\mathbf{e} = \mathbf{p}'_i$.

In the other way, let use assume that $\mathbf{p}_i \dot{\geq} \mathbf{p}'_i$, for all i. Then for any **PD** α (whose elements are denoted α_i), $\mathbf{PDT}\cdot\alpha$ is a linear combination of $\mathbf{p}_i$ with coefficients α_i, for which Claim 5 applies. Therefore $\mathbf{PDT}\cdot\alpha \dot{\geq} \mathbf{PDT}'\cdot\alpha$. □

Another useful property is that we can merge the order of **PD**s and **PDT**s:

Claim 7 *Let $\mathbf{A},\mathbf{B} \in [0,1]^{2^{|\mathcal{I}|}\times 2^{|\mathcal{O}|}}$ be two **PDT**s, and $\mathbf{p},\mathbf{p}' \in [0,1]^{2^{|\mathcal{O}|}}$ be two **PD**s. If $\mathbf{A} \dot{\leq} \mathbf{B}$ and $\mathbf{p} \dot{\leq} \mathbf{p}'$, then $\mathbf{A}\cdot\mathbf{p} \dot{\leq} \mathbf{B}\cdot\mathbf{p}'$.*

Proof. We prove the claim $\mathbf{A}\cdot\mathbf{p} \dot{\leq} \mathbf{B}\cdot\mathbf{p}'$ in two steps. First we show (i) $\mathbf{A}\cdot\mathbf{p} \dot{\leq} \mathbf{A}\cdot\mathbf{p}'$, and then we show (ii) $\mathbf{A}\cdot\mathbf{p}' \dot{\leq} \mathbf{B}\cdot\mathbf{p}'$.

(i) By Definition 8, there exists $\mathcal{W}$, $\mathcal{L}$ and $\mathcal{L}'$ associated to $\mathbf{p}$, $\mathbf{p}'$, respectively, with $\Pr[\mathcal{L}(\mathcal{W}) \subset \mathcal{L}'(\mathcal{W})] = 1$. Further, it holds $\Pr[\mathbf{A}_{\mathcal{L}(\mathcal{W})} \dot{\leq} \mathbf{A}_{\mathcal{L}'(\mathcal{W})}] = 1$ with Claim 6. Hence, $\mathbf{A}\cdot\mathbf{p} \dot{\leq} \mathbf{A}\cdot\mathbf{p}'$.

(ii) $\mathbf{A}\cdot\mathbf{p}' \dot{\leq} \mathbf{B}\cdot\mathbf{p}'$ follows from Definition 9 and $\mathbf{A} \dot{\leq} \mathbf{B}$. □

This leads to the preservation of **PDT** ordering through matrix product.

Corollary 2. *Let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ be **PDT**s. If $\mathbf{A} \dot{\leq} \mathbf{B}$ and $\mathbf{C} \dot{\leq} \mathbf{D}$, then $\mathbf{A}\cdot\mathbf{C} \dot{\leq} \mathbf{B}\cdot\mathbf{D}$.*

Proof. Let us denote by $\mathbf{X}_{*,i}$ the $(i+1)$-th column of a matrix $\mathbf{X}$. Then, for all $i \in [|\mathcal{O}|]$, $(\mathbf{A}\cdot\mathbf{C})_{*,i} = \mathbf{A}\cdot\mathbf{C}_{*,i}$ and $(\mathbf{B}\cdot\mathbf{D})_{*,i} = \mathbf{B}\cdot\mathbf{D}_{*,i}$. Hence, by Corollary 1, $\mathbf{A}\cdot\mathbf{C} \dot{\leq} \mathbf{B}\cdot\mathbf{D}$ iff $\mathbf{C}_{*,i} \dot{\leq} \mathbf{D}_{*,i}$ for all i. Using the same Corollary, we have $\mathbf{C}_{*,i} \dot{\leq} \mathbf{D}_{*,i}$. Finally, using Claim 7, we get $\mathbf{A}\cdot\mathbf{C}_{*,i} \dot{\leq} \mathbf{B}\cdot\mathbf{D}_{*,i}$ for all i. □

Finally, we relate the partial order for **PD**s and **PDT**s to the security level.

Claim 8 (Security level bound from PDT bound) *Let s be the security level of a gadget G with respect to a set of input shares $\mathcal{I}'$. Let $\boldsymbol{PDT}$ be the $\boldsymbol{PDT}$ of G and let $\boldsymbol{PDT'}$ be a $\boldsymbol{PDT}$. If $\boldsymbol{PDT'} \dot{\geq} \boldsymbol{PDT}$, then $\mathbf{e}^T \cdot \boldsymbol{PDT'} \cdot \mathbf{p}_\emptyset \geq s$, where $\mathbf{e}$ is defined as in Claim 3.*

Proof. Using Claim 3, we know that $\mathbf{e}^T \cdot \mathbf{PDT} \cdot \mathbf{p}_\emptyset \geq s$. With Claim 7, we know that $\mathbf{PDT}' \cdot \mathbf{p}_\emptyset \dot{\geq} \mathbf{PDT} \cdot \mathbf{p}_\emptyset$. Let $\mathcal{L}$ (resp. $\mathcal{L}'$) be the random process associated to $\mathbf{PDT}' \cdot \mathbf{p}_\emptyset$ (resp. $\mathbf{PDT} \cdot \mathbf{p}_\emptyset$), and let S be the simulator that simulates $\mathcal{L}$ from $\mathcal{L}'$. We have $S(\mathcal{L}'(\mathcal{I})) \subseteq \mathcal{L}'(\mathcal{I})$, hence $\Pr[\mathcal{I}' \subseteq S(\mathcal{L}'(\mathcal{I}))] \leq \Pr[\mathcal{I}' \subseteq \mathcal{L}'(\mathcal{I})]$. Since S simulates $\mathcal{L}(\mathcal{I})$, $\Pr[\mathcal{I}' \subseteq S(\mathcal{L}'(\mathcal{I}))] = \Pr[\mathcal{I}' \subseteq \mathcal{L}(\mathcal{I})]$, which leads to $\mathbf{e}^T \cdot \mathbf{PDT} \cdot \mathbf{p}_\emptyset = \Pr[\mathcal{I}' \subseteq \mathcal{L}(\mathcal{I})] \leq \Pr[\mathcal{I}' \subseteq \mathcal{L}'(\mathcal{I})] = \mathbf{e}^T \cdot \mathbf{PDT}' \cdot \mathbf{p}_\emptyset$. □

4.4 Composition rules

In this section, we give the two main composition theorems for the **PDT** of parallel and sequential gadget compositions. Next, we show how the compositions theorems can be used to compute **PDT**s for larger composite gadgets and illustrate our results on the AES S-box example.

