On the Power of Expansion: More Efficient Constructions in the Random Probing Model

Sonia Belaïd[1], Matthieu Rivain[1], and Abdul Rahman Taleb[1,2]

[1] CryptoExperts, France
[2] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

{sonia.belaid,matthieu.rivain,abdul.taleb}@cryptoexperts.com

Abstract. The random probing model is a leakage model in which each wire of a circuit leaks with a given probability p. This model enjoys practical relevance thanks to a reduction to the noisy leakage model, which is admitted as the right formalization for power and electromagnetic side-channel attacks. In addition, the random probing model is much more convenient than the noisy leakage model to prove the security of masking schemes. In a recent work, Ananth, Ishai, and Sahai (CRYPTO 2018) introduce a nice expansion strategy to construct random probing secure circuits. Their construction tolerates a leakage probability of 2^{-26}, which is the first quantified achievable leakage probability in the random probing model. In a follow-up work, Belaïd, Coron, Prouff, Rivain, and Taleb (CRYPTO 2020) generalize their idea and put forward a complete and practical framework to generate random probing secure circuits. The so-called expanding compiler can bootstrap simple base gadgets as long as they satisfy a new security notion called *random probing expandability* (RPE). They further provide an instantiation of the framework which tolerates a 2^{-8} leakage probability in complexity $\mathcal{O}(\kappa^{7.5})$ where κ denotes the security parameter.

In this paper, we provide an in-depth analysis of the RPE security notion. We exhibit the first upper bounds for the main parameter of a RPE gadget, which is known as the *amplification order*. We further show that the RPE notion can be made tighter and we exhibit strong connections between RPE and the *strong non-interference* (SNI) composition notion. We then introduce the first generic constructions of gadgets achieving RPE for any number of shares and with nearly optimal amplification orders and provide an asymptotic analysis of such constructions. Last but not least, we introduce new concrete constructions of small gadgets achieving maximal amplification orders. This allows us to obtain much more efficient instantiations of the expanding compiler: we obtain a complexity of $\mathcal{O}(\kappa^{3.9})$ for a slightly better leakage probability, as well as $\mathcal{O}(\kappa^{3.2})$ for a slightly lower leakage probability.

Keywords: Random probing model, masking, side-channel security

1 Introduction

Most commonly used cryptographic algorithms are assumed to be secure against *black-box* attacks, when the adversary is limited to the knowledge of some inputs and outputs. However, as revealed in the late nineties [18], their implementation on physical devices can be vulnerable to the more powerful *side-channel attacks.* The latter additionally exploit the physical emanations of the underlying device such as the execution time or the device temperature, power consumption, or electromagnetic radiations during the algorithm execution.

To counteract side-channel attacks which often only require cheap equipment and can be easily mounted in a short time interval, the cryptographic community has searched for efficient countermeasures. Among the different approaches, one of the most widely used is known as *masking.* Simultaneously introduced by Chari, Jutla, Rao and Rohatgi [10], and by Goubin and Patarin [16] in 1999, it happens to be strongly related to techniques usually applied in secure multi-party computation. In a nutshell, the idea is to split each sensitive variable of the implementation into n shares such that $n-1$ of them are generated uniformly at random whereas the last one is computed as a combination of the original value and the random shares. Doing so, one aims to ensure that an adversary cannot recover the secret without knowledge of all the shares. When the shares are combined by bitwise addition, the masking is said to be *Boolean*, and it enjoys simple implementation for linear operations which can be simply applied on each share separately. However, things are trickier for non-linear operations for which it is impossible to compute the result without combining shares.

In order to reason about the security of these countermeasures, the community has introduced a variety of models. Among them, the *probing model* introduced by Ishai, Sahai, and Wagner in 2003 [17] is well suited to analyze the security of masked implementations. Basically, it assumes that an adversary is able to get the exact values of a certain number t of intermediate variables in an implementation. This way, it captures the increasing difficulty of combining noisy leakage to recover secrets. Despite its wide use by the community [20, 13, 11, 8, 12], the probing model raised a number of concerns regarding its relevance in practice. Therefore, in 2013, Prouff and Rivain introduced a general and practical model, known as the *noisy leakage model* [19]. This model well captures the reality of embedded devices by assuming that all the manipulated data leak together with some noise. Unfortunately, proving the security of a masking scheme in this model is rather tedious, which is why Duc, Dziembowski, and Faust provided in 2014 a reduction showing that a scheme secure in the probing model is also secure in the noisy leakage model [14].

This reduction is based on an intermediate leakage model, known as *random probing model*, to which the security in the noisy leakage model tightly reduces. In this model, every wire of a circuit is assumed to leak with some constant leakage probability. Then, a circuit is secure if there is a negligible probability that these leaking wires actually reveal information on the secrets. Compared to the probing model, the random probing model is closer to the noisy leakage model and, in particular, captures *horizontal attacks* which exploit the repeated manipulations of variables throughout the implementation. Classical probing secure schemes are also secure in the random probing model but the tolerated leakage probability (a.k.a. leakage rate) might not be constant which is not satisfactory from a practical viewpoint. Indeed, in practice, the leakage probability translates to some side-channel noise amount which might not be customizable by the implementer.

So far, only a few constructions [1, 3, 2, 9] tolerate a constant leakage probability. The two former ones [1, 3] are based on expander graphs and the tolerated probability is not made explicit. The third construction [2] is based on multiparty computation protocols and an expansion strategy. It reaches a tolerated leakage probability of around 2^{-26} for a complexity of $\mathcal{O}(\kappa^{8.2})$ for some security parameter κ, as computed by the authors of [9]. Finally, the more recent construction [9] relies on masking gadgets and a similar expansion strategy and reaches a tolerated leakage probability of 2^{-8} for a complexity of $\mathcal{O}(\kappa^{7.5})$. While obtaining such quantified tolerated leakage probability is of great practical interest, the obtained complexity is high which makes this construction hardly practical.

Besides their explicit construction, the authors of [9] provide a complete and practical framework to generate random probing secure implementations. Namely, they formalize the *expanding compiler* which produces a random probing secure version of any circuit from three base gadgets (for addition, copy, and multiplication) achieving a *random probing expandability* (RPE) property. The advantage of this approach is that it enables to bootstrap small gadgets (defined for a small number of shares) into a circuit achieving arbitrary security in the random probing model while tolerating a constant and quantified leakage probability. Although the concrete results of [9] in terms of complexity and tolerated leakage probability are promising, the authors left open the analysis of this RPE property and the design of better gadgets in this paradigm.

Our contributions. In this paper, we provide an in-depth analysis of the random probing expandability security notion. We first provide some upper bounds for the *amplification order* of an RPE gadget, which is the crucial parameter in view of a low-complexity instantiation of the expanding compiler. We further show that the RPE notion can be made tighter and we exhibit strong relations between RPE and the *strong non-interference* (SNI) composition notion for probing-secure gadgets.

From these results, we introduce the first generic constructions of gadgets achieving RPE for any number of shares and with nearly optimal amplification orders. These generic gadgets are derived from the widely known Ishai-Sahai-Wagner (ISW) construction. We show that the obtained expanding compiler can approach a quadratic complexity depending on the leakage probability that must be tolerated: the smaller the leakage probability, the closer the complexity to $\mathcal{O}(\kappa^2)$. We further introduce a new multiplication gadget achieving the optimal amplification order, which allows us to improve the convergence to a quadratic complexity.

Finally, we provide new concrete constructions of copy, addition, and multiplication gadgets achieving maximal amplification orders for small numbers of shares. These gadgets yield much more efficient instantiations than all the previous schemes (including the analysed ISW-based constructions). While slightly improving the tolerated leakage probability to $p = 2^{-7.5}$, our 3-share instantiation achieves a complexity of $\mathcal{O}(\kappa^{3.9})$. For a slightly lower leakage probability, our 5-share instantiation drops the complexity to $\mathcal{O}(\kappa^{3.2})$.

We thus achieve a significant step forward in the quest for efficient random probing secure schemes that tolerate a quantified leakage probability. Besides our concrete instantiations, our work introduces several tools (new bounds, relations, and generic gadgets) that shall be instrumental for future constructions.

2 Preliminaries

Along the paper, we shall use similar notations and formalism as [9]. In particular, $\mathbb{K}$ shall denote a finite field. For any $n \in \mathbb{N}$, we shall denote $[n]$ the integer set $[n] = [1, n] \cap \mathbb{Z}$. For any tuple $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n$ and any set $I \subseteq [n]$, we shall denote $\boldsymbol{x}|_I = (x_i)_{i \in I}$.

2.1 Linear Sharing, Circuits, and Gadgets

In the following, the *n-linear decoding* mapping, denoted LinDec, refers to the function $\mathbb{K}^n \to \mathbb{K}$ defined as

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

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

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

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

An *arithmetic circuit* on a field $\mathbb{K}$ is a labeled directed acyclic graph whose edges are *wires* and vertices are *arithmetic gates* processing operations on $\mathbb{K}$. We consider circuits composed of addition gates, $(x_1, x_2) \mapsto x_1 + x_2$, multiplication gates, $(x_1, x_2) \mapsto x_1 \cdot x_2$, and copy gates, $x \mapsto (x, x)$. A *randomized arithmetic circuit* is equipped with an additional random gate which outputs a fresh uniform random value of $\mathbb{K}$.

In the following, we shall call an (*n-share, ℓ-to-m*) *gadget*, a randomized arithmetic circuit that maps an input $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$ to an output $\widehat{\boldsymbol{y}} \in (\mathbb{K}^n)^m$ such that $\boldsymbol{x} = \mathsf{LinDec}(\widehat{\boldsymbol{x}}) \in \mathbb{K}^\ell$ and $\boldsymbol{y} = \mathsf{LinDec}(\widehat{\boldsymbol{y}}) \in \mathbb{K}^m$ satisfy $\boldsymbol{y} = g(\boldsymbol{x})$ for some function g. In this paper, we shall consider gadgets for three types of functions (corresponding to the three types of gates): the addition $g : (x_1, x_2) \mapsto x_1 + x_2$, the multiplication $g : (x_1, x_2) \mapsto x_1 \cdot x_2$ and the copy $g : x \mapsto (x, x)$. We shall generally denote such gadgets G_{add}, G_{mult} and G_{copy} respectively.

