Random Probing Expansion: Quasi Linear Gadgets & Dynamic Compilers

Sonia Belaïd[1], Matthieu Rivain[1], Abdul Rahman Taleb[1,2], and Damien Vergnaud[2,3]

[1] CryptoExperts, France
[2] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[3] Institut Universitaire de France, France

{sonia.belaid,matthieu.rivain,abdul.taleb}@cryptoexperts.com
[4] damien.vergnaud@lip6.fr

Abstract. The masking countermeasure is widely used to protect cryptographic implementations against side-channel attacks. While many masking schemes are shown to be secure in the widely deployed probing model, the latter raised a number of concerns regarding its relevance in practice. Offering the adversary the knowledge of a fixed number of intermediate variables, it does not capture the so-called horizontal attacks which exploit the repeated manipulation of sensitive variables. Therefore, recent works have focused on the *random probing model* in which each computed variable leaks with some given probability p. This model benefits from fitting better the reality of the embedded devices. In particular, Belaïd, Coron, Prouff, Rivain, and Taleb (CRYPTO 2020) introduced a framework to generate random probing circuits. Their compiler somehow extends base gadgets as soon as they satisfy a notion called *random probing expandability* (RPE). A subsequent work from Belaïd, Rivain, and Taleb (EUROCRYPT 2021) went a step forward with tighter properties and improved complexities. In particular, their construction reaches a complexity of $\mathcal{O}(\kappa^{3.9})$, for a κ-bit security, while tolerating a leakage probability of $p = 2^{-7.5}$.

In this paper, we generalize the random probing expansion approach by considering a dynamic choice of the base gadgets at each step in the expansion. This approach makes it possible to use gadgets with high number of shares –which enjoy better asymptotic complexity in the expansion framework– while still tolerating the best leakage rate usually obtained for small gadgets. We investigate strategies for the choice of the sequence of compilers and show that it can reduce the complexity of an AES implementation by a factor 10. We also significantly improve the asymptotic complexity of the expanding compiler by exhibiting new asymptotic gadget constructions. Specifically, we introduce RPE gadgets for linear operations featuring a quasi-linear complexity as well as an RPE multiplication gadget with linear number of multiplications. These new gadgets drop the complexity of the expanding compiler from quadratic to quasi-linear.

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

1 Introduction

Implementations of cryptographic algorithms may be vulnerable to the powerful *side-channel attacks*. The latter exploit the power consumption, the electromagnetic radiations or the temperature variations of the underlying device which may carry information on the manipulated data. Entire secrets can be recovered within a short time interval using cheap equipment.

Among the several approaches investigated by the community to counteract side-channel attacks, *masking* is one of the most deployed in practice. Simultaneously introduced by Chari, Jutla, Rao, and Rohatgi [11] and by Goubin and Patarin [15] in 1999, it consists in splitting the sensitive variables into n random shares, among which any combination of $n-1$ shares does not reveal any secret information. When the shares are combined by bitwise addition, the masking is said to be *Boolean*. In this setting, the linear operations can be very easily implemented by applying on each share individually. Nevertheless, non-linear operations require additional randomness to ensure that any set of less than n intermediate variables is still independent from the original secret.

To reason on the security of masked implementations, the community has introduced so-called *leakage models*. They aim to define the capabilities of the attacker to formally counteract the subsequent side-channel attacks. Among them, the *probing model* introduced in 2003 by Ishai, Sahai, and Wagner [16] is probably the most widely used. In a nutshell, it assumes that an adversary is able to get the exact values of up to a certain number of intermediate variables. The idea is to capture the difficulty of learning information from the combination of noisy variables. Despite its wide use by the community [19, 18, 12, 6, 13], the probing model raised a number of concerns regarding its relevance in practice [4, **?**]. It actually fails to capture the huge amount of information resulting from the leakage of all manipulated data. As an example, it typically ignores the repeated manipulation of identical values which would average the noise and remove uncertainty on secret variables (see horizontal attacks [4]). Another model, the *noisy leakage model* introduced by Prouff and Rivain and inspired from [11], offers an opposite trade-off. Although it captures well the reality of embedded devices by assuming that all the data leaks with some noise, it is not convenient to build security proofs. To get the best from both worlds, Duc, Dziembowski, and Faust proved in 2014 that a scheme secure in the probing model is also secure in the noisy leakage model [14]. Nevertheless, the reduction is not very tight since the security level decreases as the size of the circuit increases (*i.e.* a secure circuit C in the probing model is also secure in the noisy model but loses at least a factor $|C|$, where $|C|$ is the number of operations in the circuit).

The reduction relies on an intermediate leakage model, referred to as *random probing model*. The latter benefits from a tight reduction with the noisy leakage model which becomes independent of the size of the circuit. In a nutshell, it assumes that every wire in the circuit leaks with some constant leakage probability. This leakage probability somehow represents the amount of side-channel noise in practice. A masked circuit is secure in the random probing model whenever its random probing leakage can be simulated without knowledge of the underlying

secret data with a negligible simulation failure. In addition to the attacks already captured by the probing model, the random probing model further encompasses the powerful *horizontal attacks* which exploit the repeated manipulations of variables in an implementation.

To the best of our knowledge, five constructions tolerate a constant leakage probability so far [1, 3, 2, 8, 9]. The two former ones use expander graphs and do not make their tolerated probability explicit. In the third construction, Ananth, Ishai, and Sahai develop an expansion strategy on top of multi-party computation protocols. According to the authors of [8], their construction tolerates a leakage probability of around 2^{-26} for a complexity of $\mathcal{O}(\kappa^{8.2})$ with respect to the security parameter κ. Finally, the two more recent constructions [8, 9] follow an expansion strategy on top of masking gadgets achieving the so-called *random probing expandability* (RPE) notion. In a nutshell, every gate in the original circuit is replaced by a corresponding gadget for some chosen number of shares. The operation is repeated until the desired security level is achieved. The improved gadgets of [9] make it possible to tolerate of leakage probability of $2^{-7.5}$ for a complexity of $\mathcal{O}(\kappa^{3.9})$.

Our Contributions. In this paper, we push the random probing expansion strategy one step further by analyzing a dynamic choice of the base gadgets. While the expanding compiler considered in [8, 9] consists in applying a compiler CC composed of base RPE gadgets a given number of times, say k, to the input circuit: $\widehat{C} = \mathsf{CC}^{(k)}(C)$, we consider a dynamic approach in which a new compiler is selected at each step of the expansion from a family of base compilers $\{\mathsf{CC}_i\}_i$. This approach is motivated by the generic gadget constructions introduced in [9] which achieve the RPE property for any number of shares n. While the asymptotic complexity of the expanding compiler decreases with n, the tolerated leakage probability p also gets smaller with n, which makes those constructions only practical for small values of n. We show that using our dynamic approach we can get the best of both worlds: our dynamic expanding compiler enjoys the best tolerated probability as well as the best asymptotic complexity from the underlying family of RPE compilers $\{\mathsf{CC}_i\}_i$. We further illustrate how this approach can reduce the complexity of a random probing secure AES implementation by a factor 10 using a dynamic choice of the gadgets from [9].

This first contribution further motivates the design of asymptotic RPE gadgets achieving better complexity. While the asymptotic constructions introduced in [9] achieve a quadratic complexity, we introduce new constructions achieving quasi-linear complexity. We obtain this result by showing that the quasi-linear refresh gadget from Battistello, Coron, Prouff, and Zeitoun [5] achieves a *strong random probing expandability* (SRPE) which makes it a good building block for linear RPE gadgets (addition, copy, multiplication by constant). We thus solve a first issue left open in [9]. With such linear gadgets, the complexity bottleneck of the expanding compiler becomes the number of multiplications in the multiplication gadget, which is quadratic in known RPE constructions. We then provide a new generic construction of RPE multiplication gadget featuring a linear number of multiplications. We obtain this construction by tweaking

the probing-secure multiplication gadget from Belaïd, Benhamouda, Passelègue, Prouff, Thillard, and Vergnaud [7]. As in the original construction, our RPE gadget imposes some constraint on the underlying finite field. We demonstrate that for any number of shares there exist a (possibly big) field on which our construction can be instantiated and we provide some concrete instantiations for some (small) number of shares.

Using our new asymptotic gadget constructions with the dynamic expansion approach we obtain random probing security for a leakage probability of $2^{-7.5}$ with asymptotic complexity of $\mathcal{O}(\kappa^2)$. Moreover, assuming that the constraint on the finite field from our multiplication gadget is satisfied, we can make this asymptotic complexity arbitrary close to $\mathcal{O}(\kappa)$ which is optimal. In practice, this means that securing circuits defined on large field against random probing leakage can be achieved at a sub-quadratic nearly-linear complexity.

2 Preliminaries

Along the paper, we shall use similar notations and formalism as [8]. 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 $\boldsymbol{x}|_I = (x_i)_{i \in I}$. Any two probability distributions D_1 and D_2 are said ε-*close*, denoted $D_1 \approx_\varepsilon D_2$, if their statistical distance is upper bounded by ε, that is

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

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

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 gates from some base $\mathbb{B} = \{g : \mathbb{K}^\ell \to \mathbb{K}^m\}$, *e.g.*, 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.