Theorem 1 (parallel composition). *Let G_1 and G_2 be two gadgets with $\boldsymbol{PDT}_{\mathsf{G}_0}$ and $\boldsymbol{PDT}_{\mathsf{G}_1}$. Further let $\mathsf{G} = \mathsf{G}_1 || \mathsf{G}_0$ with $\boldsymbol{PDT}_{\mathsf{G}}$. It holds that*

$$\boldsymbol{PDT}_{\mathsf{G}} = \boldsymbol{PDT}_{\mathsf{G}_1} \otimes \boldsymbol{PDT}_{\mathsf{G}_0}.$$

Proof. Let $\mathcal{I}_0$, $\mathcal{I}_1$, $\mathcal{O}_0$, and $\mathcal{O}_1$ the input and output wires of G_0 and G_1, respectively. Hence, $\mathcal{I} = \mathcal{I}_1 ||_{(n)} \mathcal{I}_0$, $\mathcal{O} = \mathcal{O}_1 ||_{(m)} \mathcal{O}_0$ are the input and output wires of G with $n = |\mathcal{I}_0|$ and $m = |\mathcal{O}_0|$. From Definition 2 follows for any $\mathcal{I}' = \mathcal{I}'_1 ||_{(n)} \mathcal{I}'_0 \subseteq \mathcal{I}$ and $\mathcal{O}' = \mathcal{O}'_1 ||_{(m)} \mathcal{O}'_0 \subseteq \mathcal{O}$ that $\Pr[\mathcal{S}(\mathcal{L}_p(\mathsf{G}) \cup \mathcal{O}') = \mathcal{I}']$ is the matrix entry $(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ of $\mathbf{PDT}_{\mathsf{G}}$. Considering Claim 1, we get

$$\begin{aligned}
\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') &= \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_1}\left(\mathcal{L}_p(\mathsf{G}_1) \cup \mathcal{O}'_1\right) ||_{(n)} \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0\right) = \mathcal{I}'_1 ||_{(n)} \mathcal{I}'_0\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_1}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0\right) = \mathcal{I}'_0, \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1\right) = \mathcal{I}'_1\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0) = \mathcal{I}'_0\right] \cdot \Pr\left[\mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) = \mathcal{I}'_1\right] \\
&= \mathbf{PDT}_{\mathsf{G}_0}(\tilde{\mathcal{I}}'_0, \tilde{\mathcal{O}}'_0) \cdot \mathbf{PDT}_{\mathsf{G}_1}(\tilde{\mathcal{I}}'_1, \tilde{\mathcal{O}}'_1).
\end{aligned}$$

The last transformation of the formula uses the fact that the set of probes of both gadgets are independent, and the resulting term is exactly the matrix entry $(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ of $\mathbf{PDT}_{\mathsf{G}_1} \otimes \mathbf{PDT}_{\mathsf{G}_0}$. □

Remark. Theorem 1 can be generalized to any parallel composition of sub-circuits, even if those sub-circuits are not gadgets. For instance, a share-wise gadget with n shares is the parallel composition of n identical sub-circuits (a single addition gate for the addition gadget). The **PDT** of the addition gate $\mathbf{PDT}_{\oplus}$ is given in Section 4.3, therefore $\mathbf{PDT}_{\mathsf{G}_{\oplus,n}}$ can be computed as

$$\mathbf{PDT}_{\mathsf{G}_{\oplus,n}} = P\left(\bigotimes_{i=0}^{n-1} \mathbf{PDT}_{\oplus}\right),$$

where P reorders the index of the input wires from $(x_0^0, x_0^1, x_1^0, x_1^1, \dots x_{n-1}^0, x_{n-1}^1)$ to $(x_0^0, \dots, x_{n-1}^0, x_0^1, \dots, x_{n-1}^1)$ where x_i^0 and x_i^1 are the first and second input wires of the i^{th} addition gate, respectively.

Theorem 2 (sequential composition). *Let G_0 and G_1 be two gadgets with $\boldsymbol{PDT}_{\mathsf{G}_0}$, $\boldsymbol{PDT}_{\mathsf{G}_1}$, and with n_i input wires and m_i output wires, respectively such that $m_0 = n_1$. Further let $\mathsf{G} = \mathsf{G}_1 \circ \mathsf{G}_0$ with $\boldsymbol{PDT}_{\mathsf{G}}$. It holds that*

$$\boldsymbol{PDT}_{\mathsf{G}} \dot{\leq} \boldsymbol{PDT}_{\mathsf{G}_0} \cdot \boldsymbol{PDT}_{\mathsf{G}_1}.$$

Proof. Let $\overline{\mathbf{PDT}} = \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1}$ and $\mathcal{I}_0$, $\mathcal{I}_1$, $\mathcal{O}_0$, $\mathcal{O}_1$ the input and output wire sets of G_0 and G_1, respectively. It also means that $\mathcal{I}_0$ and $\mathcal{O}_1$ are the input and output wire sets of G. Considering the fact that $\overline{\mathbf{PDT}}$ is the result of a matrix multiplication of $\mathbf{PDT}_{\mathsf{G}_0}$ and $\mathbf{PDT}_{\mathsf{G}_1}$, we get for any $\mathcal{I}' \subseteq \mathcal{I}_0$ and $\mathcal{O}' \subseteq \mathcal{O}_1$

$$\begin{aligned}
\overline{\mathbf{PDT}}(\tilde{\mathcal{I}''}, \tilde{\mathcal{O}'}) &= \sum_{\mathcal{O}'' \subseteq \mathcal{O}_0} \Pr\left[\mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'') = \mathcal{I}'\right] \cdot \Pr\left[\mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}') = \mathcal{O}''\right] \\
&= \sum_{\mathcal{O}'' \subseteq \mathcal{O}_0} \Pr\left[\mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'') = \mathcal{I}', \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}') = \mathcal{O}''\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right) = \mathcal{I}'\right].
\end{aligned}$$