2.2 Random Probing Security

Let $p \in [0, 1]$ be some constant leakage probability parameter, a.k.a. the *leakage rate*. In the p-random probing model, an evaluation of a circuit C leaks the value carried by each wire with a probability p (and leaks nothing otherwise), all the wire leakage events being mutually independent.

As in [9], we formally define the random-probing leakage of a circuit from the two following probabilistic algorithms:

- The *leaking-wires sampler* takes as input a randomized arithmetic circuit C and a probability $p \in [0, 1]$, and outputs a set $\mathcal{W}$, denoted as
$$\mathcal{W} \leftarrow \mathsf{LeakingWires}(C, p) \ ,$$
where $\mathcal{W}$ is constructed by including each wire label from the circuit C with probability p to $\mathcal{W}$ (where all the probabilities are mutually independent).
- The *assign-wires sampler* takes as input a randomized arithmetic circuit C, a set of wire labels $\mathcal{W}$ (subset of the wire labels of C), and an input $\boldsymbol{x}$, and it outputs a $|\mathcal{W}|$-tuple $\boldsymbol{w} \in (\mathbb{K} \cup \{\perp\})^{|\mathcal{W}|}$, denoted as
$$\boldsymbol{w} \leftarrow \mathsf{AssignWires}(C, \mathcal{W}, \boldsymbol{x}) \ ,$$
where $\boldsymbol{w}$ corresponds to the assignments of the wires of C with label in $\mathcal{W}$ for an evaluation on input $\boldsymbol{x}$.

Definition 2 (Random Probing Leakage). *The* p-random probing leakage *of a randomized arithmetic circuit* C *on input* $\boldsymbol{x}$ *is the distribution* $\mathcal{L}_p(C, \boldsymbol{x})$ *obtained by composing the leaking-wires and assign-wires samplers as*

$$\mathcal{L}_p(C, \boldsymbol{x}) \overset{id}{=} \mathsf{AssignWires}(C, \mathsf{LeakingWires}(C, p), \boldsymbol{x}) \ .$$

Definition 3 (Random Probing Security). *A randomized arithmetic circuit* C *with* $\ell \cdot n \in \mathbb{N}$ *input gates is* (p, ε)-random probing secure *with respect to encoding* Enc *if there exists a simulator* Sim *such that for every* $\boldsymbol{x} \in \mathbb{K}^\ell$*:*

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

2.3 Expanding Compiler

In [2], Ananth, Ishai and Sahai propose an *expansion* approach to build a random-probing-secure circuit compiler from a secure multiparty protocol. This approach was later revisited by Belaïd, Coron, Prouff, Rivain, and Taleb who formalize the notion of *expanding compiler* [9].

The principle of the expanding compiler is to recursively apply a base compiler, denoted CC, and which simply consists in replacing each gate in the input circuit by the corresponding gadget. More specifically, assume we have three n-share gadgets G_{add}, G_{mult}, G_{copy}, for the addition, the multiplication, and the copy on $\mathbb{K}$. The base compiler CC simply consists in replacing each addition gate in the original gadget by G_{add}, each multiplication gate by G_{mult}, and each copy gate by G_{copy}, and by replacing each wire by n wires carrying a sharing of the original wire. One can derive three new n^2-share gadgets by simply applying CC to each gadget: $G^{(2)}_{\text{add}} = \mathsf{CC}(G_{\text{add}})$, $G^{(2)}_{\text{mult}} = \mathsf{CC}(G_{\text{mult}})$, and $G^{(2)}_{\text{copy}} = \mathsf{CC}(G_{\text{copy}})$. Doing so, we obtain n^2-share gadgets for the addition, multiplication, and copy on $\mathbb{K}$. This process can be iterated an arbitrary number of times, say k, to an input circuit C:

$$C \xrightarrow{\mathsf{CC}} \widehat{C}_1 \xrightarrow{\mathsf{CC}} \cdots \xrightarrow{\mathsf{CC}} \widehat{C}_k \ .$$

The first output circuit $\widehat{C}_1$ is the original circuit in which each gate is replaced by a base gadget G_{add}, G_{mult}, or G_{copy}. The second output circuit $\widehat{C}_2$ is the original circuit C in which each gate is replaced by an n^2-share gadget $G^{(2)}_{\text{add}}$, $G^{(2)}_{\text{mult}}$, or $G^{(2)}_{\text{copy}}$ as defined above. Equivalently, $\widehat{C}_2$ is the circuit $\widehat{C}_1$ in which each gate is replaced by a base gadget. In the end, the output circuit $\widehat{C}_k$ is hence the original circuit C in which each gate has been replaced by a k-expanded gadget and each wire has been replaced by n^k wires carrying an (n^k)-linear sharing of the original wire.

This expanding compiler achieves random probing security if the base gadgets verify a property called *random probing expandability* [9].

2.4 Random Probing Expandability

We recall hereafter the original definition of the random probing expandability (RPE) property for 2-input 1-output gadgets.

Definition 4 (Random Probing Expandability [9]). *Let $f : \mathbb{R} \to \mathbb{R}$. An n-share gadget $G : \mathbb{K}^n \times \mathbb{K}^n \to \mathbb{K}^n$ is (t, f)-*random probing expandable (RPE) *if there exists a deterministic algorithm Sim_1^G and a probabilistic algorithm Sim_2^G such that for every input $(\widehat{x}, \widehat{y}) \in \mathbb{K}^n \times \mathbb{K}^n$, for every set $J \subseteq [n]$ and for every $p \in [0, 1]$, the random experiment*

$$\begin{aligned} &\mathcal{W} \leftarrow \mathsf{LeakingWires}(G, p) \\ &(I_1, I_2, J') \leftarrow \mathsf{Sim}_1^G(\mathcal{W}, J) \\ &out \leftarrow \mathsf{Sim}_2^G(\mathcal{W}, J', \widehat{x}|_{I_1}, \widehat{y}|_{I_2}) \end{aligned}$$

ensures that

1. *the failure events $\mathcal{F}_1 \equiv \big(|I_1| > t\big)$ and $\mathcal{F}_2 \equiv \big(|I_2| > t\big)$ verify*

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \qquad (2)$$

with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),

2. *J' is such that $J' = J$ if $|J| \leq t$ and $J' \subseteq [n]$ with $|J'| = n - 1$ otherwise,*
3. *the output distribution satisfies*

$$out \overset{id}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_{J'}\right) \tag{3}$$

where $\widehat{z} = G(\widehat{x}, \widehat{y})$.

The RPE notion can be simply extended to gadgets with 2 outputs: the Sim_1^G simulator takes two sets $J_1 \subseteq [n]$ and $J_2 \subseteq [n]$ as input and produces two sets J_1' and J_2' satisfying the same property as J' in the above definition (w.r.t. J_1 and J_2). The Sim_2^G simulator must then produce an output including $\widehat{z}_1|_{J_1'}$ and $\widehat{z}_2|_{J_1'}$ where $\widehat{z}_1$ and $\widehat{z}_2$ are the output sharings. The RPE notion can also be simply extended to gadgets with a single input: the Sim_1^G simulator produces a single set I so that the failure event $(|I| > t)$ occurs with probability ε (and the Sim_2^G simulator is then simply given $\widehat{x}|_I$ where $\widehat{x}$ is the single input sharing). We refer the reader to [9] for the formal definitions of these variants. Eventually, the RPE notion can also be extended to gadgets with an arbitrary number ℓ of inputs. The Sim_1^G simulator then produces ℓ sets $I_1, \ldots, I_\ell$ so that the corresponding failures $(|I_1| > t), \ldots, (|I_\ell| > t)$ occur with probability ε and are additionally mutually independent. The Sim_2^G simulator then simply gets use of the shares of each input as designated respectively by the corresponding sets $I_1, \ldots, I_\ell$.

Note that as explained in [9], the requirement of the RPE notion on the mutual independence of the failure events might seem too strong. We can actually use the proposed relaxation referred to as *weak random probing expandability*. Namely, the equalities (Equation (2)) are replaced by inequalities as upper bounds are sufficient in our context. We refer the reader to [9] for the concrete reduction, which does not impact the amplification orders.

2.5 Complexity of the Expanding Compiler

We start by recalling the definition of the *amplification order* of a function and of a gadget.

Definition 5 (Amplification Order).

- *Let $f : \mathbb{R} \to \mathbb{R}$ which satisfies*

$$f(p) = c_d\, p^d + \mathcal{O}(p^{d+\varepsilon})$$

as p tends to 0, for some $c_d > 0$ and $\varepsilon > 0$. Then d is called the amplification order of f.

- *Let $t > 0$ and G a gadget. Let d be the maximal integer such that G achieves (t, f)-RPE for $f : \mathbb{R} \to \mathbb{R}$ of amplification order d. Then d is called the amplification order of G (with respect to t).*

We stress that the amplification order of a gadget G is defined with respect to the RPE threshold t. Namely, different RPE thresholds t are likely to yield different amplification orders d for G (or equivalently d can be thought of as a function of t).

As shown in [9], the complexity of the expanding compiler relates to the (minimum) amplification order of the three gadgets used in the base compiler CC. If the latter achieves (t, f)-RPE with an amplification order d, the expanding compiler achieves $(p, 2^{-\kappa})$-random probing security with a complexity blowup of $\mathcal{O}(\kappa^e)$ for an exponent e satisfying

$$e = \frac{\log N_{\max}}{\log d} \tag{4}$$

with

$$N_{\max} = \max\left(N_{\mathrm{m,m}}\ ,\ \mathsf{eigenvalues}\left(\begin{pmatrix} N_{\mathrm{a,a}} & N_{\mathrm{c,a}} \\ N_{\mathrm{a,c}} & N_{\mathrm{c,c}} \end{pmatrix}\right)\right) \tag{5}$$

where $N_{\mathrm{x,y}}$ denotes the number of gates “x” in a gadget “y”, with “m” meaning multiplication, “a” meaning addition, and “c” meaning copy. As an illustration, the instantiation proposed in [9] satisfies $N_{\max} = 21$ and $d = \frac{3}{2}$ which yields an asymptotic complexity of $\mathcal{O}(\kappa^{7.5})$.