Definition 2 (Circuit Compiler from [8]). *A* circuit compiler *is a triplet of algorithms* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows:*

- CC *(circuit compilation) is a deterministic algorithm that takes as input an arithmetic circuit C and outputs a randomized arithmetic circuit $\widehat{C}$,*
- Enc *(input encoding) is a probabilistic algorithm that maps an input $\boldsymbol{x} \in \mathbb{K}^\ell$ to an encoded input $\widehat{\boldsymbol{x}} \in \mathbb{K}^{\ell'}$,*
- Dec *(output decoding) is a deterministic algorithm that maps an encoded output $\widehat{\boldsymbol{y}} \in \mathbb{K}^{m'}$ to a plain output $\boldsymbol{y} \in \mathbb{K}^m$,*

which satisfy the following properties:

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

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

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

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 [8], 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 3 (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*

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

Definition 4 (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 $\boldsymbol{x} \in \mathbb{K}^\ell$:*

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

2.3 Random Probing Expansion

In [2], Ananth, Ishai and Sahai proposed an *expansion* approach to build a random-probing-secure circuit compiler from a secure multi-party protocol. This approach was later revisited by Belaïd, Coron, Prouff, Rivain, and Taleb who formalize the notion of *expanding compiler* [8].

The principle of the expanding compiler is to recursively apply a base compiler, denoted CC and which simply consists in replacing each gate of $\mathbb{B}$ in the input circuit by the corresponding gadget. Assume we have n-share gadgets G_g for each gate g in $\mathbb{B}$. The base compiler CC simply consists in replacing each gate g in these gadgets by G_g and by replacing each wire by n wires carrying a sharing of the value. We thus obtain n^2-share gadgets by simply applying CC to each gadget: $G_g^{(2)} = \mathsf{CC}(G_g)$. 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 g is replaced by a base gadget G_g. The second output circuit $\widehat{C}_2$ is the original circuit C in which each gate is replaced by an n^2-share gadget $G_g^{(2)}$. 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.

The expanding compiler achieves random probing security if the base gadgets verify a property called *random probing expandability* [8]. We recall hereafter the original definition of the random probing expandability (RPE) property for 2-input 1-output gadgets.

Definition 5 (Random Probing Expandability [8]). *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*

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

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*

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

   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*

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

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 [8] for the formal definitions of these variants.

Note that as explained in [8], 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*. We refer the reader to [8] for the concrete reduction, which does not impact the amplification orders.

The authors of [9] eventually introduced a tighter version the RPE security property, namely the tight random probing expandability (TRPE). In this setting, a failure occurs also when the simulation requires a number of input shares which is higher than the size of the leaking set. Similarly to RPE, it can be split into two intermediate properties, namely TRPE1 and TRPE2. In the first one, the set J is constrained to satisfy $|J| \leq t$ and $J' = J$. In the second one, J' is chosen by the simulator such that $J' \subseteq [n]$ and $|J'| = n - 1$.

2.4 Complexity of the Expanding Compiler

For the RPE compiler CC, we consider β n-share gadgets G_g for the β gates $g \in \mathbb{B}$ and the n-share G_{random} gadget which just generates n independent random values as an n-sharing of the original random value. Then to each gadget a complexity vector is associated $N_G = (N_{g_1}, \ldots, N_{g_\beta}, N_r)^\mathsf{T}$ where N_{g_i} stands for the number of gates g_i and N_r for the number of random gates in the gadget G. Then a $(\beta+1) \times (\beta+1)$ square compiler complexity matrix M_CC is defined as

$$M_\mathsf{CC} = \big(N_{g_1} \mid \cdots \mid N_{g_\beta} \mid N_{G_{\text{random}}}\big) \quad \text{with} \quad N_{G_{\text{random}}} = (0, \ldots, 0, n)^\mathsf{T} \ ,$$

where the definition $N_{G_{\text{random}}}$ holds from the fact that the standard circuit compiler replaces each random gate by n random gates.

Given a circuit C with complexity $N_C = |C|$, compiling it with the base gadgets gives a circuit $\hat{C}$ of complexity $N_{\hat{C}} = M_\mathsf{CC} \cdot N_C = |\hat{C}|$. It follows that the kth power of the matrix M gives the gate counts for the level-k gadgets as:

$$M_\mathsf{CC}^k = \underbrace{M_\mathsf{CC} \cdots M_\mathsf{CC}}_{k \text{ times}} = \big(N_{g_1}^{(k)} \mid \cdots \mid N_{g_\beta}^{(k)} \mid N_{G_{\text{random}}}^{(k)}\big) \quad \text{with} \quad N_{G_{\text{random}}}^{(k)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ n^k \end{pmatrix}$$

where $N_{g_i}^{(k)}$ are the gate-count vectors for the level-k gadgets $G_{g_i}^{(k)}$. Let us denote the eigen decomposition of M_CC as $M_\mathsf{CC} = Q \cdot \Lambda \cdot Q^{-1}$, we get

$$M_\mathsf{CC}^k = Q \cdot \Lambda^k \cdot Q^{-1} \quad \text{with} \quad \Lambda^k = \begin{pmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_{\beta+1}^k \end{pmatrix}$$

where λ_i are the eigenvalues of M_CC. We then obtain an asymptotic complexity of

$$|\hat{C}| = \mathcal{O}\big(|C| \cdot \sum_{i=1}^{\beta+1} |\lambda_i|^k\big) = \mathcal{O}\big(|C| \cdot \max(|\lambda_1|, \ldots, |\lambda_{\beta+1}|)^k\big)$$

for a compiled circuit $\hat{C} = CC^{(k)}(C)$.

The complexity of the expanding compiler can be further expressed in terms of the target random probing security level κ. This complexity is related to the notion of *amplification order* that we recall hereafter.

Definition 6 (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 [8], the complexity of the expanding compiler relates to the (minimum) amplification order of the gadgets composing the base compiler CC. If the latter achieve (t, f)-RPE with an amplification order d, the expanding compiler achieves $(p, 2^{-\kappa})$-random probing security with an expansion level k such that $f^{(k)}(p) \leq 2^{-\kappa}$, which yields a complexity blowup of

$$|\hat{C}| = \mathcal{O}\big(|C| \cdot \kappa^e\big) \quad \text{with} \quad e = \frac{\log N_{\max}}{\log d} \tag{4}$$

where

$$N_{\max} = \max \ \mathsf{eigenvalues}(M_{\mathsf{CC}}) \tag{5}$$

Let us slightly explicit the complexity with the 3-gate base $\mathbb{B} = \{\text{add, mult, copy}\}$ as used in [8, 9]. Considering that multiplication gates are solely used in the multiplication gadget ($N_{G_{\text{add}},m} = N_{G_{\text{copy}},m} = 0$) which is the case in the constructions of [8, 9], it can be checked that (up to some permutation) the eigenvalues satisfy

$$(\lambda_1, \lambda_2) = \mathsf{eigenvalues}(M_{ac}) \ , \quad \lambda_3 = N_{G_{\text{mult}},m} \quad \text{and} \quad \lambda_4 = n$$

where M_{ac} is the top left 2×2 block matrix of M_{CC}

$$M_{ac} = \begin{pmatrix} N_{G_{\text{add}},a} & N_{G_{\text{copy}},a} \\ N_{G_{\text{add}},c} & N_{G_{\text{copy}},c} \end{pmatrix}$$

where $N_{x,y}$ denotes the number of gates x in a gadget y, with m for the multiplication, a for the addition, and c for the copy. We finally get

$$|\hat{C}| = \mathcal{O}\big(|C| \cdot N_{\max}^k\big) \quad \text{with} \quad N_{\max} = \max(\mathsf{eigenvalues}(M_{ac}), N_{G_{\text{mult}},m}, n) \ . \tag{6}$$

As an illustration, the expanding compiler from [9] satisfies $N_{\max} = 3n^2 - 2n$ and $d = \frac{min(t+1, n-t)}{2}$ which yields an asymptotic complexity of $\mathcal{O}(\kappa^e)$ with

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

In comparison, in this work, we shall achieve a quasi-linear complexity, *i.e.*, $N_{\max} = \mathcal{O}(n \log n)$.

2.5 Tolerated Leakage Rate

Finally, we recall the notion of maximum *tolerated leakage rate* 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.

In practice, the tolerated leakage rate should be measured on concrete devices and fixed accordingly. Hence the motivation to exhibit gadgets which tolerate a high probability to cover any setting. So far, the asymptotic constructions provide a trade-off between tolerated leakage rate and complexity. However, we only know how to compute the former for small numbers of shares and the bounds for larger values are not tight.

As an illustration, the instantiation proposed in [8] tolerates a leakage probability up to $2^{-7.80}$ for 3-share base gadgets.

3 Dynamic Random Probing Expansion

As recalled in Section 2, the principle of the expanding compiler is to apply a base circuit compiler CC which is composed of base gadgets –one per gate type in the circuit– several times, say k, to the input circuit: $\widehat{C} = \mathsf{CC}^{(k)}(C)$. The level of expansion k is chosen in order to achieve a certain desired security level κ such that $f^{(k)}(p) \leq 2^{-\kappa}$.