Further, $\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}'}, \tilde{\mathcal{O}'}) = \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right]$, and thus for any $\mathcal{O}' \subseteq \mathcal{O}_1$ the columns $\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{O}'})$ and $\overline{\mathbf{PDT}}(\tilde{\mathcal{O}'})$ are the **PD**s of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ and of $\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$, respectively. Because of Claim 2, it holds that

$$\Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') \subseteq \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)\right] = 1.$$

The last equation proves that it exists a simulator that simulates the simulatability set $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ with $\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$. Hence, it holds that $\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{O}'}) \dot{\leq} \overline{\mathbf{PDT}}(\tilde{\mathcal{O}'})$ for any column with $\mathcal{O}' \subseteq \mathcal{O}_1$. Since the inequality holds for any column, the inequality is independent from the distribution of the output wires $\mathcal{O}_1$. It follows that $\mathbf{PDT}_{\mathsf{G}}\mathbf{p} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1}\mathbf{p}$ for all **PD**s $\mathbf{p}$. This results in the claim of the theorem $\mathbf{PDT}_{\mathsf{G}} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1}$. □

Corollary 3. *Let $(\mathsf{G}_i)_{i \in [k]}$ be gadgets that can be sequentially composed to form $\mathsf{G} = \mathsf{G}_{k-1} \circ \dots \circ \mathsf{G}_0$. It holds that*

$$\boldsymbol{PDT}_{\mathsf{G}} \dot{\leq} \boldsymbol{PDT}_{\mathsf{G}_0} \cdot \ldots \cdot \boldsymbol{PDT}_{\mathsf{G}_{k-1}}.$$

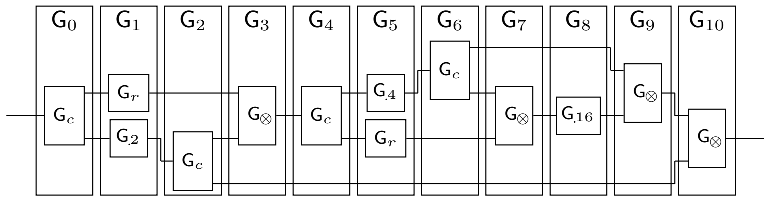


Fig. 6: AES S-box circuit (using the implementation from [31]) as a serial composition of gadgets. The symbols G_c, G_r, $\mathsf{G}_\otimes$ and $\mathsf{G}_{\cdot x}$ are respectively copy, refresh and exponentiation to the power of x gadgets.

Proof. This is a direct consequence of Theorem 2 and Corollary 2. $\square$

The **PDT** of the AES S-box depicted in Figure 6 is bounded by $\mathbf{PDT}_{\text{S-box}}$ defined in Table 1. We compute the S-box with the gadgets $\mathsf{G}_{\cdot 2}$, $\mathsf{G}_\otimes$, G_r, and G_c. In addition, we also use a identity gadget G_{id}^l as a placeholder for composition results (this gadget does not leak and has as many inputs as outputs), whose **PDT** is the identity matrix. As described in Table 1, the gadgets G_0-G_{10} are a parallel composition of the gadgets $\mathsf{G}_{\cdot 2}$, $\mathsf{G}_{\cdot 4}$, $\mathsf{G}_{\cdot 16}$, $\mathsf{G}_\otimes$, G_r, G_c, and G_{id}^l (we can compute their **PDT**s using Theorem 1). Thus, $\mathsf{G}_{\text{S-box}}$ is a sequential composition of G_0-G_{10}. We can compute its **PDT** using Corollary 3, as shown in Table 1.

G_0	G_c	$\mathbf{PDT}_{\mathsf{G}_0} = \mathbf{PDT}_{\mathsf{G}_c}$
G_1	$\mathsf{G}_r \Vert \mathsf{G}_{\cdot 2}$	$\mathbf{PDT}_{\mathsf{G}_1} = \mathbf{PDT}_{\mathsf{G}_r} \otimes \mathbf{PDT}_{\mathsf{G}_{\cdot 2}}$
G_2	$\mathsf{G}_{id} \Vert \mathsf{G}_c$	$\mathbf{PDT}_{\mathsf{G}_2} = \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_c}$
G_3	$\mathsf{G}_\otimes \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_3} = \mathbf{PDT}_{\mathsf{G}_\otimes} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_4	$\mathsf{G}_c \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_4} = \mathbf{PDT}_{\mathsf{G}_c} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_5	$\mathsf{G}_{\cdot 4} \Vert \mathsf{G}_r \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_5} = \mathbf{PDT}_{\mathsf{G}_{\cdot 4}} \otimes \mathbf{PDT}_{\mathsf{G}_r} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_6	$\mathsf{G}_c \Vert \mathsf{G}_{id} \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_6} = \mathbf{PDT}_{\mathsf{G}_c} \otimes \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_7	$\mathsf{G}_{id} \Vert \mathsf{G}_\otimes \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_7} = \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_\otimes} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_8	$\mathsf{G}_{id} \Vert \mathsf{G}_{\cdot 16} \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_8} = \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_{\cdot 16}} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_9	$\mathsf{G}_\otimes \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_9} = \mathbf{PDT}_{\mathsf{G}_\otimes} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_{10}	$\mathsf{G}_\otimes$	$\mathbf{PDT}_{\mathsf{G}_{10}} = \mathbf{PDT}_{\mathsf{G}_\otimes}$
$\mathsf{G}_{\text{S-box}}$	$\mathsf{G}_{10} \circ \mathsf{G}_9 \circ \ldots \circ \mathsf{G}_0$	$\mathbf{PDT}_{\text{S-box}} \leq \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1} \cdot \ldots \cdot \mathbf{PDT}_{\mathsf{G}_{10}}$

Table 1: Composition of the AES S-box and its approximated **PDT**.