Finally, we recall the notion of maximum *tolerated leakage probability* which corresponds to the maximum value p for which we have $f(p) < p$. This happens to be a necessary and sufficient condition for the expansion strategy to apply with (t, f)-RPE gadgets. The instantiation proposed in [9] tolerates a leakage probability up to $2^{-7.80}$.

3 Bounding the Amplification Order

As recalled above, the amplification order of a gadget is a crucial parameter of its random probing expandability. The higher the amplification order, the lower the asymptotic complexity of the expanding compiler, *ceteris paribus*. A natural question which was left open in [9] is to determine the best amplification order that can be hoped for given the different parameters of a gadget. In this section, we exhibit concrete upper bounds on the amplification order that can be achieved by a gadget depending on its input-output dimensions (ℓ, m), its number of shares n, and its RPE threshold t.

Before giving the bounds let us make a key observation on the amplification order of a gadget. Let G be a 2-to-1 n-share gadget achieving (t, f)-RPE. A subset $\mathcal{W}$ of the wires of G is said to be a *failure set* with respect to the first input (resp. the second input) if there exists a set $J \subseteq [n]$ such that $(I_1, I_2, J') \leftarrow \mathsf{Sim}_1^G(\mathcal{W}, J)$ implies $|I_1| > t$ (resp. $|I_2| > t$), namely if a leaking set $\mathcal{W}$ implies the failure event $\mathcal{F}_1$ (resp. $\mathcal{F}_2$) in the definition of RPE. One can check that G has amplification order $d \leq d_{up}$ if one of the two following events occurs:

1. there exists a failure set $\mathcal{W}$ w.r.t. the first input *or* the second input such that $|\mathcal{W}| = d_{up}$,

2. there exists a failure set $\mathcal{W}$ w.r.t. the first input *and* the second input such that $|\mathcal{W}| = 2d_{up}$.

In the former case, the existence of the failure set implies that the function $f(p)$ has a non-zero coefficient in $p^{d_{up}}$ and hence $d \leq d_{up}$. In the latter case, the existence of the double failure set implies that the function $f^2(p)$ has a non-zero coefficient in $p^{2d_{up}}$ and hence $d \leq d_{up}$. The case of a single-input gadget is simpler: it has amplification order $d \leq d_{up}$ if there exists a failure set $\mathcal{W}$ (w.r.t. its single input) such that $|\mathcal{W}| = d_{up}$.

We start by exhibiting a generic upper bound for the amplification order and then look at the particular case of what we shall call a *standard* multiplication gadget.

3.1 Generic Upper Bound

In the following we will say that a function $g : \mathbb{K}^\ell \to \mathbb{K}^m$ is *complete* if at least one of its m outputs is functionally dependent on the ℓ inputs. Similarly, we say that a gadget G is complete if its underlying function g is complete.

The following lemma gives our generic upper bound on the amplification order.

Lemma 1. *Let $f : \mathbb{R} \to \mathbb{R}$, $n \in \mathbb{N}$ and $\ell, m \in \{1, 2\}$. Let $G : (\mathbb{K}^n)^\ell \to (\mathbb{K}^n)^m$ be an ℓ-to-m n-share complete gadget achieving (t, f)-RPE. Then its amplification order d is upper bounded by*

$$\min((t + 1), (3 - \ell) \cdot (n - t)).$$

Proof. The first part of the bound on the amplification order $d \leq (t + 1)$ is immediate since by probing $t + 1$ shares of any input, the considered set will be a failure set of cardinality $t + 1$. We then consider two cases depending on the number of inputs:

1. *1-input gadgets ($\ell = 1$):* We show that we can exhibit a failure set of size $2(n - t)$. Let us denote the output shares $z_1, \ldots, z_n$ (for two-output gadgets, *i.e.* $m = 2$, $z_1, \ldots, z_n$ can be any of the output sharings). In the evaluation of the (t, f)-RPE property, t shares among the z_i's (corresponding to the set J) must be simulated. Without loss of generality, let $z_1, \ldots, z_t$ be those shares (*i.e.* $J = [t]$). By including both input gates of each of the remaining output shares $z_{t+1}, \ldots, z_n$ in the set $\mathcal{W}$, the distribution to be simulated requires the knowledge of the full input (by completeness of the gadget). The set $\mathcal{W}$ is thus a failure set with $2(n - t)$ elements.
2. *2-input gadgets ($\ell = 2$):* Considering the same failure set as in the above case, the simulation of *out* requires the full two input sharings. Hence $\mathcal{W}$ is a failure set of size $2(n - t)$ with respect to the two inputs, and so the amplification order satisfies $d \leq (n - t)$.

We hence conclude that $d \leq \min((t + 1), 2(n - t))$ for one-input gadgets, and $d \leq \min((t + 1), (n - t))$ for two-input gadgets. □

Corollary 1 (One-input gadget). *The amplification order d of a one-input gadget achieving (t,f)-RPE is upper bounded by*

$$d \leq \frac{2(n+1)}{3} \ .$$

The above corollary directly holds from Lemma 1 for a RPE threshold $t = \frac{2n-1}{3}$ (which balances the two sides of the min).

Corollary 2 (Two-input gadget). *The amplification order d of a two-input gadget achieving (t,f)-RPE is upper bounded by*

$$d \leq \frac{n+1}{2} \ .$$

The above corollary directly holds from Lemma 1 for a RPE threshold $t = \frac{n-1}{2}$ (which balances the two sides of the min).

We deduce from the two above corollaries that for a circuit composed of addition, multiplication and copy gadgets, the amplification order is upper bounded

$$d \leq \min\left(\frac{2(n+1)}{3}, \frac{n+1}{2}\right) = \frac{n+1}{2} \ ,$$

which can only be achieved for an odd number of shares by taking $t = \frac{n-1}{2}$ as RPE threshold.

3.2 Upper Bound for Standard Multiplication Gadgets

The generic bound exhibited above is not tight in the special case of a standard multiplication gadget which computes cross products between the input shares, such as the ISW multiplication gadget [17]. We exhibit hereafter a tighter bound for such gadgets.

Formally, a n-share multiplication gadget G is a *standard multiplication gadget*, if on input $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathbb{K}^n)^2$, G computes the cross products $x_i \cdot y_j$ for $1 \leq i, j \leq n$. Our upper bound on the amplification order for such gadgets is given in the following lemma.

Lemma 2. *Let $f : \mathbb{R} \to \mathbb{R}$ and $n \in \mathbb{N}$. Let G be an n-share standard multiplication gadget achieving (t,f)-RPE. Then its amplification order d is upper bounded by*

$$d \leq \min\left(\frac{t+1}{2}, (n-t)\right) .$$

Proof. The second part of the bound $(n-t)$ holds directly from Lemma 1. We now prove the bound $(t+1)/2$ by exhibiting a failure set of size $t+1$ with t output shares, which will be a failure on both inputs. Let $\{m_{ij}\}_{0 \leq i,j \leq n}$ denote the cross products such that $m_{ij} = x_i \cdot y_j$. Consider a set $\mathcal{W}$ made of $t+1$ such variables $\{m_{ij}\}$ for which the indexes i and j are all distinct. Specifically,

$\mathcal{W} = \{x_{i_1} \cdot y_{j_1}, \ldots, x_{i_{t+1}} \cdot y_{j_{t+1}}\}$ such that $\{i_\ell\}_{1 \leq \ell \leq t+1}$ and $\{j_\ell\}_{1 \leq \ell \leq t+1}$ are both sets of $(t+1)$ distinct indexes. Clearly, such a set is a failure set for both inputs $\boldsymbol{x}$ and $\boldsymbol{y}$ since it requires $t+1$ shares of each of them to be perfectly simulated (even without considering the output shares to be also simulated). We hence have a double failure set of cardinality $t+1$ which implies the $(t+1)/2$ upper bound on the amplification order. $\square$

The above lemma implies that the highest amplification order for standard multiplication gadgets might be achieved for a RPE threshold $t = \frac{2n-1}{3}$ which yields the following maximal upper bound:

$$d \leq \frac{n+1}{3} \ ,$$

which is lower than the generic upper bound for 2-to-1 gadgets exhibited in Corollary 2. This loss suggests that better amplification orders could be achieved for multiplication gadgets that do not compute direct cross products of the input shares. We actually provide new constructions of multiplication gadgets avoiding this loss in Section 5.

4 A Closer Look at Random Probing Expandability

In this section, we give a closer look at the RPE notion. We first show that it naturally splits into two different notions, that we shall call RPE1 and RPE2, and further introduce a tighter variant which will be useful for our purpose. We then study the relations between (tight) RPE and the *Strong Non-Interference* (SNI) notion used for probing security. We exhibit strong connections between (tight) RPE1 and SNI, which will be very useful for our constructive results depicted in Section 5.

4.1 Splitting RPE

From Definition 4, we can define two sub-properties which are jointly equivalent to RPE. In the first one, designated by RPE1, the set J is constrained to satisfy $|J| \leq t$ and $J' = J$ (the simulator does not choose J'). In the second one, designated by RPE2, J' is chosen by the simulator such that $J' \subseteq [n]$ with $|J'| = n-1$ (and J does not matter anymore). For the sake of completeness, these two notions are formally defined in the full version of this paper.

This split is somehow a partition of the RPE notion since we have:

$$G \text{ is } (t,f)\text{-RPE} \iff G \text{ is } (t,f)\text{-RPE1 } \textit{and } G \text{ is } (t,f)\text{-RPE2}$$

for any gadget G. As a result of the above equivalence, we can show that a gadget achieves RPE1 and RPE2 independently in order to obtain RPE for this gadget. Formally, we use the following lemma.

Lemma 3. *An n-share gadget $G : \mathbb{K}^n \times \mathbb{K}^n \to \mathbb{K}^n$ which is (t, f_1)-RPE1 and (t, f_2)-RPE2 is also (t, f)-RPE with $f(p) \geq \max(f_1(p), f_2(p))$ for every $p \in [0, 1]$.*

We can refine the upper bounds introduced in Section 3 with respect to this split. In Lemma 1, the bound $d \leq t + 1$ applies to both RPE1 and RPE2, while the bound $d \leq (3 - \ell) \cdot (n - t)$ only applies to RPE1. Similarly, in Lemma 2, the bound $d \leq (t + 1)/2$ applies to both RPE1 and RPE2, while the bound $d \leq (n - t)$ only applies to RPE1.