In this section, we generalize this approach to chose the circuit compiler dynamically at the different steps of the expansion. Let $\{\mathsf{CC}_i\}_i$ be a family of circuit compilers, the *dynamic expanding compiler* for this family with respect to the expansion sequence $k_1, \ldots k_\mu$, is defined as

$$\widehat{C} = \mathsf{CC}_\mu^{k_\mu} \circ \mathsf{CC}_{\mu-1}^{k_{\mu-1}} \circ \ldots \cdots \circ \mathsf{CC}_1^{k_1}(C) \ . \tag{7}$$

The idea behind this generalization is to make the most from a family of RPE compilers $\{\mathsf{CC}_i\}_i$ which is defined with respect to the number of shares n_i in the base gadgets. If we assume that each compiler CC_i with n_i shares achieves the maximum amplification order $d_i = \frac{n_i+1}{2}$, then the benefit of using a compiler with higher number of shares is to increase the amplification order and thus reduce the number of steps necessary to achieve the desired security level κ. On the other hand, the tolerated leakage rate of existing constructions decreases with n. As we show hereafter, a dynamic increase of n can ensure both, the tolerated leakage rate of a small n and the better complexity of a high n.

3.1 Dynamic Expanding Compiler

We formally introduce the dynamic expanding compiler hereafter.

Definition 7 (RPE Compiler). *Let $\mathbb{B} = \{g : \mathbb{K}^\ell \to \mathbb{K}^m\}$ be an arithmetic circuit basis. Let $n_i, t \in \mathbb{N}$, and let $\{G_g\}_{g\in\mathbb{B}}$ be a family of (t, f_g)-RPE n_i-share gadgets for the gate functionalities in $\mathbb{B}$. The* RPE compiler CC_i *associated to $\{G_g\}_{g\in\mathbb{B}}$ is the circuit compiler which consists in replacing each gate from a circuit over $\mathbb{B}$ by the corresponding gadget G_g. Moreover,*

– *the* expanding function *of* CC_i *is the function* f_i *defined as*

$$f_i : p \mapsto \max_g f_g(p)$$

– *the* amplification order *of* CC_i *is the integer* d_i *defined as*

$$d_i = \min_g d_g$$

where d_g *is the amplification order of* f_g*,*

– *the* gadget complexity *of* CC_i *is the integer* s_i *defined as*

$$s_i = \max_g |G_g|$$

where $|G_g|$ *denotes the number of wires in the gadget* G_g*,*

– *the* tolerated leakage rate *of* CC_i *is the real number* $q_i \in [0, 1)$ *such that* $f_i(p) < p$ *for every* $p < q_i$*.*

In the following, we state the security and asymptotic complexity of the dynamic expanding compiler. We start with a formal definition of this compiler:

Definition 8 (Dynamic Expanding Compiler). *Let* $\{\mathsf{CC}_i\}_i$ *be a family of RPE compilers. The* dynamic expanding compiler *for* $\{\mathsf{CC}_i\}_i$ *with* expansion levels $k_1, \ldots, k_\mu$*, is the circuit compiler* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *where*

1. *The input encoding* Enc *is a* $\left(\prod_{i=1}^{\mu} n_i^{k_i}\right)$*-linear encoding.*
2. *The output decoding* Dec *is the* $\left(\prod_{i=1}^{\mu} n_i^{k_i}\right)$*-linear decoding mapping.*
3. *The circuit compilation is defined as*

$$\mathsf{CC}(\cdot) = \mathsf{CC}_\mu^{k_\mu} \circ \mathsf{CC}_{\mu-1}^{k_{\mu-1}} \circ \ldots \cdots \circ \mathsf{CC}_1^{k_1}(\cdot) \ .$$

The following theorem states the random probing security of the dynamic expanding compiler. The proof of the theorem is very similar to the proof of RPE security (Theorem 2) from [8]. The main difference is that at each level of the expansion, we can use a different expanding compiler with different sharing orders. Besides that, the proof follows the same baselines as in [8]. For the sake of completeness, we provide the full proof in the full version of this paper.

Theorem 1 (Security). *Let* $\{\mathsf{CC}_i\}_i$ *be a family of RPE compilers with expanding functions* $\{f_i\}_i$*. The dynamic expanding compiler for* $\{\mathsf{CC}_i\}_i$ *with expansion levels* $k_1, \ldots, k_\mu$ *is* (p, ε)*-random probing secure with*

$$\varepsilon = f_\mu^{k_\mu} \circ \cdots \circ f_1^{k_1}(p) \ .$$

We now state the asymptotic complexity of the dynamic expanding compiler in the next theorem. The proof is given in the full version of this paper.

Theorem 2 (Asymptotic Complexity). *Let $\{\mathsf{CC}_i\}_i$ be a family of circuit compilers with complexity matrices $\{M_{\mathsf{CC}_i}\}_i$. For any input circuit C, the output circuit $\widehat{C} = \mathsf{CC}_{\mu}^{k_\mu} \circ \cdots\cdots \circ \mathsf{CC}_1^{k_1}(C)$ is of size*

$$|\widehat{C}| = |C| \cdot \mathcal{O}\Big(\prod_{i=1}^{\mu} \lambda_i^{k_i}\Big) \quad \textit{with} \quad \lambda_i := \max \ \mathsf{eigenvalues}(M_{\mathsf{CC}_i}) \ . \tag{8}$$

In the following, we shall call λ_i as defined above, the *eigen-complexity* of the compiler CC_i. We shall further call the product $\prod_{i=1}^{\mu} \lambda_i^{k_i}$ the *complexity blowup* of the dynamic expanding compiler. We note that minimizing the complexity blowup is equivalent to minimizing the log complexity blowup, which is

$$\sum_{i=1}^{\mu} k_i \cdot \log_2(\lambda_i) \ . \tag{9}$$

3.2 General Bounds for Asymptotic Constructions

The following theorem introduces general bounds on the tolerated leakage rate and the expanding function of an RPE compiler with respect to its amplification order and gadget complexity. The proof of the theorem is given in the full version of this paper.

Theorem 3. *Let CC_i be an RPE circuit compiler of amplification order d_i and gadget complexity s_i. The tolerated leakage rate q_i of CC_i is lower bounded by*

$$q_i \geq \bar{q}_i := \frac{1}{\mathrm{e}} \left(\frac{1}{2\,\mathrm{e}}\right)^{\frac{1}{d_i-1}} \left(\frac{d_i}{s_i}\right)^{1+\frac{1}{d_i-1}} \tag{10}$$

For any $p < \bar{q}_i$, the expanding function f_i of CC_i is upper bounded by

$$f_i(p) \leq 2\binom{s_i}{d_i} p^{d_i} \leq 2\left(\frac{\mathrm{e} \cdot s_i}{d_i}\right)^{d_i} p^{d_i} \ . \tag{11}$$

The lower bound $\bar{q}_i$ on the tolerated leakage rate quickly converges to the ratio $d_i/(\mathrm{e}s_i)$ as d_i grows. In other words, an RPE compiler family $\{\mathsf{CC}_i\}_i$ indexed by the number of shares n_i of its base gadgets tolerates a leakage probability which is linear in the ratio between its amplification order d_i and its complexity s_i. For known families of RPE compilers from [9] this ratio is in $\mathcal{O}(1/n_i)$.

From Theorem 3, we obtain the following bound for the composition $f_i^{(k)}$. The proof of the corollary is given in the full version of this paper.

Corollary 1. *Let CC_i be an RPE compiler of expanding function f_i, amplification order d_i and gadget complexity s_i. For any $p < \bar{q}_i$ as defined in* (10)*, we have*

$$f_i^{(k)}(p) \ \leq \ \left[2\binom{s_i}{d_i}\right]^{\left(1+\frac{1}{d_i-1}\right)d_i^{k-1}} p^{d_i^k} \ \leq \ \left[\left(\frac{2^{\frac{1}{d_i}}\mathrm{e}s_i}{d_i}\right)^{\left(1+\frac{1}{d_i-1}\right)} p\right]^{d_i^k} \ .$$

The following lemma gives an explicit lower bound on the expansion level $\{k_i\}_i$ to reach some arbitrary target probability $p_{out} = 2^{-\kappa_{out}}$ from a given input probability $p_{in} = 2^{-\kappa_{in}} n$ by applying $\mathsf{CC}_i^{(k_i)}$.

Lemma 1. *Let $p_{in} = 2^{-\kappa_{in}} < q_i$ and $p_{out} = 2^{-\kappa_{out}} \in (0, 1]$. For any integer k_i satisfying*

$$k_i \geq \log_{d_i}(\kappa_{out}) - \log_{d_i}(\kappa_{in} - \Delta_i)$$

with

$$\Delta_i := \left(1 + \frac{1}{d_i - 1}\right)\left(\frac{1}{d_i} + \log_2\left(\frac{\mathrm{e}s_i}{d_i}\right)\right)$$

we have

$$f_i^{(k_i)}(p_{in}) \leq p_{out} = 2^{-\kappa_{out}} \ .$$

In the above lemma, Δ_i represents a lower bound for κ_{in} which matches the upper bound $\bar{q}_i$ of $p_{in} = 2^{\kappa_{in}}$. Assuming that s_i and d_i are both monotonically increasing with i, we get that the threshold Δ_i tends towards $\log_2\left(\frac{\mathrm{e}s_i}{d_i}\right)$.