We conclude by noting that some well-known matrix product and tensor product distributive and associative properties mirror the properties of the gadget compositions (when the operations are well-defined):

$$(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{C} = \mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{C}) \qquad (\mathsf{G}_0 \circ \mathsf{G}_1) \circ \mathsf{G}_2 = \mathsf{G}_0 \circ (\mathsf{G}_1 \circ \mathsf{G}_2)$$
$$(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) \qquad (\mathsf{G}_0 \Vert \mathsf{G}_1) \Vert \mathsf{G}_2 = \mathsf{G}_0 \Vert (\mathsf{G}_1 \Vert \mathsf{G}_2)$$
$$(\mathbf{A} \cdot \mathbf{B}) \otimes (\mathbf{C} \cdot \mathbf{D}) = (\mathbf{A} \otimes \mathbf{C}) \cdot (\mathbf{B} \otimes \mathbf{D}) \qquad (\mathsf{G}_0 \circ \mathsf{G}_1) \Vert (\mathsf{G}_2 \circ \mathsf{G}_3) = (\mathsf{G}_0 \Vert \mathsf{G}_2) \circ (\mathsf{G}_1 \Vert \mathsf{G}_3)$$

This means that our composition theorems give the same result independently of the way we decompose a composite gadget. This gives us freedom to choose, e.g., the most efficient way when we deal with relatively large computations.

5 Practical security of composite circuits

In this section, we adapt the method of Section 3 to compute bounds for **PDT**s. We then show how to turn those bounds into gadget security levels using the **PDT** properties and composition theorems. We finally describe the tool that implements our methodology and discuss its result for well-known gadgets.

5.1 Bounding PDTs

We first describe how to adapt the method of Section 3 to bound **PDT**s. That is, given a gadget G, we want to generate an upper bound $\mathbf{PDT}^U$ such that $\mathbf{PDT}^U \geq \mathbf{PDT}$ with probability at least $1-\alpha$ (e.g., $1-10^{-6}$), and the $\geq$ operator defined for matrices and vectors as element-wise. We note that $\mathbf{PDT}^U$ is not a **PDT**: the sum of the elements in one of its columns may be ≥ 1.

There are two main differences with the bound of Section 3: (1) we have to handle all possible cases for the probes on the output shares of the gadgets (i.e., all the columns of the **PDT**), and (2) we care about the full distribution of the input probes, not only the probability of successful attack.

The upper bound $\mathbf{PDT}^U$ can be computed by grouping probe sets by size (similarly to Equation (3)):

$$\mathbf{PDT}^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') = \sum_{i=0}^{|\mathcal{W}|} p^i (1-p)^{|\mathcal{W}|-i} \cdot \left|\mathcal{W}^{(i)}\right| \cdot \mathbf{R}_i^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$$

satisfies $\mathbf{PDT}^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') \geq \mathbf{PDT}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ if

$$\mathbf{R}_i^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') \geq \frac{\left|\left\{\mathcal{W}' \subseteq \mathcal{W}^{(i)} \text{ s.t. } \mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right\}\right|}{\left|\mathcal{W}^{(i)}\right|} \tag{6}$$

for all $i \in \{0, \dots, |\mathcal{W}|\}$. Therefore, if Equation (6) is satisfied for each $(\mathcal{I}', \mathcal{O}', i)$ tuple with probability at least $1 - \alpha / \left((|\mathcal{W}|+1)\, 2^{|\mathcal{I}|\cdot|\mathcal{O}|}\right)$, then $\mathbf{PDT}^U \geq \mathbf{PDT}$ with probability at least $1-\alpha$ (by the union bound).

The computation of all the elements $P_i^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ can be performed identically to the computation of r_i^U in Section 3.1, except for changing the criterion for a Monte-Carlo sample $\mathcal{W}'$ to be counted as positive (i.e., be counted in s_i): $\mathcal{S}(\mathcal{W}', \mathcal{O}') = \mathcal{I}'$ (instead of $\delta_{\mathcal{W}'} = 1$). Furthermore, the algorithm can be optimized by running only one sampling for each $(i, \mathcal{O}')$ pair: we take $t_{i,\mathcal{O}'}$ samples, and we classify each sample $\mathcal{W}'$ according to $\mathcal{S}(\mathcal{W}', \mathcal{O}')$. This gives sample counts $s_{i,\mathcal{O}',\mathcal{I}'}$ for all $\mathcal{I}' \subseteq \mathcal{I}$, and from there we can use Equation (4).[4]

[4] The random variables $s_{i,\mathcal{O}',\mathcal{I}'}$ for all $\mathcal{I}' \subseteq \mathcal{I}$ are not mutually independent, hence the derived bounds are not independent from each other, but this is not an issue since the union bound does not require independent variables.

Finally, we use the hybrid strategy of Algorithm 1, with the aforementioned modifications.[5] The computation of a statistical-only lower bound $\mathbf{PDT}^L$ is done in the same way, except that Equation (5) is used instead of Equation (4).

5.2 From PDT bound to security level bound.

Let us take positive matrices $A^U \geq A$ and $B^U \geq B$. It always holds that $A^U \otimes B^U \geq A \otimes B$ and $A^U \cdot B^U \geq A \cdot B$. Therefore, if we use **PDT** bounds in composition Theorem 1 (resp., Corollary 3), we get as a result – denoted $\overline{\mathbf{PDT}}^U$ and computed as $A^U \cdot B^U$ (resp., $A^U \otimes B^U$) – a corresponding bound for the composite **PDT** – denoted $\overline{\mathbf{PDT}}$ and computed as $A \cdot B$ (resp., $A \otimes B$): $\overline{\mathbf{PDT}}^U \geq \overline{\mathbf{PDT}} \dot{\geq} \mathbf{PDT}$. Then, if we use $\overline{\mathbf{PDT}}^U$ in the formula for the computation of the security level (Claim 8) instead of $\overline{\mathbf{PDT}}$, we get

$$s^U = \mathbf{e}^T \cdot \overline{\mathbf{PDT}}^U \cdot \mathbf{p}_\emptyset \geq \mathbf{e}^T \cdot \overline{\mathbf{PDT}} \cdot \mathbf{p}_\emptyset \geq s.$$

We compute the statistical-only lower bound s^L in a similar manner. One should however keep in mind that $s^L \leq s$ does not hold in general, since Claim 8 and the sequential composition theorem only guarantee an upper bound (in addition to the non-tightness coming from the maskVerif algorithm). Again, the statistical-only lower bound is however useful for estimating the uncertainty on the security level that comes from the Monte-Carlo method: if there is a large gap between s^L and s^U, increasing the number of samples in the Monte-Carlo sampling can result in a better s^U (on the other hand, s^L gives a limit on how much we can hope to reduce s^U by increasing the number of samples).