4.2 Tightening RPE

We introduce a tighter version of the RPE security property. The so-called *tight random probing expandability* (TRPE) is such that a failure occurs when the simulation requires more than t input shares (as in the original RPE notion) but also whenever this number of shares is greater than the size of the leaking set $\mathcal{W}$. Formally, the failure event $\mathcal{F}_j$ is defined as

$$\mathcal{F}_j \equiv \big(|I_j| > \min(t, |\mathcal{W}|)\big)$$

for every $j \in [\ell]$.

This tighter security property will be instrumental in the following to obtain generic RPE constructions. Similarly to the original RPE property, the TRPE property can be split into two intermediate properties, namely TRPE1 and TRPE2 and Lemma 3 also applies to the case of TRPE. Moreover the upper bounds on the amplification order for RPE in Lemmas 1 and 2 further apply to the amplification order for TRPE (which holds by definition). The formal TRPE, TRPE1, and TRPE2 definitions are given in the full version of this paper for the sake of completeness.

We show hereafter that the TRPE notion is actually equivalent to the RPE notion if and only if the function f is of maximal amplification order $t + 1$.

Lemma 4. *Let $t \in \mathbb{N}$, let $f : \mathbb{R} \to \mathbb{R}$ of amplification order d. Let G be a gadget.*

1. *If G achieves (t, f)-TRPE, then it achieves (t, f')-RPE for some $f' : \mathbb{R} \to \mathbb{R}$ of amplification order $d' \geq d$.*
2. *If G is of amplification order d with respect to t (i.e. d is the max amplification order of a function f for which G is (t, f)-RPE), then for all $f' : \mathbb{R} \to \mathbb{R}$ for which G achieves (t, f')-TRPE, f' is of amplification order $d' \leq d$.*
3. *If $d = t + 1$, then G achieves (t, f)-TRPE if and only if G achieves (t, f)-RPE.*

Proof. The proof for the first two points is easy. In particular, for the first point, if G achieves TRPE with an amplification order of d, then G achieves RPE with amplification order at least d, since a failure in the TRPE setting *i.e.* $|I_j| > \min(t, |\mathcal{W}|)$ does not necessarily imply a failure in the RPE setting *i.e.* $|I_j| > t$, meanwhile if there is no failure for TRPE for a leaking set of wires $\mathcal{W}$, then this implies that $|I_j| \leq \min(t, |\mathcal{W}|) \leq t$ so there is no failure in the RPE setting either.

As for the second point, the proof is similar: if G achieves an amplification of d in the RPE setting, then it achieves an amplification order of at most d in the TRPE setting, since a failure in the RPE setting *i.e.* $|I_j| > t$ immediately implies a failure in the TRPE setting $|I_j| > \min(t, |\mathcal{W}|)$. But also, even if there is no failure for a leaking set of wires $\mathcal{W}$ in the RPE setting we might still have a failure in the TRPE setting for the same set $\mathcal{W}$. This is mainly the case where $\mathcal{W}$ can be simulated with sets of input shares I_j such that $|\mathcal{W}| < |I_j| \leq t$, so we have $|I_j| \leq t$ (*i.e.* no failure for RPE) and $|I_j| > \min(t, |\mathcal{W}|) = |\mathcal{W}|$ (*i.e.* failure on TRPE). This concludes the proof for the second point.

We will now prove the third point. Let $d = t+1$. We will show that for every set $J' \subseteq [n]$ of output shares and every leaking set of wires $\mathcal{W}$, a failure occurs in the TRPE setting if and only if a failure also occurs in the RPE setting. If $|\mathcal{W}| \geq t$, then the two settings are equivalent since $\min(t, |\mathcal{W}|) = t$. We will thus only focus on the case $|\mathcal{W}| < t$. Clearly, a failure in the RPE setting, *i.e.* $|I_j| > t$, implies a failure in the TRPE setting, *i.e.* $|I_j| > \min(t, |\mathcal{W}|)$. Let us now show that the converse is also true.

We assume by contradiction that there exists J' and $\mathcal{W}$ implying a TRPE failure which is not an RPE failure, that is a set I_j satisfying $|\mathcal{W}| < |I_j| \leq t$. We then show that there exists a leaking set $\mathcal{W}'$ of size $|\mathcal{W}'| < t+1$ for which an RPE failure always occurs, which implies an amplification order strictly lower than $t+1$ and hence contradicts the lemma hypothesis. This set $\mathcal{W}'$ is constructed as $\mathcal{W}' = \mathcal{W} \cup I'_j$ for some set $I'_j \subset [n] \setminus I_j$ such that $|I'_j| = t+1-|I_j|$. The simulation of $\mathcal{W}'$ and J' then requires the input shares from $I_j \cup I'_j$. However, we have

$$|I_j \cup I'_j| = |I_j| + |I'_j| = t+1$$

implying an RPE failure, and

$$|\mathcal{W}'| = |\mathcal{W} \cup I'_j| \leq |\mathcal{W}| + |I'_j| = |\mathcal{W}| + t + 1 - |I_j| < |\mathcal{W}| + t + 1 - |\mathcal{W}| = t+1.$$

Thus, we have built a failure set $\mathcal{W}'$ of size strictly less than the amplification order $t+1$, which contradicts the hypothesis and hence concludes the proof. □

The above proof also applies to the case of the split notions, specifically for ((t, f)-RPE1, (t, f)-TRPE1) and for ((t, f)-RPE2, (t, f)-TRPE2).

4.3 Unifying (Tight) RPE and SNI

Strong non-interference (SNI) is a widely used notion to compose probing-secure gadgets [5]. In [9], the authors exhibit a relation between the SNI and the *random probing composability* (RPC) property in their Proposition 1. We go one step further and study the relation between SNI and (T)RPE.

We state hereafter some equivalence results between the (T)RPE1 and SNI notions, up to some constraints on the parameters. Let us first recall the definition of the SNI notion.

Definition 6 (Strong Non-Interference (SNI)). *Let n, ℓ and τ be positive integers. An n-share gadget $G : (\mathbb{K}^n)^\ell \to \mathbb{K}^n$ is τ-SNI if there exists a deterministic algorithm Sim_1^G and a probabilistic algorithm Sim_2^G such that for every set $J \subseteq [n]$ and subset $\mathcal{W}$ of wire labels from G satisfying $|\mathcal{W}| + |J| \leqslant \tau$, the following random experiment with any $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$*

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

yields

$$|I_1| \leqslant |\mathcal{W}|, \dots, |I_\ell| \leqslant |\mathcal{W}| \tag{6}$$

and

$$out \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, \widehat{\boldsymbol{x}})\ ,\ \widehat{\boldsymbol{y}}|_J\big) \tag{7}$$

where $\boldsymbol{I} = (I_1, \dots, I_\ell)$ and $\widehat{\boldsymbol{y}} = G(\widehat{\boldsymbol{x}})$.

We first formally show that (T)RPE1 implies SNI.

Lemma 5. *Let $t \in \mathbb{N}$ and $f : \mathbb{R} \to \mathbb{R}$ of amplification order $t + 1$. Let G be a gadget which achieves (t, f)-TRPE1. Then G is also t-SNI.*

Proof. By definition of TRPE1 and by hypothesis on the amplification order, there exist input sets $I_1, \dots, I_\ell$ which can perfectly simulate any leaking wires set $\mathcal{W}$ such that $|\mathcal{W}| \le t$ and any set of output shares J such that $|J| \le t$, satisfying $|I_1|, \dots, |I_\ell| \le |\mathcal{W}|$. Consequently, there exist input sets $I_1, \dots, I_\ell$ which can perfectly simulate any leaking wires set $\mathcal{W}$ such that $|\mathcal{W}| = t_i \le t$ and any set of output shares J such that $|\mathcal{W}| + |J| \le t$ with $|I_1|, \dots, |I_\ell| \le t_i$. G is thus t-SNI. □

We now show that SNI implies TRPE1 up to some constraints on the parameters t and τ.

Lemma 6. *Let $\tau, \ell \in \mathbb{N}$. Let G be an ℓ-to-1 gadget which achieves τ-SNI. Then G satisfies (t, f)-TRPE1 for some $f : \mathbb{R} \to \mathbb{R}$ with an amplification order of*

$$d \ge \frac{1}{\ell} \min(t + 1, \tau - t + 1)\ .$$

Proof. Since G is τ-SNI, then for any set of leaking wires $\mathcal{W}$ and output shares J such that $|\mathcal{W}| + |J| \le \tau$, the wires indexed by $\mathcal{W}$ and the output shares indexed by J can be perfectly simulated from input shares indexed by $I_1, \dots, I_\ell$ such that $|I_j| \le |\mathcal{W}|$ for every $1 \le j \le \ell$. In the TRPE1 property, the set J of output shares can be any set of size $|J| \le t$ so we can assume $|J| = t$ without loss of generality.

For a leaking set $\mathcal{W}$ of size $|\mathcal{W}| < \min(t+1, \tau-t+1)$ no failure event occurs. Indeed τ-SNI and $|\mathcal{W}| < \tau - t + 1$ implies $|\mathcal{W}| + |J| \le \tau$ and hence the existence of the sets $I_1, \dots, I_\ell$ allowing the simulation with $|I_j| \le |\mathcal{W}|$. And $|\mathcal{W}| < t + 1$ implies $|I_j| \le \min(t, |\mathcal{W}|)$ for every j which implies the absence of failure. Then for a leaking set $\mathcal{W}$ of size $|\mathcal{W}| \geq \min(t+1, \tau - t + 1)$, no condition remains to rule out simulation failures and one could actually get a failure for every input. In the latter case, the amplification order would equal $\frac{1}{\ell}\min(t+1, n-t)$, but in all generality it could be higher (*i.e.* this value is a lower bound). $\square$

An illustrative summary of the relations between RPE1, TRPE1 and SNI is depicted in Figure 1 (d denotes the amplification order of the function f). We hence observe an equivalence between the three notions up to some constraints on the parameters t, d, τ and ℓ.

$$\begin{tikzcd}
\boxed{\tau\text{-SNI}} \arrow[r, bend left, "d \geq \frac{1}{\ell}\min(t+1, \tau-t+1)"] & \boxed{(t,f)\text{-TRPE1}} \arrow[l, bend left, "\tau = t"] \arrow[r, bend left] & \boxed{(t,f)\text{-RPE1}} \arrow[l, bend left, dashed, "\text{iff } d = t+1"]
\end{tikzcd}$$

Fig. 1: Summary of relations between the different notions.