From Lemma 1, we further get that the cost induced by the choice of the compiler CC_i to go from an input probability p_{in} to a target output probability p_{out} is

$$k_i \cdot \log_2(\lambda_i) \geq \frac{\log_2(\lambda_i)}{\log_2(d_i)}\big(\log_2(\kappa_{out}) - \log_2(\kappa_{in} - \Delta_i)\big) \tag{12}$$

(in terms of the log complexity blowup (9)). Note that this upper bound is tight: it could be replaced by an equality at the cost of ceiling the term between parentheses (*i.e.* the term corresponding to k_i). We further note that the above equation is consistent with the complexity analysis of the expanding compiler provided in [8]. Indeed going from a constant leakage probability $p_{in} = p$ to a target security level $p_{out} = 2^{-\kappa}$ by applying k_i times a single RPE compiler CC_i, we retrieve a complexity of $\mathcal{O}(\kappa^e)$ with $e = \frac{\log_2(\lambda_i)}{\log_2(d_i)}$.

Equation (12) shows that using CC_i to go from input probability p_{in} to output probability p_{out} induces a log complexity cost close to

$$\frac{\log_2(\lambda_i)}{\log_2(d_i)}\big(\log_2(\kappa_{out}) - \log_2(\kappa_{in})\big)$$

provided that κ_{in} is sufficiently greater than Δ_i. So given the latter informal condition, it appears that the parameter i minimizing the ratio $\frac{\log_2(\lambda_i)}{\log_2(d_i)}$ gives the best complexity.

Application. For the asymptotic construction introduced in [9], the RPE compiler CC_i features

- an amplification order $d_i = \mathcal{O}(n_i)$,
- a gadget complexity $s_i = \mathcal{O}(n_i^2)$,
- an eigen-complexity $\lambda_i = \mathcal{O}(n_i^2)$.

For such a construction, the ratio $\frac{\log_2(\lambda_i)}{\log_2(d_i)}$ is decreasing and converging towards 2 as n_i grows. On the other hand Δ_i tends to $\log_2(n_i)$ which implies that CC_i should only be applied to an input probability lower than $\frac{1}{n_i}$.

3.3 Selection of the Expansion Levels

In this section, we investigate the impact of the choice of the expansion levels k_i on the complexity of the dynamic expanding compiler. We first assess the asymptotic complexity obtained from a simple approach and then provide some application results for some given gadgets.

In the following CC_0 shall denote an RPE compiler with constant parameters while $\{\mathsf{CC}_i\}_{i\geq 1}$ shall denote a family of RPE compilers indexed by a parameter i. We do this distinction since the goal of the CC_0 compiler shall be to tolerate the highest leakage rate and to transit from a (possibly high) leakage probability p to some lower failure probability p_i which is in turn tolerated by at least one compiler from $\{\mathsf{CC}_i\}_i$.

A Simple Approach. We consider a simple approach in which the compiler CC_0 is iterated k_0 times and then a single compiler CC_i is iterated k_i times. The complexity blowup of this compiler is $\lambda_0^{k_0}\lambda_i^{k_i}$. The first expansion level k_0 is chosen to ensure that the intermediate probability $p_i := f_0^{(k_0)}(p)$ is lower than $\bar{q}_i$ (the lower bound on the tolerated leakage rate of CC_i from Theorem 3). Then k_i is chosen so that $f_i^{(k_i)} \leq 2^{-\kappa}$.

Concretely, we set $\kappa_i := \Delta_i + 1$ which, by Lemma 1, gives

$$k_0 = \left\lceil \log_{d_0}(\Delta_i + 1) - \log_{d_0}(\log_2(p) - \Delta_0) \right\rceil \ , \tag{13}$$

and

$$k_i = \left\lceil \log_{d_i}(\kappa) \right\rceil = \mathcal{O}\big(\log_{d_i}(\kappa)\big) \ . \tag{14}$$

For some constant leakage probability p and some start compiler CC_0 with constant parameters, we get $k_0 = \mathcal{O}\big(\log_{d_0}(\Delta_i)\big)$ giving an asymptotic complexity blowup of

$$\mathcal{O}\big(\lambda_0^{k_0}\lambda_i^{k_i}\big) = \mathcal{O}\big(\Delta_i^{e_0}\kappa^{e_i}\big) \quad \text{with} \quad e_0 = \frac{\log_2(\lambda_0)}{\log_2(d_0)} \quad \text{and} \quad e_i = \frac{\log_2(\lambda_i)}{\log_2(d_i)} \ . \tag{15}$$

Then for any constant choice of i (*i.e.* irrespective to κ) we get an asymptotic complexity blowup of $\mathcal{O}\big(\kappa^{e_i}\big)$ which is the same asymptotic complexity as the standard expanding compiler with base compiler CC_i. On the other hand, our simple dynamic compiler $\mathsf{CC}_i^{(k_i)} \circ \mathsf{CC}_0^{(k_0)}$ tolerates the same leakage rate as CC_0.

Using this simple approach we hence get the best of both worlds

- a possibly inefficient RPE compiler CC_0 tolerating a high leakage rate q_0,

– a family of RPE compilers $\{\mathsf{CC}_i\}_i$ with complexity exponent $e_i = \frac{\log_2(\lambda_i)}{\log_2(d_i)}$ decreasing with i.

We stress that for monotonously increasing λ_i and d_i, the asymptotic complexity of our simple approach is $\mathcal{O}(\kappa^e)$ where e can be made arbitrary close to $\lim_{i\to\infty} \frac{\log_2(\lambda_i)}{\log_2(d_i)}$.

Application. To illustrate the benefits of our dynamic approach, we simply get back to the experimentations on the AES implementation from [8]. The authors apply either a 3-share or 5-share compiler repeatedly until they reach their targeted security level. While using the 5-share compiler reduces the tolerated probability, we demonstrate that we can use both compilers to get the best tolerated probability as well as a better complexity.

Figure 1 illustrates the trade-offs in terms of achieved security level and complexity of the expansion strategy when using different compilers at each iteration of the expansion. Starting from a tolerated leakage probability p ($2^{-7.6}$ on the left and $2^{-9.5}$ on the right), the empty bullets ($\circ$) give this trade-off when only the 3-share compiler is iterated. In this case, the final security function ε from Theorem 1 is equal to $f_3^{(k_3)}(p)$ if we consider f_3 to be the failure function of the 3-share compiler, for a certain number of iterations k_3 which is written next to each empty bullet on the figure. On the other hand, the black bullets ($\bullet$) represent the trade-offs achieved in terms of complexity and security levels while combining both compilers with different numbers of iterations. In this case, we start the expansion with a certain number of iterations k_3 of the 3-share compiler, and then we continue with k_5 iterations of the 5-share compiler of failure function f_5, the final compiled circuit is then random probing secure with $\varepsilon = f_5^{(k_5)}(f_3^{(k_3)}(p))$ for $p \in \{2^{-7.6}, 2^{-9.5}\}$. The number of iterations of the compilers is written next to each black bullet in the format k_3-k_5.

For instance, starting from the best tolerated probability $2^{-7.6}$, the static compiler from [8, 9] requires 11 applications of the 3-share compiler to achieve a security level of at least 80 bits. This effort comes with an overall complexity of $10^{17.52}$. Using our dynamic approach, we can combine the 3-share and the 5-share to achieve this 80 bits security level for the same tolerated probability but with a complexity of $10^{16.04}$. That would require 7 iterations of the 3-share compiler and 2 iterations of the 5-share compiler. Starting from the same leakage probability, a security level of at least 128 bits is achieved also with 11 applications of the 3-share compiler with a complexity of $10^{17.52}$. In order to achieve at least the same security, we would need more iterations of both compilers in the dynamic approach. With 7 iterations of the 3-share compiler and 3 iterations of the 5-share compiler, we get a complexity of $10^{17.62}$ which is very close to the complexity of the 3-share application alone, while achieving a security level of 231 bits. That is, we almost double the security level achieved using 11 iterations of the 3-share compiler with an almost equal complexity. For a tolerated probability of $2^{-7.6}$ and at least 128 bits of security, note that 11 applications of the 3-share compiler yield a security order of 2^{-135} while both other trade-offs directly yield security

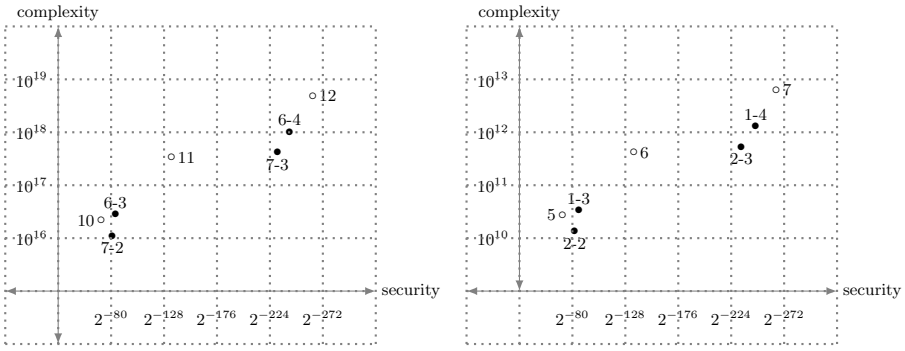

Fig. 1: Complexity of random probing AES for different security levels for a tolerated probability of $2^{-7.6}$ (left) or $2^{-9.5}$ (right).

orders of 2^{-242} (6 iterations of 3-share and 4 iterations of 5-share) and 2^{-231} (7 iterations of 3-share and 3 iterations of 5-share), with one less iteration they would be below 128 bits, which explains their more important complexity. The same behavior can be observed with a starting tolerated leakage probability of $2^{-9.5}$ on the right.