5.3 Tool

We implemented the computation of the above bounds in the open-source tool STRAPS (Sampled Testing of the RAndom Probing Security). This tool contains a few additional algorithmic optimizations that do not change the results but significantly reduce the execution time (e.g., we exploit the fact that, in some circuits, many wires carry the same value, and we avoid to explicitly compute **PDT**s of large composite gadgets to reduce memory usage). Regarding performance, for the computation of the security of the AES S-box (see Figure 10), almost all of the execution time goes into computing the **PDT** of the ISW multiplication gadgets. Computing the **PDT**s of the other gadgets is much faster as they are smaller, and computing the composition takes a negligible amount of time (less than 1 %). The total running time for the AES S-box is less than 5 s for 1, 2 and 3 shares, 30 s for 4 shares, 3 min for 5 shares, and 33 h for 6 shares on a 24-core computer (dual 2.3 GHz Intel(R) Xeon(R) CPU E5-2670 v3).

[5] And additionally the change of the condition $s_i < N_t$ by $s_{i,\mathcal{O}'\mathcal{I}} < N_t$. The rationale for this condition is that, intuitively, if we have many "worst-case" samples, then we should have a sufficient knowledge of the distribution $\left(P_i(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')\right)_{\mathcal{I}' \subseteq \mathcal{I}}$.

STRAPS presents a few similarities with VRAPS [8]. While STRAPS mainly computes **PDT** bounds and VRAPS computes random probing expandability bounds, both metrics relate to the random probing security of a gadget, and both tools are based on the maskVerif dependency test algorithm. The main differences between these tools are twofold. First, STRAPS uses a mix of Monte-Carlo sampling and full exploration of the sets of probes, whereas VRAPS does only full exploration. Second, STRAPS computes and uses the simulatability set for a given set of internal and output probes, while VRAPS only stores whether the size of the simulatability set exceeds a given threshold. Thanks to this weaker requirement, VRAPS is able to exploit the set exploration algorithm of maskVerif, which accelerates the full exploration of the sets of probes by avoiding an exhaustive enumeration of all subsets [4].

5.4 Experiments & SOTA comparison

In this final section, we illustrate how to use our **PDT** bounding tool and the **PDT** composition theorems in order to bound the security of larger circuits, and to extract useful intuitions about the trade-off between the number of shares and level of noise required to reach a given security level. We also compare our results with previous works by Dziembowski et al. [20] and Belaïd et al. [8,9].

We begin by evaluating the impact of using composition theorems instead of a direct security evaluation. In Section 3.2, we concluded that directly analyzing the security of even a single multiplication gadget in the random probing model tightly is computationally intensive. On Figure 7, we show the security of a slightly more complex $\mathsf{ISW}(\mathsf{x}, \mathsf{SNI\text{-}Ref}(\mathsf{x}^2))$ gadget evaluated as either the composition of four gadgets (a split gadget, a squaring, an SNI refresh and an ISW multiplication), or as a single gadget (we call it integrated evaluation). We can see that when the gadget becomes large ($n = 5$) and for a similar computational complexity, the results for the **PDT** composition are statistically tighter thanks to the lower size of its sub-gadgets. We also observe that, when upper and lower bounds converge, the security level computed from **PDT** composition is close to the one computed by the integrated evaluation, although the latter one is slightly better. We conclude that the **PDT** composition technique can provide useful results in practically relevant contexts where we build gadget compositions for which the integrated evaluation is not satisfying.

Next, we investigate different refreshing strategies when computing the x^3 operation with an ISW multiplication gadget. Namely, we compare the situation with no refreshing which is known to be insecure in the threshold probing model [16], the simple refreshing with linear randomness complexity which does not offer strong composability guarantees, and an SNI refresh gadget from [12]. The results are illustrated in Figure 8. In the first case (with no refreshing), we observe the well-known division by two of the statistical security order (reflected by the slope of the security curves in the asymptotic region where the noise is sufficient and curves become linear): the security level is asymptotically proportional to $p^{\lceil (n-1)/2 \rceil}$. On the other side of the spectrum, the composition with an SNI refresh guarantees a statistical security order of $n-1$. Finally, the most

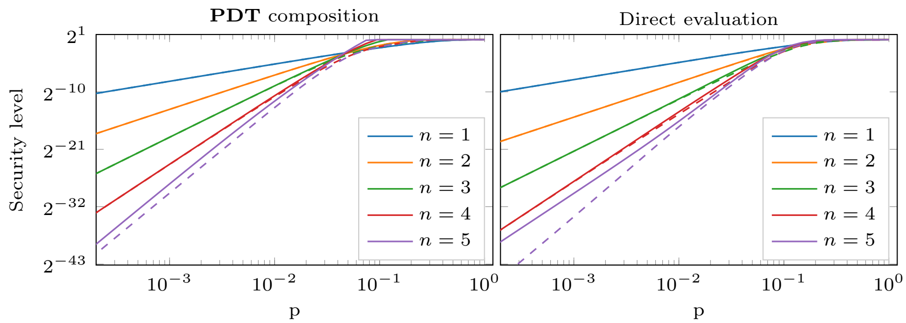


Fig. 7: Security of a cubing gadget $\mathsf{ISW}(\mathsf{x}, \mathsf{SNI\text{-}Ref}(\mathsf{x}^2))$. The left plot comes from **PDT** composition while the right plot is a direct security evaluation of the full circuit as a single gadget. The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 2 \times 10^6$, $N_t = 1000$.

interesting case is the one of the simple refresh gadget, for which we observe a statistical security order reduction for $n \geq 3$, of which the impact may remain small for low noise levels. For instance, we can see that for $p \geq 2 \times 10^{-3}$, the curves for the simple and the SNI refresh gadgets are almost the same, with the security order reduction becoming more and more apparent only for lower values of p. So this analysis provides us with a formal quantitative understanding of a gadget's security level which, for example, suggests that depending on the noise levels, using SNI gadgets may not always be needed.