Relation and separation between (T)RPE2 and SNI. For a given n-share gadget G, the (T)RPE2 notion exclusively focuses on the simulation of a set of leaking intermediate variables together with a chosen set of $(n-1)$ output shares. If G is τ-SNI for $\tau < n-1$, then nothing can be claimed on the simulation of the latter sets. But if G is $(n-1)$-SNI, then any set of $(n-1)$ output shares can be perfectly simulated without the knowledge of any input share. Concretely, it implies that G is (t,f)-(T)RPE2 of amplification order at least 1 as a chosen output set of $(n-1)$ shares alone can be perfectly simulated without any additional knowledge on the input shares. Namely, we have

$$(n-1)\text{-SNI} \Rightarrow (t,f)\text{-(T)RPE2 of amplification order at least 1.}$$

Nevertheless, there is no relation from τ-SNI to (t,f)-(T)RPE2 for amplification orders strictly greater than 1 as (T)RPE2 would then consider leaking sets of size larger than or equal to n (for n-share gadgets, $\tau < n$). On the other side, there is no direct implication either from (t,f)-(T)RPE2 to τ-SNI since the former property does not consider all possible output sets of size $(n-1)$, but only a chosen one.

5 Generic Constructions

To the best of our knowledge, the only RPE gadgets in the literature are the ones designed in [9] which are restricted to a small number of shares, specifically $n \in \{2, 3\}$. A natural open question is the definition of RPE gadgets with good amplification orders, typically achieving or approaching the upper bounds exhibited in Section 3, for *any* number of shares n. In this section, we exhibit copy, addition, and multiplication gadgets derived from the widely known Ishai-Sahai-Wagner (ISW) construction [17]. Based on the results demonstrated in Section 4, we are able to show that these gadgets achieve RPE for any number of shares n with amplification orders close to the upper bounds (up to a small constant factor). We further provide an asymptotic analysis of the expanding compiler using these gadgets as well as a new multiplication gadget reaching the optimal amplification order hence improving the convergence to a better asymptotic complexity.

5.1 Generic Copy and Addition Gadgets

As intuitively proposed in [9] for small gadgets, copy and addition gadgets can be naturally derived from a refresh gadget. Such a gadget takes one sharing as input and outputs a new refreshed sharing of the same value. We formally introduce these natural constructions hereafter and show that their RPE security can be reduced to that of the underlying refresh gadget.

Generic Copy Gadget. Algorithm 1 displays the generic construction for the copy gadget from a refresh gadget. It simply consists in refreshing the input sharing twice to obtain two fresh copies.

Algorithm 1: Copy gadget G_{copy}

```
Input  : $(a_1, \ldots, a_n)$ input sharing
Output: $(e_1, \ldots, e_n)$, $(f_1, \ldots, f_n)$ fresh copies of $(a_1, \ldots, a_n)$
$(e_1, \ldots, e_n) \leftarrow G_{\text{refresh}}(a_1, \ldots, a_n)$;
$(f_1, \ldots, f_n) \leftarrow G_{\text{refresh}}(a_1, \ldots, a_n)$;
```

We have the following lemma (see the proof in the full version of this paper).

Lemma 7. *Let $G_{refresh}$ be an n-share (t, f)-TRPE refresh gadget of amplification order d. Then, the copy gadget G_{copy} displayed in Algorithm 1 is (t, f')-TRPE also of amplification order d.*

As a consequence of this result, a TRPE refresh gadget directly yields a TRPE copy gadget achieving the same amplification order. Both gadgets can then reach the upper bound for 1-input gadgets whenever $t + 1 = 2(n - t)$ implying an amplification order $d = \frac{2(n+1)}{3}$.

Generic Addition Gadget. Algorithm 2 displays the generic construction for the addition gadget from a refresh gadget. It simply consists in refreshing both input sharings before adding them.

```
Algorithm 2: Addition Gadget $G_{\mathrm{add}}$
    Input  : $(a_1, \ldots, a_n), (b_1, \ldots, b_n)$ input sharings
    Output: $(c_1, \ldots, c_n)$ sharing of $a + b$
    $(e_1, \ldots, e_n) \leftarrow G_{\mathrm{refresh}}(a_1, \ldots, a_n)$;
    $(f_1, \ldots, f_n) \leftarrow G_{\mathrm{refresh}}(b_1, \ldots, b_n)$;
    $(c_1, \ldots, c_n) \leftarrow (e_1 + f_1, \ldots, e_n + f_n)$;
```

We have the following lemma (see the proof in the full version of this paper).

Lemma 8. *Let $G_{refresh}$ be an n-share refresh gadget and let G_{add} be the corresponding addition gadget displayed in Algorithm 2. Then if $G_{refresh}$ is (t, f)-RPE (resp. (t, f)-TRPE) of amplification order d, then G_{add} is (t, f')-RPE (resp. (t, f')-TRPE) for some f' of amplification order $d' \geq \lfloor \frac{d}{2} \rfloor$.*

The above lemma shows that a (T)RPE refresh gadget of amplification order d directly yields a (T)RPE addition gadget of amplification order at least $\lfloor \frac{d}{2} \rfloor$. If the refresh gadget achieves the optimal $d = \frac{2(n+1)}{3}$, then the generic addition gadget has an amplification order at least $\lfloor \frac{n}{3} \rfloor$ which is not far from the upper bound for two-input gadgets of $\frac{n+1}{2}$.

We stress that the results of Lemma 7 and Lemma 8 are general and apply for any refresh gadget satisfying the (T)RPE property. In the rest of the section, we shall focus on a particular refresh gadget, namely the ISW-based refresh gadget. We show that this gadget achieves (T)RPE from which we obtain (T)RPE copy and addition gadgets for any number of shares n and with amplification orders close to the upper bound (up to a small constant factor).

5.2 ISW-based Copy and Addition Gadgets

As a basis of further constructions, we focus our analysis on the most deployed refresh gadget, which is based on the ISW construction [17].

ISW Refresh Gadget. This refresh can be seen as an ISW multiplication between the input sharing and the n-tuple $(1, 0, \ldots, 0)$. This is formally depicted in Algorithm 3.

Algorithm 3: ISW Refresh

```
Input  : $(a_1, \ldots, a_n)$ input sharing, $\{r_{ij}\}_{1 \le i < j \le n}$ random values
Output: $(c_1, \ldots, c_n)$ such that $c_1 + \cdots + c_n = a_1 + \cdots + a_n$
for $i \leftarrow 1$ to $n$ do
    $c_i \leftarrow a_i$;
end
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $i - 1$ do
        $c_i \leftarrow c_i + r_{ji}$;
    end
    for $j \leftarrow i + 1$ to $n$ do
        $c_i \leftarrow c_i + r_{ij}$;
    end
end
return $(c_1, \ldots, c_n)$;
```

We demonstrate through Lemma 9 that the ISW refresh gadget satisfies TRPE with an amplification order close to the optimal one. The proof is given in the full version of this paper.

Lemma 9. *Let $n \in \mathbb{N}$. For every $t \le n - 2$, the n-share ISW refresh gadget is (t, f_1)-TRPE1 and (t, f_2)-TRPE2 for some functions $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ of amplification orders d_1, d_2 which satisfy:*

- *$d_1 = \min(t + 1, n - t)$ for f_1,*
- *$d_2 = t + 1$ for f_2.*

Corollary 3 then directly follows from Lemma 3 applied to TRPE and Lemma 9.

Corollary 3. *Let $n \in \mathbb{N}$. For every $t \le n - 2$, the n-share ISW refresh gadget is (t, f)-TRPE of amplification order*

$$d = \min(t + 1, n - t).$$

According to Lemma 1, the upper bound on the amplification order of 1-input gadgets is $d \le \min(t + 1, 2(n - t))$ which gives $d \le \frac{2n+2}{3}$ for $t = \frac{2n-1}{3}$. In contrast, the ISW refresh gadget reaches $d = \lfloor \frac{n+1}{2} \rfloor$ by taking $t = \lceil \frac{n-1}{2} \rceil$. While applying this result to the generic constructions of addition and copy gadgets introduced above, we obtain:

- a copy gadget of amplification order $d_c = \lfloor \frac{n+1}{2} \rfloor$ (Lemma 7),
- an addition gadget of amplification order at least $d_a = \lfloor \frac{n+1}{4} \rfloor$ (Lemma 8).

In the following, we demonstrate a tighter result than Lemma 8 for the ISW-based addition gadget (namely which does not imply the loss of a factor 2).

ISW-based Copy Gadget. The copy gadget G_{copy} that uses the n-share ISW refresh gadget as a building block in Algorithm 1 achieves the same amplification order as the ISW refresh for the TRPE setting, *i.e.* $d = \min(t+1, n-t)$. This is a direct implication from Lemma 7. Then, from Lemma 4, we have that ISW-based G_{copy} also achieves (t, f')-RPE with amplification order $d' \geq d$. We can actually prove that ISW-based G_{copy} achieves (t, f')-RPE with amplification order d' exactly equal to the amplification order in the TRPE setting, *i.e.* $d' = d = \min(t+1, n-t)$. This is stated in the following lemma which proof is given in the full version of this paper.