These results motivate our work in the rest of this paper, namely finding RPE compilers which achieve the maximal amplification orders and which benefit from good asymptotic complexity (*i.e.* gadgets defined for any number of shares n with amplification order increasing with n) in order to optimize the security-efficiency trade-off and to tolerate the best possible leakage probability. We showed this far that the tolerated leakage probability decreases with an increasing number of shares n. So if we want to tolerate the best leakage probability, we would start with a few iterations of a compiler with a small number of shares and which tolerates a good leakage probability (which can be computed for instance with the verification tool VRAPS[5] [8]), typically a 3-share construction. Meanwhile, after a few constant number of iterations, we can change to a different compiler, say with an asymptotic construction, which benefits from better asymptotic complexities, allowing a good overall asymptotic complexity. In the constructions from [9], the bottleneck in terms of asymptotic complexity was from the linear gadgets (addition and copy). Thanks to the quasilinear refresh gadget we introduce later in this paper, the bottleneck becomes the multiplication gadget (with n^2 multiplications), which we also improve in the following sections under some conditions on the base field.

4 Linear Gadgets with Quasi-Linear Complexity

In a first attempt, we aim to reduce the complexity of the linear gadgets that are to be used in our dynamic compiler.

[5] acronym for (V)erifier of (RA)ndom (P)robing (S)ecurity

In [9], the authors provide new constructions of generic addition and copy gadgets, using a refresh gadget G_{refresh} as a building block. The construction works for any number of shares and the authors prove the RPE security of the gadgets based on the security of G_{refresh}. In a nutshell, given a n-share refresh gadget G_{refresh}, the authors construct a copy gadget G_{copy} which on input sharing $(a_1, \ldots, a_n)$, outputs the sharings

$$\Big(G_{\text{refresh}}(a_1, \ldots, a_n), G_{\text{refresh}}(a_1, \ldots, a_n)\Big)$$

with two independent executions of G_{refresh}. The authors also construct an addition gadget G_{add} which, on input sharings $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$, first refreshes the inputs separately, then outputs the sharewise sum of the results

$$\Big(G_{\text{refresh}}(a_1, \ldots, a_n) + G_{\text{refresh}}(b_1, \ldots, b_n)\Big).$$

If the refresh gadget G_{refresh} is TRPE of amplification order d, the authors show that G_{copy} is also TRPE of amplification order d, and G_{add} is TRPE of amplification order at least $\lfloor d/2 \rfloor$.

While the copy gadgets from [9] achieve an optimal amplification order, this is not the case yet for addition gadgets and we first aim to fill this gap. Precisely, we introduce a new property which, when satisfied by its inherent refresh gadget G_{refresh}, makes the addition gadget TRPE with the same amplification order as G_{refresh}. We then prove that this new property is actually satisfied by the refresh gadget from [4] which has quasi-linear complexity $\mathcal{O}(n \log n)$ in the sharing order n. Using this refresh gadget as a building block, we obtain linear gadgets G_{add} and G_{copy} with quasi-linear complexities.

Constructions of Linear Gadgets from a Stronger Building Block. We first define our new property (as a variant of properties defined in [8, 9]) which proves to be a useful requirement for refresh gadgets when used as a building block of linear gadgets.

Definition 9 (t-(Strong)TRPE2). *Let G be an n-share 1-input gadget. Then G is t-(Strong)TRPE2 (we denote it t-STRPE2) if and only if for any set J' of output shares indices and any set W of probed wires such that $|W| + |J'| \leq t$, then there exists a set J of output shares indices such that $J' \subseteq J$ and $|J| = n-1$ and such that W and J can be perfectly simulated from input shares indexed in a set I such that $|I| \leq |W| + |J'|$.*

Remark 1. This new property directly implies the TRPE2 property with maximal amplification order introduced in [9]. Namely G is t-TRPE2 with maximal amplification order if and only if for any set W of probed wires such that $|W| < \min(t+1, n-t)$, there exists a set J of output shares indices such that $|J| = n-1$ and such that W and J can be perfectly simulated from input shares indexed in a set I such that $|I| \leq |W|$.

Having a refresh gadget which satisfies the property from Definition 9 results in tighter constructions for generic addition gadgets as stated in Lemma 2. Its proof is given in the full version of this paper.

Lemma 2. *Let $G_{refresh}$ be an n-share refresh gadget and let G_{add} be the share-wise sum of the independent refresh of its two inputs. Then if $G_{refresh}$ is (t, f)-TRPE for any $t \leq n-1$ of amplification order $d \geq \min(t+1, n-t)$ and $G_{refresh}$ is $(n-1)$-STRPE2 , then G_{add} is (t, f')-RPE (resp. (t, f')-TRPE) for any $t \leq n-1$ for some f' of amplification order $\min(t+1, n-t)$.*

Instantiation of Linear Gadgets with Quasi-Linear Refresh Gadget. The refresh gadget with $\mathcal{O}(n \log n)$ complexity was introduced in [4]. In a nutshell, the idea is to add a linear number of random values on the shares at each step, to split the shares in two sets to apply the recursion, and then to add a linear number of random values again. For the sake of completeness, we provide the full algorithm description in the full version of this paper. It benefits from a quasi-linear complexity in $\mathcal{O}(n \log n)$ and was proven to be $(n-1)$-SNI in [4]. In Lemma 3, we show that this gadget is also (t, f)-TRPE of amplification order $\min(t+1, n-t)$ and that it satisfies $(n-1)$-STRPE2 . The proof is given in the full version of this paper.

Lemma 3. *Let $G_{refresh}$ be the n-share refresh gadget described above from [4]. Then $G_{refresh}$ is (t, f)-TRPE for some function $f : \mathbb{R} \to \mathbb{R}$ of amplification order $d \geq \min(t+1, n-t)$. $G_{refresh}$ is additionally $(n-1)$-STRPE2 .*

Hence, we can actually explicitly construct the two linear gadgets G_{add} and G_{copy} with quasi-linear complexity in $\mathcal{O}(n \log n)$ using the above refresh gadget as a building block.

Regarding the asymptotic complexity of the expanding compiler, the eigenvalues λ_1, λ_2 from Section 2 are hence now both in $\mathcal{O}(n \log n)$. At this point, only the quadratic number of multiplications in the multiplication gadget still separates us from a compiler of quasi-linear complexity. We tackle this issue in the next section by constructing a generic multiplication gadget. We finally end up with a full expanding compiler with quasi-linear asymptotic complexity.

5 Towards Optimal Multiplication Gadgets

In the seminal works [8, 9], the number of multiplications is the prominent term of the expanding compiler's complexity. While the most deployed multiplication gadgets (*e.g.*, [16]) require a quadratic number of multiplications in the masking order, the authors of [7] exhibited a probing secure higher-order masking multiplication with only a linear number of bilinear (or non-scalar) multiplications. Their construction, which applies on larger fields, is built from the composition of two subgadgets G_{submult} and G_{compress}, as described in Figure 2. In a nutshell, on inputs a and b, the subgadget G_{submult} performs the multiplication between

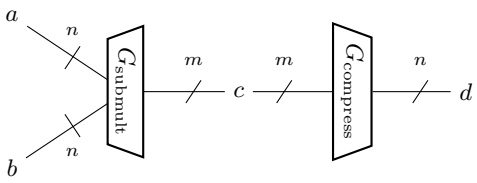

Fig. 2: n-share multiplication gadget G_{mult} from two subgadgets G_{submult} and G_{compress} such that $d = a \times b$

the input shares of a and b and outputs a m-share variable c where $m \geq n^6$. Next, the compression gadget G_{compress} compresses the m-share variable c back into n shares.

The authors of [7] instantiate this construction with a sub-multiplication gadget which performs only $\mathcal{O}(n)$ bilinear multiplications and with the compression gadget from [10]. In addition to regular additions, copies and non-linear multiplications, their sub-multiplication gadget additionally requires a quadratic number of linear multiplications (*i.e.*, multiplications by a constant).

In the following, we rely on the construction [7] with its gadget G_{submult} which offers a linear number of quadratic multiplications to build a more efficient RPE multiplication gadget. In order to use it in our expanding compiler, we integrate an additional gate for the multiplication by a constant and discuss the resulting asymptotic complexity. We additionally demonstrate that the compression gadget of [7] is not $(n-1)$-SNI as claimed in the paper, and show that we can rely on other simple and more efficient compression gadgets which satisfy the expected properties.

5.1 Global Multiplication Gadget

We first define two new properties that G_{submult} and G_{compress} will be expected to satisfy to form a (t, f)-RPE multiplication gadget with the maximum amplification order from the construction [7].

Contrary to the usual simulation notions, the first *partial*-NI property distinguishes the number of probes on the gadget, and the number of input shares that must be used to simulate them. It additionally tolerates a *simulation failure* on at most one of the inputs (*i.e.*, no limitation on the number of shares for the simulation).