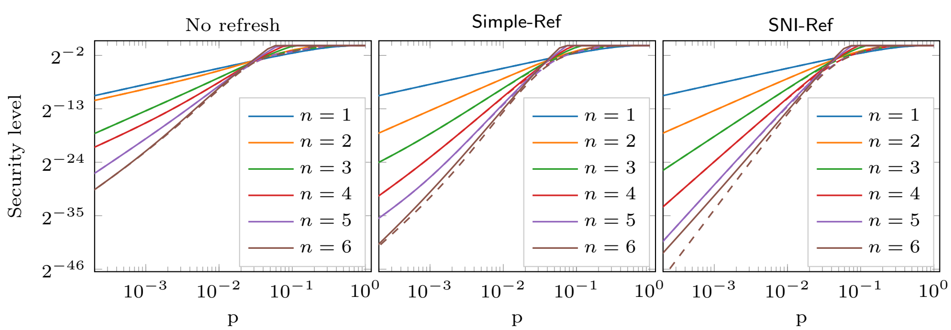


Fig. 8: Security of the cubing $\mathsf{ISW}(\mathsf{x}, \mathsf{Ref}(\mathsf{x}^2))$, where Ref is identity (no refreshing), $\mathsf{Simple\text{-}Ref}$, or $\mathsf{SNI\text{-}Ref}$ gadget. The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 10^8$, $N_t = 100$.

We extend this analysis of a simple gadget to the case of a complete AES S-box in Figure 9. All the previous observations remain valid in this case as well. Furthermore, this figure confirms that our results get close to the ones reported for concrete worst-case attacks in [18]. Namely, already for the (low) number of shares and (practical) levels of noise we consider, we observe a statistical security order of $n-1$ for a practically relevant (AES S-box) circuit.[6]

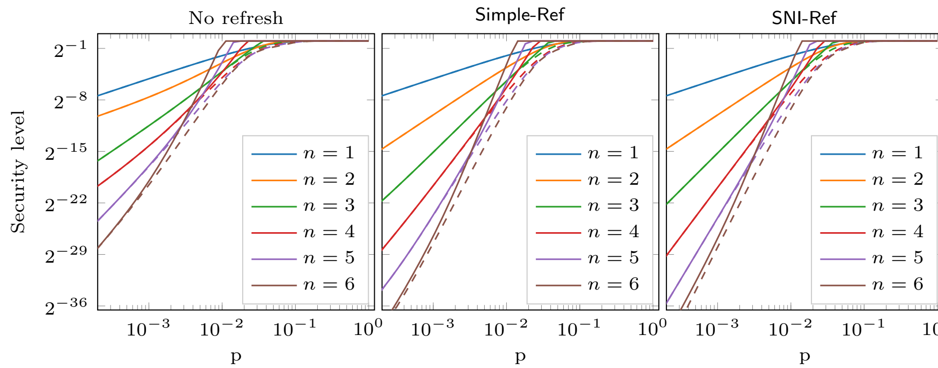


Fig. 9: Security of the non-linear part of an AES S-box in $\mathbb{F}_{256}$, where Ref is either an identity (no refreshing), the Simple-Ref gadget, or the SNI-Ref gadget. The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 10^8$, $N_t = 100$.

Eventually, we compare our bounds with state-of-the-art results for the non-linear part of the AES S-box in Figure 10, in order to highlight that such tight results were not available with existing solutions. Precisely, we compare our results with the works that provide the best bounds in the low-noise region that we consider: the Simple Refreshing (SR) strategy of Dziembowski et al. [20], and the first (RPE1) [8] and second (RPE2) [9] sets of gadgets from the Random Probing Expansion strategy of Belaïd et al. We see that amongst the previous works we consider here, RPE2 with 27 shares achieves the best maximum tolerated leakage probability and statistical security order. Our **PDT**-based analysis of the SNI-refreshed AES S-box with the ISW multiplication achieves a similar security level with only 6 shares. In this last experiment, the number of shares n is an indicator for the circuit size since all schemes have a circuit size in $\mathcal{O}(n^2)$. So we conclude that our results enable a significant improvement of the provable security claims of practical masked circuits in the random probing model.

Acknowledgments. Gaëtan Cassiers and François-Xavier Standaert are resp. Research Fellow and and Senior Associate Researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). Maximilan Orlt is founded by the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG). This work has been funded in part by the ERC project 724725 and by Deutsche

[6] To make the results more easily comparable, one can just assume connect the leakage probability with the mutual information of [18] by just assuming that the mutual information per bit (i.e., when the unit is the field element) equals p.

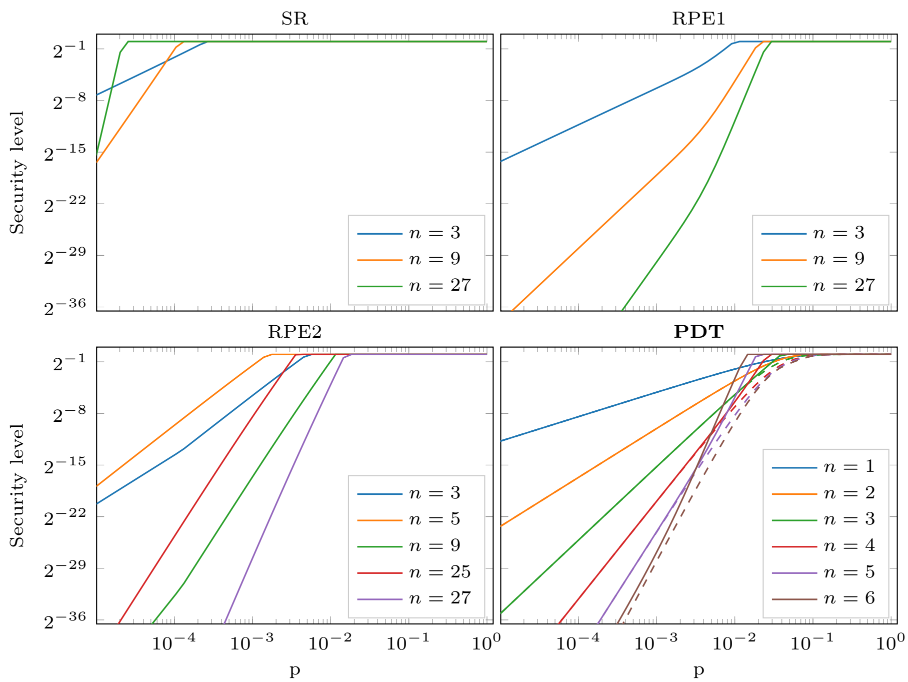


Fig. 10: Security of the non-linear part of an AES S-box in $\mathbb{F}_{256}$, based on the best result of each paper. For the **PDT**, we take use a SNI refresh gadget. All the circuits have a size $\mathcal{O}(n^2)$.

Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1119 - 236615297 (Project S7).

References

1. M. Ajtai. Secure computation with information leaking to an adversary. In *STOC*, pages 715–724. ACM, 2011.
2. P. Ananth, Y. Ishai, and A. Sahai. Private circuits: A modular approach. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455. Springer, 2018.
3. M. Andrychowicz, S. Dziembowski, and S. Faust. Circuit compilers with o(1/\log (n)) leakage rate. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615. Springer, 2016.
4. G. Barthe, S. Belaïd, G. Cassiers, P. Fouque, B. Grégoire, and F. Standaert. maskverif: Automated verification of higher-order masking in presence of physical defaults. In *ESORICS (1)*, volume 11735 of *Lecture Notes in Computer Science*, pages 300–318. Springer, 2019.
5. G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, P. Strub, and R. Zucchini. Strong non-interference and type-directed higher-order masking. In *CCS*, pages 116–129. ACM, 2016.
6. G. Barthe, M. Gourjon, B. Grégoire, M. Orlt, C. Paglialonga, and L. Porth. Masking in fine-grained leakage models: Construction, implementation and verification. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):189–228, 2021.
7. A. Battistello, J. Coron, E. Prouff, and R. Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.
8. S. Belaïd, J. Coron, E. Prouff, M. Rivain, and A. R. Taleb. Random probing security: Verification, composition, expansion and new constructions. In *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368. Springer, 2020.
9. S. Belaïd, M. Rivain, and A. R. Taleb. On the power of expansion: More efficient constructions in the random probing model. *IACR Cryptol. ePrint Arch.*, 2021:434, 2021.
10. R. Bloem, H. Groß, R. Iusupov, B. Könighofer, S. Mangard, and J. Winter. Formal verification of masked hardware implementations in the presence of glitches. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.
11. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
12. G. Cassiers, B. Gregoire, I. Levi, and F. X. Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, pages 1–1, 2020.
13. G. Cassiers and F. Standaert. Towards globally optimized masking: From low randomness to low noise rate or probe isolating multiplications with reduced randomness and security against horizontal attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):162–198, 2019.
14. S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

15. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
16. J. Coron, E. Prouff, M. Rivain, and T. Roche. Higher-order side channel security and mask refreshing. In *FSE*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.
17. A. Duc, S. Dziembowski, and S. Faust. Unifying leakage models: From probing attacks to noisy leakage. *J. Cryptol.*, 32(1):151–177, 2019.
18. A. Duc, S. Faust, and F. Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *J. Cryptol.*, 32(4):1263–1297, 2019.
19. S. Dziembowski, S. Faust, and M. Skorski. Noisy leakage revisited. In *EUROCRYPT (2)*, volume 9057 of *Lecture Notes in Computer Science*, pages 159–188. Springer, 2015.
20. S. Dziembowski, S. Faust, and K. Zebrowski. Simple refreshing in the noisy leakage model. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 315–344. Springer, 2019.
21. S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.
22. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
23. D. Goudarzi, A. Joux, and M. Rivain. How to securely compute with noisy leakage in quasilinear complexity. In *ASIACRYPT (2)*, volume 11273 of *Lecture Notes in Computer Science*, pages 547–574. Springer, 2018.
24. Y. Ishai, A. Sahai, and D. A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
25. D. Knichel, P. Sasdrich, and A. Moradi. SILVER - statistical independence and leakage verification. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 787–816. Springer, 2020.
26. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
27. S. Mangard, T. Popp, and B. M. Gammel. Side-channel leakage of masked CMOS gates. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
28. S. Nikova, V. Rijmen, and M. Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptol.*, 24(2):292–321, 2011.
29. T. Prest, D. Goudarzi, A. Martinelli, and A. Passelègue. Unifying leakage models on a rényi day. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712. Springer, 2019.
30. E. Prouff and M. Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.
31. M. Rivain and E. Prouff. Provably secure higher-order masking of AES. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
32. T. Schneider and A. Moradi. Leakage assessment methodology - extended version. *J. Cryptogr. Eng.*, 6(2):85–99, 2016.
33. F. Scholz. Confidence bounds & intervals for parameters relating to the binomial, negative binomial, poisson and hypergeometric distributions with applications to rare events, 2008.

A Simple masked gadgets

Algorithm 2 Linear gadget for a any linear function $\circ$ over n shares

Input $(x_i)_{i\in[n]}$, $(y_i)_{i\in[n]}$
Output Sharing $(z_i)_{i\in[n]}$ such that $\bigoplus_i z_i = \bigoplus_i x_i \circ \bigoplus_i y_i$
  for $i = 0$ to $n-1$ **do**
    $z_i \leftarrow x_i \circ y_i$;

Algorithm 3 ISW multiplication gadget over n shares.

Input Factors $(x_i)_{i\in[n]}, (y_i)_{i\in[n]}$
Output Sharing $(z_i)_{i\in[n]}$ such that $\bigoplus_i z_i = \bigoplus_i x_i \odot \bigoplus_i y_i$.
  for $i = 0$ to $n-1$ **do**
    for $j = i+1$ to $n-1$ **do**
      $r_{ij} \xleftarrow{\$} \mathbb{F}_q$; $r_{ji} \leftarrow r_{ij}$;
  for $i = 0$ to $n-1$ **do**
    for $j = 0$ to $n-1$, $j \neq i$ **do**
      $p_{ij} \leftarrow (x_i \odot y_j) \oplus r_{ij}$;
  for $i = 0$ to $n-1$ **do**
    $z_i \leftarrow x_i \odot y_i + \bigoplus_i j = 0, j \neq i^{n-1} p_{ij}$;

Algorithm 4 Simple refresh gadget over n shares.