Lemma 10. *Let G_{copy} be the n-share copy gadget displayed in Algorithm 1 and instantiated with the ISW refresh gadget. Then for every $t \leq n-2$, G_{copy} achieves (t, f)-RPE with amplification order $d = \min(t+1, n-t)$.*

ISW-based Addition Gadget. The addition gadget G_{add} that uses the n-share ISW refresh gadget as a building block in Algorithm 2 achieves the same amplification order as the ISW refresh gadget, which is tighter than the bound from Lemma 8. This is stated in the following Lemma, which follows from Lemma 9, and from the fact that ISW refresh is $(n-1)$-SNI. The proof is given in the full version of this paper.

Lemma 11. *Let G_{add} be the n-share addition gadget displayed in Algorithm 2 and instantiated with the ISW refresh gadget. Then for every $t \leq n-2$, G_{add} achieves (t, f_1)-TRPE1 and (t, f_2)-TRPE2 for some functions $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ of amplification orders d_1, d_2 which satisfy:*

- $d_1 = \min(t+1, n-t)$,
- $d_2 = t+1$.

Corollary 4 then directly follows from Lemma 11 by applying Lemma 3 (TRPE1 $\cap$ TRPE2 $\Rightarrow$ TRPE) and Lemma 4 (TRPE $\Rightarrow$ RPE).

Corollary 4. *Let $n \in \mathbb{N}$. For every $t \leq n-2$, the n-share gadget G_{add} displayed in Algorithm 2 and instantiated with the ISW refresh gadget is (t, f)-RPE of amplification order $d = \min(t+1, n-t)$.*

5.3 ISW Multiplication Gadget

In contrast to the copy and addition gadgets that are built from generic schemes with a refresh gadget as a building block, the multiplication gadget can be directly defined as the standard ISW multiplication, which is recalled in Algorithm 4.

Algorithm 4: ISW Multiplication

```
Input  : $(a_1, \dots, a_n)$,$(b_1, \dots, b_n)$ input sharings, $\{r_{ij}\}_{1 \le i < j \le n}$ random
         values
Output: $(c_1, \dots, c_n)$ sharing of $a \cdot b$
for $i \leftarrow 1$ to $n$ do
    $c_i \leftarrow a_i \cdot b_i$;
end
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow i + 1$ to $n$ do
        $c_i \leftarrow c_i + r_{ij}$;
        $r_{ji} \leftarrow (a_i \cdot b_j + r_{ij}) + a_j \cdot b_i$;
        $c_j \leftarrow c_j + r_{ji}$;
    end
end
return $(c_1, \dots, c_n)$;
```

We have the following lemma (see the proof in the full version of this paper).

Lemma 12. *Let $n \in \mathbb{N}$. For every $t \le n - 2$, the n-share ISW multiplication gadget displayed in Algorithm 4 is (t, f_1)-RPE1 and (t, f_2)-RPE2 for some functions $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ of amplification orders d_1, d_2 which satisfy:*

- $d_1 = \dfrac{\min(t+1, n-t)}{2}$,
- $d_2 = \dfrac{t+1}{2}$.

Corollary 5 then directly follows from Lemma 12 by applying Lemma 3 (RPE1 $\cap$ RPE2 $\Rightarrow$ RPE).

Corollary 5. *Let $n \in \mathbb{N}$. For every $t \le n - 2$, the n-share ISW multiplication gadget displayed in Algorithm 4 is (t, f)-RPE of amplification order*

$$d = \frac{\min(t+1, n-t)}{2} \; .$$

According to Lemma 2, the upper bound on the amplification order of a standard multiplication gadget (*i.e.* which starts with the cross-products of the input shares) is $d \le \min((t+1)/2, (n-t))$ which gives $d \le (n+1)/3$ for $t = (2n-1)/3$. In contrast, the ISW multiplication gadget reaches $d = \lfloor \frac{n+1}{4} \rfloor$ by taking $t = \lceil \frac{n-1}{2} \rceil$.

5.4 Application to the Expanding Compiler

As recalled in Section 2.5, instantiating the expanding compiler with three RPE base gadgets gives a $(p, 2^{-\kappa})$-random probing secure compiler (*i.e.* achieving κ

bits of security against a leakage probability p) with a complexity blowup of $\mathcal{O}(\kappa^e)$ for an exponent e satisfying

$$e = \frac{\log N_{\max}}{\log d}$$

where $N_{\max}$ satisfies (5) and where d is the minimum amplification order of the three base gadgets.

We can instantiate the expanding compiler using the above ISW-based gadgets. Specifically, we use the ISW multiplication for the multiplication gadget G_{mult}, and the generic constructions of addition and copy gadgets based on the ISW refresh. From Lemmas 10, 11, and 12, the maximum amplification order achievable by the compiler is the minimum of the three gadgets, which is the order of the ISW multiplication gadget:

$$d = \frac{\min(t+1, n-t)}{2} \ .$$

Hence, for a given number of shares n, the maximum amplification order achievable is

$$d_{\max} = \left\lfloor \frac{n+1}{4} \right\rfloor$$

which is obtained for $t = \lceil \frac{n-1}{2} \rceil$. On the other hand, the value of $N_{\max}$ can be characterized in terms of the number of shares n from the ISW algorithm. Recall from Section 2.5 that

$$N_{\max} = \max\left(N_{\text{m,m}} \ , \ \textsf{eigenvalues}\left(\begin{pmatrix} N_{\text{a,a}} & N_{\text{c,a}} \\ N_{\text{a,c}} & N_{\text{c,c}} \end{pmatrix} \right) \right) .$$

In the case of the ISW-based gadgets, we have $N_{\text{m,m}} = n^2$ and

$$\begin{pmatrix} N_{\text{a,a}} & N_{\text{c,a}} \\ N_{\text{a,c}} & N_{\text{c,c}} \end{pmatrix} = \begin{pmatrix} n(2n-1) & 2n(n-1) \\ n(n-1) & n^2 \end{pmatrix} .$$

The eigenvalues of the above matrix are $\lambda_1 = n$ and $\lambda_2 = 3n^2 - 2n$, implying $N_{\max} = 3n^2 - 2n$. Thus, the expanding compiler instantiated by our ISW-based gadgets has a complexity blowup $\mathcal{O}(\kappa^e)$ with exponent

$$e = \frac{\log(3n^2 - 2n)}{\log(\lfloor (n+1)/4 \rfloor)} .$$

Figure 2 (blue curve) shows the evolution of the value of this exponent with respect to the number of shares n (where we assume an odd n). The value of e clearly decreases as the number of shares grows, and this decrease is faster for a small number of shares ($5 \leq n \leq 10$). The exponent value reaches $e \approx 4$ for a number of shares around 25 and then slowly converges towards $e = 2$ as n grows. This is to be compared with the $\mathcal{O}(\kappa^{7.5})$ complexity achieved by the instantiation from [2, 9].

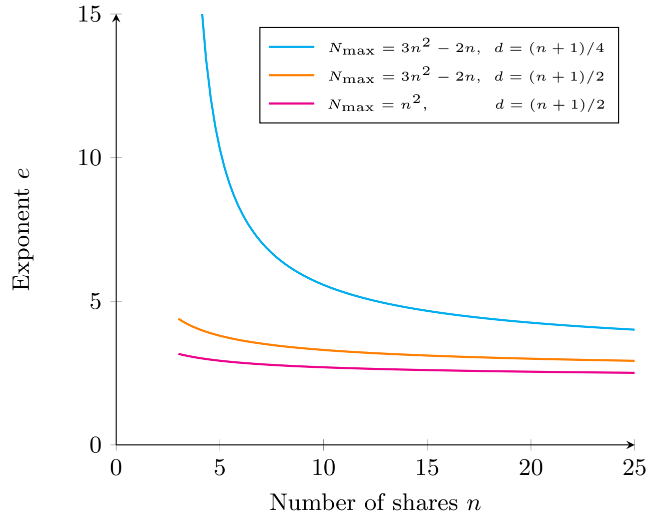

Fig. 2: Evolution of the complexity exponent $e = \log(N_{\max})/\log(d)$ with respect to the number of shares n. The blue curve matches the instantiation with the ISW-based gadgets; the orange curve assumes the optimal amplification order (*i.e.* an improvement of the multiplication gadget); the pink curve assumes a better complexity for addition and copy gadgets (so that $N_{\max}$ matches $N_{\mathrm{m,m}} = n^2$).

Towards a Better Complexity. Choosing gadgets which attain the upper bound $\min(t+1, n-t)$ on the amplification order from Lemma 1 allows the compiler to have the maximum amplification order $d = (n+1)/2$ and thus have the lowest complexity blowup. Our ISW-based copy and addition gadgets achieve this bound while the ISW multiplication gadget is limited to $(n+1)/4$ (Lemma 12). To reach the optimal amplification order, one would need a different multiplication gadget and in particular a multiplication gadget which does not perform a direct product of shares (because of the bound from Lemma 2). We introduce such a multiplication gadget hereafter (see Section 5.5). Specifically, our new multiplication gadget achieves the upper bound on the amplification order $\min(t+1, n-t)$ by avoiding a direct product of shares using a prior refresh on the input sharings. The orange curve in Figure 2 shows the evolution of the value of the exponent when instantiating the expanding compiler with our previous addition and copy gadgets and this new multiplication gadget. For such an instantiation, the complexity exponent still slowly converges towards $e = 2$ but, as we can see from Figure 2, the exponent value is much better for small values of n. For example, we obtain $e \approx 3$ for $n = 20$.

Another possible direction for improvement would be to lower the complexity of the addition and copy gadgets, which is mainly dominated by the refreshing. Assume that we can design a (T)RPE refresh gadget in sub-quadratic complexity, *e.g.* as the refresh gadgets proposed in [20, 7, 15], then the eigenvalues of the matrix in (5) would also be sub-quadratic and the value of $N_{\max}$ from equation (5) would drop to $N_{\mathrm{m,m}} = n^2$ (if the multiplication gadget still requires n^2 multiplication gates). The pink curve in Figure 2 depicts the evolution of the exponent value under this assumption. We still have a slow convergence towards $e = 2$ but the exponent value is yet better for small values of n. For example, a complexity blowup of $\mathcal{O}(\kappa^{2.5})$ is obtained with 20 shares. We leave the task of finding such a sub-quadratic (T)RPE refresh gadget as an open question for further research.