Definition 10 ((x, y)-partial NI). *Let G be a gadget with two inputs a and b. Then G is (x, y)-partial NI if and only any set W of probes on G such that $|W| \leq y$ can be perfectly simulated from shares $(a_i)_{i \in I_a}$ of a and $(b_i)_{i \in I_b}$ of b such that $|I_a| \leq x$ **or** $|I_b| \leq x$.*

The second property is a variant of the classical TRPE property that we refer to as *comp-TRPE*.

[6] In case of a sharewise multiplication for instance, we would have $m = n^2$.

Definition 11 ((t, f)-comp-TRPE). *Let G be a one-input one-output gadget with m input shares and n output shares such that $m > n$. Let $t \leq n-1$ and $d = \min(t+1, n-t)$. Then G is (t, f)-comp-TRPE if and only if for all set of probes W on the internal wires of G with $|W| \leq 2d-1$, we have :*

1. *$\forall\ J, |J| \leq t$ a set of output shares of G, J and W can be simulated from a set of input shares I of the input of G, such that $|I| \leq |W|$.*
2. *$\exists\ J', |J'| = n-1$ a set of output shares of G, such that J' and W can be simulated from a set of input shares I of the input of G, such that $|I| \leq |W|$.*

Similarly to what was done in [7] for the SNI property, we can prove that the composition of a gadget G_{submult} and G_{compress} which satisfy well chosen properties results in an overall multiplication gadget which is (t, f)-RPE specifically for any $t \leq n-1$ achieving the maximum amplification order $d = \min(t+1, n-t)$. This is formally stated in Lemma 4 which proof is given in the full version of this paper.

Lemma 4. *Consider the n-share multiplication gadget of Figure 2 formed by a 2-to-1 multiplication subgadget $G_{submult}$ of m output shares and a 1-to-1 compression gadget $G_{compress}$ of m input shares such that $m > n$. Let $t \leq n-1$ and $d = \min(t+1, n-t)$. If*

- *$G_{submult}$ is $(d-1)$-NI and $(d-1, 2d-1)$-partial NI,*
- *$G_{compress}$ is (t, f)-comp-TRPE,*

then the multiplication gadget G_{mult} is (t, f)-RPE of amplification order d.

5.2 Construction of G_{compress}

In a first attempt, we analyze the compression function that was introduced in [10] and used to build a multiplication gadget in [7]. As it turns out not to be SNI or meet our requirements for the expanding compiler, we exhibit a new and also more efficient construction in a second attempt.

G_{compress} from [7, 10]. The authors of [7] use the $[m : n]$-compression gadget introduced in [10] for any input sharing m, using a $[2n : n]$-compression subgadget as a building block. In a nutshell, it first generates an ISW-refresh of the zero n-sharing $(w_1, \ldots, w_n)$. Then, these shares are added to the input ones $(c_1, \ldots, c_n)$ to produce the sequence of output shares $(c_1 + w_1, \ldots, c_n + w_n)$.

The compression gadget is claimed to be $(n-1)$-SNI in [7]. However, we demonstrate that it is not with the following counterexample. Let $n > 2$ and $i \in [n]$. We consider the set composed of a single output share of the compression procedure $J = \{(c_i + w_i) + c_{n+i}\}$ and the set of probes on the internal wires $W = \{w_i\}$. For the compression to be 2-SNI, we must be able to perfectly simulate both the wires in W and J with at most $|W| = 1$ share of the input c. However, we can easily observe that $(c_i + w_i) + c_{n+i} - w_i = c_i + c_{i+n}$ requires the two input shares c_i and c_{i+n} to be simulated, which does not satisfy the 2-SNI

property. In conclusion, the above gadget is actually not SNI, and interestingly it is not sufficient either for our construction, *i.e.* it does not satisfy Definition 11. This observation motivates our need for a new compression gadget which satisfies the necessary property for our construction.

New Construction for G_{compress}. In Algorithm 1, we exhibit a new $[m : n]$-compression technique using an m-share refresh gadget G_{refresh} as a building block. We demonstrate in Lemma 5 that this new compression gadget satisfies the necessary properties for our construction as long as $m \geq 2n$. The proof is given in the full version of this paper.

Algorithm 1: $[m : n]$-compression gadget

Input : $(c_1, \ldots, c_m)$ such that $m \geq 2n$, m-share refresh gadget G_{refresh}
Output: $(d_1, \ldots, d_n)$ such that $\sum_{i=1}^{n} d_i = \sum_{i=1}^{m} c_i$
$K \leftarrow \lfloor m/n \rfloor$;
$(c'_1, \ldots, c'_m) \leftarrow G_{\text{refresh}}(c_1, \ldots, c_m)$;
$(d_1, \ldots, d_n) \leftarrow (c'_1, \ldots, c'_n)$;
for $i = 1$ **to** $K - 1$ **do**
  $(d_1, \ldots, d_n) \leftarrow (d_1 + c'_{1+i \cdot n}, \ldots, d_n + c'_{n+i \cdot n})$;
end
for $i = 1$ **to** $m - K \cdot n$ **do**
  $d_i \leftarrow d_i + c'_{i+K \cdot n}$;
end
return $(d_1, \ldots, d_n)$;

Lemma 5. *Let $G_{compress}$ be the $[m : n]$-compression gadget from Algorithm 1 such that $m \geq 2n$. If $G_{refresh}$ is $(m-1)$-SNI and $(m-1)$-STRPE2 , then $G_{compress}$ is (t, f)-comp-TRPE (Definition 11).*

As shown in Section 4, the refresh gadget from [4] is actually $(m-1)$-SNI and $(m-1)$-STRPE2 for any sharing order m. This gadget can then be used as a building block for the $[m : n]$-compression gadget, giving it a complexity of $\mathcal{O}(m \log m)$, which, on top of satisfying the necessary properties, is also an improvement over the complexity of the proposed gadget in [7] which has a complexity of $\mathcal{O}(\lfloor \frac{m}{n} \rfloor n^2)$ (because it performs a n-share ISW-refreshing $\lfloor \frac{m}{n} \rfloor$ times, see [7] for more details on the algorithm).

5.3 Construction of G_{submult}

To complete the construction of the overall multiplication gadget, we now exhibit relevant constructions for G_{submult}. We first rely on the construction from [7]

which happens to achieve the desired goal in some settings. While all the cases are not covered by the state-of-the-art proposal, we then slightly modify the construction to meet all our requirements. Both constructions rely on linear multiplications that are not included yet on the expanding compiler. We thus start with a construction for this additional linear gadget that we further denote G_{cmult}.

Straightforward Construction for G_{cmult}. We give a straightforward construction for G_{cmult} in Algorithm 2 which simply multiplies each share of the variable input to the constant input and then applies a (t, f)-RPE refresh n-share gadget G_{refresh}. Basically, with a (T)RPE refresh gadget G_{refresh}, we obtain a (T)RPE linear multiplication gadget G_{cmult} as stated in Lemma 6. The proof is given in the full version of the paper.

Algorithm 2: n-share multiplication by a constant

Input : sharing $(a_1, \ldots, a_n)$, constant value c, n-share refresh gadget G_{refresh}
Output: sharing $(d_1, \ldots, d_n)$ such that $d_1 + \cdots + d_n = c.(a_1 + \ldots + a_n)$
$(b_1, \ldots, b_n) \leftarrow (c.a_1, \ldots, c.a_n)$;
$(d_1, \ldots, d_n) \leftarrow G_{\text{refresh}}((b_1, \ldots, b_n))$;
return $(d_1, \ldots, d_n)$;

Lemma 6. *Let $G_{refresh}$ be a (t, f)-(T)RPE n-share refresh gadget of amplification order d. Then G_{cmult} instantiated with $G_{refresh}$ is (t, f')-(T)RPE of amplification order d.*

Relying on an additional gate for the linear multiplication does not impact the security analysis and the application of the compilation, but it modifies the complexity analysis of the expanding compiler. From the analysis given in Section 2.4, a complexity vector is associated to each base gadget $N_G = (N_a, N_c, N_{cm}, N_m, N_r)^{\mathsf{T}}$ where N_a, N_c, N_{cm}, N_m, N_r stand for the number of addition gates, copy gates, multiplication by a constant gates, multiplication gates and random gates respectively in the corresponding gadget. The matrix M_{CC} is now a 5×5 square matrix defined as

$$M = \left(N_{G_{\text{add}}} \mid N_{G_{\text{copy}}} \mid N_{G_{\text{cmult}}} \mid N_{G_{\text{mult}}} \mid N_{G_{\text{random}}}\right)$$

including, for each vector, the number of linear multiplications. Five eigenvalues λ_1, λ_2, λ_3, λ_4, λ_5 are to be computed, *i.e.*, one more eigenvalue compared to the original compiler in the original setting.