Input $(x_i)_{i\in[n]}$
Output Refreshed sharing $(y_i)_{i\in[n]}$ such that $\bigoplus_i y_i = \bigoplus_i x_i$
  $t_0 \leftarrow x_{n-1}$;
  for $i = 0$ to $n-2$ **do**
    $r_i \xleftarrow{\$} \mathbb{F}_q$;
    $y_i \leftarrow x_i \oplus r_i$;
    $t_{i+1} \leftarrow t_i \oplus r_i$;
  $y_{n-1} \leftarrow t_{n-1}$;

B Proofs

B.1 Proof of Claim 1

Proof. The proof of this claim is divided into two parts. Firstly, we prove that (i) $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ is a subset of $\mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) \;||_{(|\mathcal{I}_0|)}\; \mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0)$. Then, we show that (ii) $\mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) \;||_{(|\mathcal{I}_0|)}\; \mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0)$ is a subset of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$:

(i) We build a simulator for G out of the simulator of G_0 and G_1, as depicted in Figure 5. The input wire set of G is $\mathcal{I} = \mathcal{I}_1||_{(|\mathcal{I}_0|)}\mathcal{I}_0$, where $\mathcal{I}_0$ is the input wire set of G_0 and $\mathcal{I}_1$ is the input wire set of G_1. The same applies to $\mathcal{O} = \mathcal{O}_1||_{(|\mathcal{O}_0|)}\mathcal{O}_0$. Using the definition of $\mathcal{L}_p(\mathsf{G})$, we get

$$\mathcal{L}_p(\mathsf{G}) = \mathcal{L}_p(\mathcal{W}, \mathcal{I}) = \mathcal{L}_p\left((\mathcal{W}_1||_{(|\mathcal{W}_0|)}\mathcal{W}_0), (\mathcal{I}_1||_{(|\mathcal{I}_0|)}\mathcal{I}_0)\right).$$

Since $\mathcal{L}_p(\mathcal{X})$ operates independently on each wire of $\mathcal{X}$, this gives

$$\mathcal{L}_p(\mathsf{G}) = \left(\mathcal{L}_p(\mathcal{W}_1)||_{(|\mathcal{W}_0|)}\mathcal{L}_p(\mathcal{W}_0), \mathcal{L}_p(\mathcal{I}_1)||_{(|\mathcal{I}_0|)}\mathcal{L}_p(\mathcal{I}_0)\right). \tag{7}$$

By definition of $\mathcal{S}$, for all $i \in \{0, 1\}$, there exists a simulator of G_i that simulates $(\mathcal{O}'_i, \mathcal{L}_p(\mathcal{W}_i), \mathcal{L}_p(\mathcal{I}_i))$ with the set of wires $\mathcal{S}^{\mathsf{G}_i}\left(\mathcal{L}_p(\mathsf{G}_i), \mathcal{O}'_i\right)$. Concatenating the inputs and outputs of the two simulators, we can simulate $(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ with the set of wires $\mathcal{S}^{\mathsf{G}_1}\left(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1\right)||_{(|\mathcal{I}_0|)}\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0\right)$. Finally, the simulatability set is a subset of any set that allows to simulate $(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$, which implies claim (i).

(ii) Starting from Equation (7) and for all $i \in \{0, 1\}$, we observe that $\mathcal{L}_p(\mathsf{G}_i) = (\mathcal{L}_p(\mathcal{W}_i), \mathcal{L}_p(\mathcal{I}_i))$ is a subset of $\mathcal{L}_p(\mathsf{G})$, and moreover $\mathcal{O}'_i$ is a subset of $\mathcal{O}'$. Therefore, if a set of wires can simulate $(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$, it can also simulate $(\mathcal{L}_p(\mathsf{G}_i, \mathcal{O}'_i))$, and thus $\mathcal{S}^{\mathsf{G}_i}\left(\mathcal{L}_p(\mathsf{G}_i), \mathcal{O}'_i\right) \subseteq \mathcal{S}^{\mathsf{G}}\left(\mathcal{L}_p(\mathsf{G}), \mathcal{O}'\right)$. This implies claim (ii).

$\square$

C Proofs

C.1 Proof of Claim 2

Proof. Let $\mathcal{I}_i$, $\mathcal{W}_i$, and $\mathcal{O}_i$ be the input wires, internal wires and output gates of G_i. The set of outputs of G is $\mathcal{O} = \mathcal{O}_1$ and the set of inputs of G is $\mathcal{I} = \mathcal{I}_0$. Further, we have

$$\mathcal{L}_p(\mathsf{G}) = \mathcal{L}_p(\mathcal{W}, \mathcal{I}) = \mathcal{L}_p(\mathcal{W}_1||_{(n_2)}\mathcal{I}_1||_{(n_1)}\mathcal{W}_0, \mathcal{I}_0)$$

with $n_1 = |\mathcal{W}_0|$, and $n_2 = |\mathcal{I}_1| + n_1$. Moreover, since $\mathcal{L}_p$ operates independently for each of its inputs wires,

$$\mathcal{L}_p(\mathsf{G}) = \left(\mathcal{L}_p(\mathcal{W}_1)||_{(n_2)}\mathcal{L}_p(\mathcal{I}_1)||_{(n_1)}\mathcal{L}_p(\mathcal{W}_0), \mathcal{L}_p(\mathcal{I}_0)\right).$$

By definition, we can simulate $(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) = \left((\mathcal{L}_p(\mathcal{W}_1), \mathcal{L}_p(\mathcal{I}_1)), \mathcal{O}'\right)$ using the set of wires $\mathcal{S}^{\mathsf{G}_1}\left(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'\right)$. Let us denote $\mathcal{O}'_0 = \mathcal{S}^{\mathsf{G}_1}_c\left(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'\right)$.

For G_0, we can simulate $(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0) = \left((\mathcal{L}_p(\mathcal{W}_0), \mathcal{L}_p(\mathcal{I}_0)), \mathcal{O}'_0\right)$ using the set $\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0\right)$. Finally, combining both simulators, we can simulate $(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ using $\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}_c(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$. $\square$