The above analysis shows that the expanding compiler can theoretically approach a quadratic complexity at the cost of increasing the number of shares in the base gadgets. The downside of it is that the tolerated leakage probability is likely to decrease as the number of shares grow. For instance, the ISW construction is known to only tolerate a leakage probability $p = \mathcal{O}(1/n)$ [14]. The number of shares hence offers multiple trade-offs between the tolerated probability and the asymptotic complexity of the compiler. Starting from a target leakage probability p, one could determine the highest number of shares admissible from a generic construction (such as the ISW-based instantiation exhibited above) and thus deduce the best complexity exponent achievable. In Section 6, we exhibit concrete trade-offs that can be reached for small values of n.

5.5 Multiplication Gadget with Maximal Amplification Order

Constructing a multiplication gadget which achieves the upper bound on the amplification order from Lemma 1 is tricky. First, as a standard multiplication gadget (*i.e.* which computes the cross products of the input shares), the ISW multiplication cannot achieve the maximal amplification order (see Lemma 2). In order to reach the upper bound for two-input gadgets (see Corollary 2), we need a non-standard multiplication gadget, *i.e.* which does not perform a direct product between the input shares. As an additional observation, the addition, copy, and random gates are *virtually free* in a multiplication gadget since they do not impact the final complexity of the expanding compiler (see Section 2.5). This suggests that we can be greedy in terms of randomness to reach the maximal amplification order.

In the following, we will describe the construction of a new multiplication gadget which achieves the maximum amplification order $\min(t + 1, n - t)$. We first describe our standard n-share multiplication gadget and then explain how we avoid the initial cross products of shares. First, the gadget constructs the matrix of the cross product of input shares:

$$M = \begin{pmatrix} a_1 \cdot b_1 & a_1 \cdot b_2 & \cdots & a_1 \cdot b_n \\ a_2 \cdot b_1 & a_2 \cdot b_2 & \cdots & a_2 \cdot b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n \cdot b_1 & a_n \cdot b_2 & \cdots & a_n \cdot b_n \end{pmatrix}$$

Then, it picks n^2 random values which define the following matrix:

$$R = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,n} \end{pmatrix}$$

It then performs an element-wise addition between the matrices M and R:

$$P = M + R = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} \end{pmatrix}$$

At this point, the gadget randomizes each product of input shares from the matrix M with a single random value from R. In order to generate the correct output, the gadget adds all the columns of P into a single column V of n elements, and adds all the columns of the transpose matrix R^T into a single column X of n elements:

$$V = \begin{pmatrix} p_{1,1} + \cdots + p_{1,n} \\ p_{2,1} + \cdots + p_{2,n} \\ \vdots \\ p_{n,1} + \cdots + p_{n,n} \end{pmatrix}, \qquad X = \begin{pmatrix} r_{1,1} + \cdots + r_{n,1} \\ r_{1,2} + \cdots + r_{n,2} \\ \vdots \\ r_{1,n} + \cdots + r_{n,n} \end{pmatrix}$$

The n-share output is finally defined as $(c_1, \ldots, c_n) = V + X$.

In order to further increase the maximum amplification order attainable by the gadget, we need to avoid performing a direct product of shares (because of the bound proved in Lemma 2). For this, we add a pre-processing phase to the gadget using a refresh gadget G_{refresh}. Specifically, we refresh the input $(b_1, \ldots, b_n)$ each time it is used. In other terms, each row of the matrix M uses a fresh copy of $(b_1, \ldots, b_n)$ produced using the considered refresh gadget. This amounts to performing n independent refreshes of the input $(b_1, \ldots, b_n)$. The matrix M is thus defined as

$$M = \begin{pmatrix} a_1 \cdot b_1^{(1)} & a_1 \cdot b_2^{(1)} & \cdots & a_1 \cdot b_n^{(1)} \\ a_2 \cdot b_1^{(2)} & a_2 \cdot b_2^{(2)} & \cdots & a_2 \cdot b_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ a_n \cdot b_1^{(n)} & a_n \cdot b_2^{(n)} & \cdots & a_n \cdot b_n^{(n)} \end{pmatrix}$$

where $(b_1^{(j)}, \ldots, b_n^{(j)})$, $j \in [n]$, are the n independent refreshings of the input $(b_1, \ldots, b_n)$.

With this refreshing scheme, we avoid using the same share more than once for one of the two input sharings. As a consequence, the double failure set of size $t+1$ which is the reason behind the bound $(t+1)/2$ in Lemma 2, becomes a simple failure set (*i.e.* provoking a failure on a single input sharing). In addition, the computational overhead of these additional n refreshes is negligible compared to the joint contribution of the copy and addition gadgets to the complexity of the expanding compiler.

For the sake of completeness, we present the full algorithm for this multiplication gadget in Algorithm 5.

Algorithm 5: Our multiplication gadget

```
Input  : $(a_1, \dots, a_n)$,$(b_1, \dots, b_n)$ input sharings, $\{r_{ij}\}_{1 \leq i \leq n, 1 \leq j \leq n}$
         random values, refresh gadget $G_{\text{refresh}}$
Output: $(c_1, \dots, c_n)$ sharing of $a \cdot b$
for $i \leftarrow 1$ to $n$ do
    $(b_1^{(i)}, \dots, b_n^{(i)}) \leftarrow G_{\text{refresh}}(b_1, \dots, b_n)$;
end
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $n$ do
        $p_{i,j} \leftarrow a_i \times b_j^{(i)} + r_{i,j}$;
    end
end
$(v_1, \dots, v_n) \leftarrow (0, \dots, 0)$;
$(x_1, \dots, x_n) \leftarrow (0, \dots, 0)$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $n$ do
        $v_i \leftarrow v_i + p_{i,j}$;
        $x_i \leftarrow x_i + r_{i,j}$;
    end
end
for $i \leftarrow 1$ to $n$ do
    $c_i \leftarrow v_i + x_i$;
end
return $(c_1, \dots, c_n)$;
```

In the following lemma, we show that if the refresh gadget G_{refresh} achieves the TRPE1 property with the amplification order at least $d = \min(t+1, n-t)$ for any t, then the multiplication gadget depicted in Algorithm 5 achieves TRPE with the maximum amplification orders. The proof is given in the full version of this paper.

Lemma 13. *Let $t \leq n-1$. Let $G_{refresh}$ be a (t, f')-TRPE1 refresh gadget for some function $f' : \mathbb{R} \to \mathbb{R}$, and G_{mult} the n-share multiplication gadget from Algorithm 5. If f' is of amplification order $d' \geq d = \min(t+1, n-t)$, then G_{mult} achieves (t, f)-TRPE for some function $f : \mathbb{R} \to \mathbb{R}$ of amplification order $d = \min(t+1, n-t)$.*

Corollary 6 then directly follows from Lemma 13 by applying Lemma 4 (TRPE $\Rightarrow$ RPE).

Corollary 6. *Let $t \leq n-1$. Let $G_{refresh}$ be a (t, f')-TRPE1 refresh gadget for some function $f' : \mathbb{R} \to \mathbb{R}$, and G_{mult} the n-share multiplication gadget from Algorithm 5. If f' is of amplification order $d' \geq d = \min(t+1, n-t)$, then G_{mult} achieves (t, f)-RPE for some function $f : \mathbb{R} \to \mathbb{R}$ of amplification order $d = \min(t+1, n-t)$.*

6 Efficient Small Gadgets

This section displays our new constructions of small gadgets for copy, addition, and multiplication operations with a low number of shares. As explained in [9], we cannot achieve RPE security with relevant amplification orders for gadgets of less than 3 shares. Then, as explained in Section 3.1, the highest amplification orders can only be achieved for gadgets with an odd number of shares. We therefore omit 4-share gadgets and display our best trade-offs in terms of RPE security and complexity for 3-share and 5-share gadgets. Each one of these gadgets is experimentally verified using the `VRAPS` verification tool from [9].

Addition and Copy Gadgets. For the construction of small 3-share and 5-share addition and copy gadgets, we use the generic constructions depicted in Algorithms 1 and 2 (in Section 5) which naturally use a refresh gadget as a building block. We hence start by looking for refresh gadgets that have a good complexity in terms of gates count, and achieve the upper bound on the amplification order for the specific case of 3-share and 5-share constructions (but not necessarily for a higher number of shares).

Multiplication gadget. For the construction of small 3-share and 5-share multiplication gadgets, we use the generic construction depicted in Algorithm 5 from Section 5.5 which, to the best of our knowledge, is the only multiplication gadget which achieves the maximum amplification order for any number of shares, and specifically for 3-share and 5-share constructions. As for the refresh gadget G_{refresh} which is used to perform n refreshes on the second input, we use the same scheme as for the construction of small addition and copy gadgets (and which shall satisfy the necessary condition on G_{refresh} from Corollary 6).

While the multiplication gadget from Section 5.5 achieves the desired amplification order, we add another pre-processing phase to the gadget in order to further improve the tolerated leakage probability. In addition to the n refreshes performed on the second input b (see Algorithm 5), we add another single refresh of the input $(a_1, \ldots, a_n)$ before computing the cross-products, using the same refresh gadget G_{refresh}. Refreshing the input $(a_1, \ldots, a_n)$ before usage experimentally shows a further increase in the maximum tolerated leakage probability, by adding more randomness to the input shares before computing the cross-product matrix M in Algorithm 5. And since the refresh gadget G_{refresh} achieves the maximum amplification order, the amplification order achieved by G_{mult} is not affected by adding another refresh to the first input a.

The above construction achieves the maximum amplification order for 3-share ($d = 2$) and 5-share ($d = 3$) gadgets based on natural refresh gadgets detailed hereafter.

6.1 3-share Gadgets

We start with the construction of 3-share gadgets for our three base operations.