We can consider as before that multiplication gates are solely used in G_{mult} ($N_{G_{\text{add}},m} = N_{G_{\text{copy}},m} = N_{G_{\text{cmult}},m} = 0$) and that multiplication by a constant

gates are eventually solely used in G_{cmult} and G_{mult} ($N_{G_{\text{add}},cm} = N_{G_{\text{copy}},cm} = 0$) which is the case in the constructions we consider in this paper. It can be checked that (up to some permutation) the eigenvalues satisfy

$$(\lambda_1, \lambda_2) = \mathsf{eigenvalues}(M_{ac}) \ , \quad \lambda_3 = N_{G_{\text{cmult}},cm} \ , \quad \lambda_4 = N_{G_{\text{mult}},m} \quad \text{and} \quad \lambda_5 = n$$

where M_{ac} is the top left 2×2 block matrix of M_{CC}

$$M_{ac} = \begin{pmatrix} N_{G_{\text{add}},a} & N_{G_{\text{copy}},a} \\ N_{G_{\text{add}},c} & N_{G_{\text{copy}},c} \end{pmatrix} \ .$$

We get two complexity expressions for the expansion strategy

$$|\hat{C}| = \mathcal{O}\big(|C| \cdot N_{\max}^k\big) \tag{16}$$

with $N_{\max} = \max(\mathsf{eigenvalues}(M_{ac}), N_{G_{\text{cmult}},cm}, N_{G_{\text{mult}},m}, n)$ and with the security parameter κ

$$|\hat{C}| = \mathcal{O}\big(|C| \cdot \kappa^e\big) \quad \text{with} \quad e = \frac{\log N_{\max}}{\log d} \ .$$

Note that exhibited construction for the linear multiplication gadget requires $N_{G_{\text{cmult}},cm} = n$ linear multiplications. Hence $\lambda_3 = N_{G_{\text{cmult}},cm} = \lambda_5 = N_{G_{\text{random}},r} = n$ and the global complexity (16) can be rewritten as

$$|\hat{C}| = \mathcal{O}\big(|C| \cdot N_{\max}^k\big) \quad \text{with} \quad N_{\max} = \max(\mathsf{eigenvalues}(M_{ac}), N_{G_{\text{mult}},m})$$

if the number of multiplications is greater than n. The asymptotic complexity of the RPE compiler is thus not affected by our new base gadget G_{cmult}. We now use it as a building block for our constructions of G_{submult}.

***G_{submult}* from [7].** The authors of [7] provide a $(n-1)$-NI construction for G_{submult} which outputs $2n-1$ shares while consuming only a linear number of quadratic multiplications in the masking order. We first recall their construction which relies on two square matrices of $(n-1)^2$ coefficients in the working field. As shown in [7], these matrices are expected to satisfy some condition for the compression gadget to be $(n-1)$-NI. Since we additionally want the compression gadget to be $(d-1, 2d-1)$-partial NI, we introduce a stronger condition and demonstrate the security of the gadget in our setting.

Let $\mathbb{F}_q$ be a finite field such that q is some prime power. Let $\boldsymbol{\gamma} = (\gamma_{i,j})_{1 \le i,j < n} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ be a constant matrix, and let $\boldsymbol{\delta} = (\delta_{i,j})_{1 \le i,j < n} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ be the matrix defined by $\delta_{i,j} = 1 - \gamma_{j,i}$ for all $1 \le i, j < n-1$. G_{submult} takes as input two n-sharings $\boldsymbol{a}$ and $\boldsymbol{b}$ and outputs the following a $(2n-1)$-sharing $\boldsymbol{c}$ such that:

- $c_1 = \Big(a_1 + \sum_{i=2}^{n}(r_i + a_i)\Big) \cdot \Big(b_1 + \sum_{i=2}^{n}(s_i + b_i)\Big)$
- $c_i = -r_i \cdot \Big(b_1 + \sum_{j=2}^{n}(\delta_{i-1,j-1} s_j + b_j)\Big)$ for $i = 2, \ldots, n$

- $c_{i+n-1} = -s_i \cdot \left(a_1 + \sum_{j=2}^{n} (\gamma_{i-1,j-1} r_j + a_j)\right)$ for $i = 2, \ldots, n$

where r_i and s_i are randomly generated values for all $2 \leq i \leq n$. It can be easily checked that G_{submult} performs $2n-1$ bilinear multiplications, and that it is correct, *i.e.* $\sum_{i=1}^{2n-1} c_i = \sum_{i=1}^{n} a_i \cdot \sum_{i=1}^{n} b_i$.

In [7], the authors prove that a gadget is $(n-1)$-NI if one cannot compute a linear combination of any set of $n-1$ probes which can reveal all of the n secret shares of the inputs and which does not include any random value in its algebraic expression. We refer to [7] for more details on this result.
Based on this result, the authors demonstrate in [7], that G_{submult} is $(n-1)$-NI if the matrices γ and δ satisfy Condition 1 that we recall below.

Condition 1 ***(from [7])*** *Let $\ell = (2(n-1)+4) \cdot (n-1) + 1$. Let $\boldsymbol{I}_{n-1} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ be the identity matrix, $\mathbf{0}_{x\times y} \in \mathbb{F}_q^{x\times y}$ be a matrix of zeros (when $y = 1$, $\mathbf{0}_{x\times y}$ is also written $\mathbf{0}_x$), $\mathbf{1}_{x\times y} \in \mathbb{F}_q^{x\times y}$ be a matrix of ones, $\boldsymbol{D}_{\boldsymbol{\gamma},j} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ be the diagonal matrix such that $D_{\boldsymbol{\gamma},j,i,i} = \gamma_{j,i}$, $\boldsymbol{T}_{n-1} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ be the upper-triangular matrix with just ones, and $\boldsymbol{T}_{\boldsymbol{\gamma},j} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ be the upper-triangular matrix for which $T_{\boldsymbol{\gamma},j,i,k} = \gamma_{j,i}$ for $i \leq k$:*

$$\boldsymbol{I}_{n-1} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix} \qquad \boldsymbol{D}_{\boldsymbol{\gamma},j} = \begin{pmatrix} \gamma_{j,1} & 0 & \ldots & 0 \\ 0 & \gamma_{j,2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma_{j,n-1} \end{pmatrix}$$

$$\boldsymbol{T}_{n-1} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 0 & 1 & & 1 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix} \qquad \boldsymbol{T}_{\boldsymbol{\gamma},j} = \begin{pmatrix} \gamma_{j,1} & \gamma_{j,1} & \cdots & \gamma_{j,1} \\ 0 & \gamma_{j,2} & & \gamma_{j,2} \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma_{j,n-1} \end{pmatrix}$$

We define the following matrices (with $n' = n-1$):

$$\boldsymbol{L} = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c} 1 & \mathbf{0}_{1\times n'} & \mathbf{0}_{1\times n'} & \mathbf{0}_{1\times n'} & \mathbf{0}_{1\times n'} & \ldots & \mathbf{0}_{1\times n'} & \mathbf{1}_{1\times n'} & \mathbf{1}_{1\times n'} & \ldots & \mathbf{1}_{1\times n'} \\ \mathbf{0}_{n'} & \boldsymbol{I}_{n'} & \mathbf{0}_{n'\times n} & \boldsymbol{I}_{n'} & \boldsymbol{I}_{n'} & \ldots & \boldsymbol{I}_{n'} & \boldsymbol{T}_{n'} & \boldsymbol{T}_{n'} & \ldots & \boldsymbol{T}_{n'} \end{array} \right)$$

$$\boldsymbol{M} = \left(\begin{array}{c|c|c|c|c|c|c|c|c} \mathbf{0}_{n'} & \mathbf{0}_{n'\times n} & \boldsymbol{I}_{n'} & \boldsymbol{I}_{n'} & \boldsymbol{D}_{\boldsymbol{\gamma},1} & \ldots & \boldsymbol{D}_{\boldsymbol{\gamma},n'} & \boldsymbol{T}_{n'} & \boldsymbol{T}_{\boldsymbol{\gamma},1} & \ldots & \boldsymbol{T}_{\boldsymbol{\gamma},n'} \end{array} \right)$$

Condition 1 is satisfied for a matrix $\boldsymbol{\gamma}$ *if for any vector $\boldsymbol{v} \in \mathbb{F}_q^{\ell}$ of Hamming weight $\mathsf{hw}(\boldsymbol{v}) \leq n-1$ such that $\boldsymbol{L} \cdot \boldsymbol{v}$ contains no coefficient equal to 0 then $\boldsymbol{M} \cdot \boldsymbol{v} \neq \mathbf{0}_{n-1}$.*

In the above condition, the matrices $\mathbf{L}$ and $\mathbf{M}$ represent the vectors of dependencies for each possible probe. All the probes involving shares of a for matrix $\boldsymbol{\gamma}$ (and symmetrically shares of b for matrix $\boldsymbol{\delta}$) are covered in the columns of $\mathbf{L}$ and $\mathbf{M}$. Namely, the first column represents the probe a_0. As it does not involve

any random, it results in a zero column in $\mathbf{M}$. The next columns represents the probes a_i, then the probes r_i. They are followed by columns for the probes $(a_i + r_i)$, then $(a_i + \gamma_{j-1,i-1} r_i)$ (for $2 \leq j \leq n$), then $a_1 + \sum_{i=2}^{k}(r_i + a_i)$ (for $2 \leq k \leq n$), and finally then $a_1 + \sum_{j=2}^{k}(\gamma_{i-1,j-1} r_j + a_j)$ (for $2 \leq i \leq n$ and $2 \leq k \leq n$. The above condition means that there is no linear combination of $(n-1)$ probes which can include the expression of all of the input shares, and no random variable.

From this result, it is then straightforward to conclude[7] that G_{submult} is $(d-1)$-NI for $d = \min(t+1, n-t)$ for any $t \leq n-1$. Lemma 4 also requires G_{submult} to be $(d-1, 2d-1)$-partial NI to get an overall RPE multiplication gadget. For G_{submult} to satisfy this second property, we need to rely on a stronger condition for matrices $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ that we present in Condition 2.