Copy and Addition Gadgets. We build our copy and addition gadgets from the instantiation of the generic constructions of Section 5 (Algorithms 1 and 2) with 3 shares. However, we do not use the ISW refresh gadget but the following more efficient construction with only two random values (instead of three):

$$\begin{aligned} G_{\text{refresh}} : c_1 &\leftarrow r_1 + a_1 \\ c_2 &\leftarrow r_2 + a_2 \\ c_3 &\leftarrow (r_1 + r_2) + a_3. \end{aligned}$$

This refresh is sufficient to reach the upper bounds on the amplification orders (from Lemma 1). From this basis, we obtain the following 3-share addition gadget with four random values:

$$\begin{aligned} G_{\text{add}} : c_1 &\leftarrow (r_1 + a_1) + (r_3 + b_1) \\ c_2 &\leftarrow (r_2 + a_2) + (r_4 + b_2) \\ c_3 &\leftarrow \big((r_1 + r_2) + a_3\big) + \big((r_3 + r_4) + b_3\big) \end{aligned}$$

and the following 3-share copy gadget with also four random values:

$$\begin{aligned} G_{\text{copy}} : c_1 &\leftarrow r_1 + a_1; & d_1 &\leftarrow r_3 + a_1 \\ c_2 &\leftarrow r_2 + a_2; & d_2 &\leftarrow r_4 + a_2 \\ c_3 &\leftarrow (r_1 + r_2) + a_3; & d_3 &\leftarrow (r_3 + r_4) + a_3. \end{aligned}$$

Multiplication Gadget. The following construction is a 3-share instantiation of the multiplication gadget described in Section 5.5. For the input refreshing, we use the 3-share refresh gadget described above with two uniformly random values. The construction achieves the bound on the amplification order from Lemma 1 with 17 random values:

$$\begin{aligned} G_{\text{mult}} : i_{1,1} &\leftarrow r_1 + b_1; & i_{1,2} &\leftarrow r_2 + b_2; & i_{1,3} &\leftarrow (r_1 + r_2) + b_3 \\ i_{2,1} &\leftarrow r_3 + b_1; & i_{2,2} &\leftarrow r_4 + b_2; & i_{2,3} &\leftarrow (r_3 + r_4) + b_3 \\ i_{3,1} &\leftarrow r_5 + b_1; & i_{3,2} &\leftarrow r_6 + b_2; & i_{3,3} &\leftarrow (r_5 + r_6) + b_3 \\ a'_1 &\leftarrow r_7 + a_1; & a'_2 &\leftarrow r_8 + a_2; & a'_3 &\leftarrow (r_7 + r_8) + a_3 \end{aligned}$$

$$\begin{aligned} c_1 &\leftarrow (a'_1 \cdot i_{1,1} + r_{1,1}) + (a'_1 \cdot i_{1,2} + r_{1,2}) + (a'_1 \cdot i_{1,3} + r_{1,3}) + (r_{1,1} + r_{2,1} + r_{3,1}) \\ c_2 &\leftarrow (a'_2 \cdot i_{2,1} + r_{2,1}) + (a'_2 \cdot i_{2,2} + r_{2,2}) + (a'_2 \cdot i_{2,3} + r_{2,3}) + (r_{1,2} + r_{2,2} + r_{3,2}) \\ c_3 &\leftarrow (a'_3 \cdot i_{3,1} + r_{3,1}) + (a'_3 \cdot i_{3,2} + r_{3,2}) + (a'_3 \cdot i_{3,3} + r_{3,3}) + (r_{1,3} + r_{2,3} + r_{3,3}). \end{aligned}$$

Results. Table 1 displays the results for the above gadgets obtained through the `VRAPS` tool. The second column gives the complexity, where N_a, N_c, N_m, N_r stand for the number of addition gates, copy gates, multiplication gates and random gates respectively. The third column provides the amplification order of the gadget. And the last column gives the maximum tolerated leakage probability. The last row gives the global complexity, amplification order, and maximum tolerated leakage probability for the expanding compiler using these three gadgets from the results provided in [9].

Table 1: Results for the 3-share gadgets for $(t = 1, f)$-RPE, achieving the bound on the amplification order.

Gadget	**Complexity** (N_a, N_c, N_m, N_r)	**Amplification order**	$\log_2$ **of maximum tolerated proba**
G_{refresh}	$(4, 2, 0, 2)$	2	−5.14
G_{add}	$(11, 4, 0, 4)$	2	−4.75
G_{copy}	$(8, 7, 0, 4)$	2	−7.50
G_{mult}	$(40, 29, 9, 17)$	2	−7.41
Compiler	$\mathcal{O}(\lvert C\rvert \cdot \kappa^{\mathbf{3.9}})$	**2**	**−7.50**

6.2 5-share Gadgets

We now present our 5-share gadgets for our three base operations, which reach the optimal amplification order from Lemma 1.

Copy and Addition Gadgets. As for the 3-share case, we use the generic constructions from Section 5. Instead of using the ISW refresh gadget which would require 10 uniformly random values for a 5-share construction, we use the *circular refresh gadget* described in [4, 6] (a.k.a. *block refresh gadget*):

$$\begin{aligned} G_{\text{refresh}} : c_1 &\leftarrow (r_1 + r_2) + a_1 \\ c_2 &\leftarrow (r_2 + r_3) + a_2 \\ c_3 &\leftarrow (r_3 + r_4) + a_3 \\ c_4 &\leftarrow (r_4 + r_5) + a_4 \\ c_5 &\leftarrow (r_5 + r_1) + a_5. \end{aligned}$$

This gadget only uses n randoms for an n-share construction, and while it does not achieve enough security in the generic case (unless the refresh block is iterated on the input a certain number of times [4, 6]), it proves to be more than enough to achieve the necessary amplification order for our 5-share constructions. We use a variant of the original version (also suggested in [4]): we choose to sum the random values first (thus obtaining a sharing of 0) before adding them to the input shares. The idea is to avoid using the input shares in any of the intermediate variables, so that input shares only appear in the input variables $\{a_i\}_{1\leq i\leq n}$ and the final output variables $\{c_i\}_{1\leq i\leq n}$. Intuitively, this trick allows to have less failure tuples in the gadget because there are less variables that could leak information about the input. This is validated experimentally where we obtain better results in terms of amplification order and tolerated leakage probability for small gadgets.

From this circular refresh, we obtain an addition gadget and a copy gadget that both reach the upper bound on the amplification order while making use of ten random values. The description of those 5-share gadgets is given in the full version of the paper.

Multiplication Gadget. We use the 5-share instantiation of the multiplication gadget described in Section 5.5. For the input refreshing, we use the 5-share circular refresh gadget described above. The gadget advantageously achieves the optimal amplification order (given by Lemma 1) with 55 random values. The description of this 5-share multiplication gadget is given in the full version of the paper.

Results. Table 2 gives the results for the above gadgets obtained through the `VRAPS` tool.

Table 2: Results for the 5-share gadgets for $(t = 2, f)$-RPE, achieving the bound on the amplification order.

Gadget	**Complexity**	**Amplification order**	**$\log_2$ of maximum tolerated proba**
G_{refresh}	$(10, 5, 0, 5)$	3	-4.83
G_{add}	$(25, 10, 0, 10)$	3	$[-6.43, -3.79]$
G_{copy}	$(20, 15, 0, 10)$	3	$[-6.43, -5.78]$
G_{mult}	$(130, 95, 25, 55)$	3	$[-12.00, -6.03]$
Compiler	$\mathcal{O}(\lvert C\rvert \cdot \kappa^{\mathbf{3.23}})$	**3**	$[\mathbf{-12.00}, \mathbf{-6.03}]$

From Tables 1 and 2, we observe that the asymptotic complexity is better for the instantiation based on 5-share gadgets as they provide a better amplification order with limited overhead. While this result can seem to be counterintuitive, it actually comes from the fact that each gadget will be expended less in the second scenario. We stress that we could only obtain an interval $[2^{-12}, 2^{-6}]$ for the tolerated leakage probability because it was computationally too expensive to obtain a tighter interval from the VRAPS tool, but this could probably be improved in the future. Meanwhile, we can consider that our best complexity $\mathcal{O}(|C| \cdot \kappa^{3.2})$ comes at the price of a lower tolerated leakage probability of 2^{-12} (5-share gadget) compared to the $\mathcal{O}(|C| \cdot \kappa^{3.9})$ complexity and $2^{-7.5}$ tolerated leakage probability obtained for our 3-share instantiation.

In comparison, the previous instantiation of the expanding compiler [9] could only achieve a complexity of $\mathcal{O}(|C| \cdot \kappa^{7.5})$ for maximum tolerated probabilities of 2^{-8}, and the instantiation of the expanding approach with a multi-party computation protocol [2], could only achieve a complexity of $\mathcal{O}(|C| \cdot \kappa^{8.2})$ for maximum tolerated probabilities of 2^{-26}.

Acknowledgments. This work is partly supported by the French FUI-AAP25 VeriSiCC project.

References

1. Miklós Ajtai. Secure computation with information leaking to an adversary. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 715–724. ACM Press, June 2011.
2. Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 427–455. Springer, Heidelberg, August 2018.
3. Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 586–615. Springer, Heidelberg, May 2016.
4. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Improved parallel mask refreshing algorithms: generic solutions with parametrized non-interference and automated optimizations. *Journal of Cryptographic Engineering*, 10(1):17–26, April 2020.
5. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 116–129. ACM Press, October 2016.
6. Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 535–566. Springer, Heidelberg, April / May 2017.
7. Alberto Battistello, Jean-Sebastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme.

Cryptology ePrint Archive, Report 2016/540, 2016. http://eprint.iacr.org/2016/540.

8. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 616–648. Springer, Heidelberg, May 2016.
9. Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Abdul Rahman Taleb. Random probing security: Verification, composition, expansion and new constructions. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 339–368. Springer, Heidelberg, August 2020.
10. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Heidelberg, August 1999.
11. Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 441–458. Springer, Heidelberg, May 2014.
12. Jean-Sébastien Coron, Aurélien Greuet, and Rina Zeitoun. Side-channel masking with pseudo-random generator. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 342–375. Springer, Heidelberg, May 2020.
13. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 410–424. Springer, Heidelberg, March 2014.
14. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, Heidelberg, May 2014.
15. Stefan Dziembowski, Sebastian Faust, and Karol Zebrowski. Simple refreshing in the noisy leakage model. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 315–344. Springer, Heidelberg, December 2019.
16. Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, Heidelberg, August 1999.
17. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Heidelberg, August 2003.
18. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996.
19. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, Heidelberg, May 2013.
20. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 413–427. Springer, Heidelberg, August 2010.