Condition 2 *Let* $z = (2(n-1)+4).(n-1)+1$. *Let* $\boldsymbol{I}_{n-1} \in \mathbb{F}_q^{(n-1)\times(n-1)}$, $\mathbf{0}_{\ell \times n} \in \mathbb{F}_q^{\ell \times n}$, $\mathbf{1}_{\ell \times n} \in \mathbb{F}_q^{\ell \times n}$, $\boldsymbol{D}_{\gamma,j} \in \mathbb{F}_q^{(n-1)\times(n-1)}$, $\boldsymbol{T}_{n-1} \in \mathbb{F}_q^{(n-1)\times(n-1)}$, $\boldsymbol{T}_{\gamma,j} \in \mathbb{F}_q^{(n-1)\times(n-1)}$ *and* $\boldsymbol{L}$ *and* $\boldsymbol{M}$ *the same matrices as defined in Condition 1.*

Condition 2 is satisfied for a matrix $\boldsymbol{\gamma}$ *if and only if for any vector* $\boldsymbol{v} \in \mathbb{F}_q^z$ *of Hamming weight* $\mathsf{hw}(\boldsymbol{v}) \leq n-1$, *and for any* $i_1, \ldots, i_K \in [z]$ *such that* $v_{i_1} \neq 0, \ldots, v_{i_K} \neq 0$ *and the corresponding columns* $i_1, \ldots, i_K$ *in* $\boldsymbol{L}$ *and in* $\boldsymbol{M}$ *have no zero coefficient (i.e there are* K *probes of the form* $a_1 + \sum_{i=2}^{n}(r_i + a_i)$ *or* $a_1 + \sum_{j=2}^{n}(\gamma_{i-1,j-1} r_j + a_j)$ *for any* $i \in \{2, \ldots, n\}$*), if* $\mathbf{M}.v = 0$, *then we have* $\mathsf{hw}(\boldsymbol{L} \cdot \boldsymbol{v}) \leq \mathsf{hw}(\boldsymbol{v}) - K$.

Based on this new condition, we can prove our second property G_{submult}, as stated in Lemma 7. The proof is given in the full version of this paper.

Lemma 7. *Let* $t \leq n-1$ *such that either* n *is even or* $t \neq \lfloor \frac{n-1}{2} \rfloor$ *and let* $d = \min(t+1, n-t)$. *Let* $G_{submult}$ *the multiplication subgadget introduced in [7]. If both matrices* γ *and* δ *satisfy Condition 2, then* $G_{submult}$ *is* $(d-1)$*-NI and* $(d-1, 2d-1)$*-partial NI.*

The condition on t and n on Lemma 7 implies that the maximum amplification order for the multiplication gadget cannot be achieved for an odd number of shares (since the maximum order is reached when $t = \lfloor \frac{n-1}{2} \rfloor$). This is not a proof artifact but a limitation of the gadget G_{submult} with respect to the new $(d-1, 2d-1)$-partial NI property. We can easily show that under this extreme conditions on t and n, we have $2d - 1 = n$. If we consider the instantiation of G_{submult} for $n = 3$ input shares, we obtain the following $2n - 1 = 5$ output

[7] Using the equivalence between non-interference and tight non-interference developed in [7].

shares:

$$
\begin{aligned}
c_1 &= (a_1 + (r_2 + a_2) + (r_3 + a_3)) \cdot (b_1 + (s_2 + b_2) + (s_3 + b_3)) \\
c_2 &= -r_2 \cdot (b_1 + (\delta_{1,1} \cdot s_2 + b_2) + (\delta_{1,2} \cdot s_3 + b_3)) \\
c_3 &= -r_3 \cdot (b_1 + (\delta_{2,1} \cdot s_2 + b_2) + (\delta_{2,2} \cdot s_3 + b_3)) \\
c_4 &= -s_2 \cdot (a_1 + (\gamma_{1,1} \cdot r_2 + a_2) + (\gamma_{1,2} \cdot r_3 + a_3)) \\
c_5 &= -s_3 \cdot (a_1 + (\gamma_{2,1} \cdot r_2 + a_2) + (\gamma_{2,2} \cdot r_3 + a_3))
\end{aligned}
$$

To prove the $(d-1, 2d-1)$-partial NI property, we need to ensure that any set of at most $2d-1 = 3$ probes can be perfectly simulated from at most $d-1 = 1$ shares of one of the input and any number of shares from the other one. However, the three probes on c_1, c_3, c_4 reveal information on each of their sub-product. In particular, $(a_1 + (r_2 + a_1) + (r_3 + a_3))$ (from c_1), r_3 (from c_3) and $(a_1 + (\gamma_{1,1} \cdot r_2 + a_2) + (\gamma_{1,2} \cdot r_3 + a_3))$ (from c_4) would reveal a. Similarly, $(b_1 + (s_2 + b_2) + (s_3 + b_3))$ (from c_1), $(b_1 + (\delta_{2,1} \cdot s_2 + b_2) + (\delta_{2,2} \cdot s_3 + b_3))$ from (c_3) and s_2 (from c_4) would reveal b. Hence, the gadget is not $(d-1, 2d-1)$-partial NI. This counterexample with 3 shares can be directly extended to any odd number of shares.

This counterexample motivates a new construction for G_{submult} which would cover all values for n and t. In the following, we slightly modify the construction from [7] to achieve the maximum amplification order in any setting.

Remark 2. The current construction of G_{submult} outputs $m = 2n-1$ shares, which does not satisfy the requirement $m \geq 2n$ shares for the compression gadget. Nevertheless, it is enough to add an artificial extra share c_{2n-1} equal to zero between both building blocks. In particular, the compression gadget (and subsequently and the refresh gadget) does not expect the input sharing to be uniform to achieve the stated security properties.

New Construction for G_{submult}. As stated earlier, Lemma 7 does not hold for G_{submult} in the case where n is odd and $t = (n-1)/2$. In order to cover this case, we propose a slightly modified version of G_{submult} with two extra random values r_0 and s_0. In this version, we let $\boldsymbol{\gamma} = (\gamma_{i,j})_{1 \leq i,j \leq n} \in \mathbb{F}_q^{n \times n}$ be a constant matrix, and let $\boldsymbol{\delta} \in \mathbb{F}_q^{n \times n}$ be the matrix defined by $\delta_{i,j} = 1 - \gamma_{i,j}$. The sub-gadget G_{submult} outputs $2n+1$ shares:

- $c_1 = \Big(\sum_{i=1}^{n} (r_i + a_i) \Big) \cdot \Big(\sum_{i=1}^{n} (s_i + b_i) \Big)$
- $c_{i+1} = -r_i \cdot \Big(\sum_{j=1}^{n} (\delta_{i,j} s_j + b_j) \Big)$ for $i = 1, \ldots, n$
- $c_{i+n+1} = -s_i \cdot \Big(\sum_{j=1}^{n} (\gamma_{i,j} r_j + a_j) \Big)$ for $i = 1, \ldots, n$

where r_i and s_i are randomly generated values. It can be easily checked that G_{submult} now performs $2n+1$ bilinear multiplications, and that it is correct, *i.e.* $\sum_{i=1}^{2n+1} c_i = \sum_{i=1}^{n} a_i \cdot \sum_{i=1}^{n} b_i$.

We now need the following slightly modified version of condition 2 on $\boldsymbol{\gamma}$ and on $\boldsymbol{\delta}$, which instead of considering a linear combination of at most $n-1$ probes as in Condition 2, considers up to n probes:

Condition 3 *Let $z = (2n+4) \cdot n$. Let $\boldsymbol{I}_n \in \mathbb{F}_q^{n \times n}$ be the identity matrix, $\mathbf{0}_{\ell \times n} \in \mathbb{F}_q^{\ell \times n}$ be the matrix of zeros, $\mathbf{1}_{\ell \times n} \in \mathbb{F}_q^{\ell \times n}$ be the matrix of ones, $\boldsymbol{D}_{\gamma,j} \in \mathbb{F}_q^{n \times n}$ be the diagonal matrix such that $\boldsymbol{D}_{\gamma,j,i,i} = \gamma_{j,i}$, $\boldsymbol{T}_n \in \mathbb{F}_q^{n \times n}$ be the upper triangular matrix with just ones, $\boldsymbol{T}_{\gamma,j} \in \mathbb{F}_q^{n \times n}$ be the upper triangular matrix such that $\boldsymbol{T}_{\gamma,j,i,k} = \gamma_{j,i}$ for $i \leq k$. We define the following matrices:*

$$\boldsymbol{L} = \left[\begin{array}{c|c|c|c|c|c|c|c|c} \boldsymbol{I}_n & \mathbf{0}_{n \times n} & \boldsymbol{I}_n & \boldsymbol{I}_n & \dots & \boldsymbol{I}_n & \boldsymbol{T}_n & \boldsymbol{T}_n & \dots \; \boldsymbol{T}_n \end{array} \right]$$
$$\boldsymbol{M} = \left[\begin{array}{c|c|c|c|c|c|c|c|c} \mathbf{0}_{n \times n} & \boldsymbol{I}_n & \boldsymbol{I}_n & \boldsymbol{D}_{\gamma,1} & \dots & \boldsymbol{D}_{\gamma,n} & \boldsymbol{T}_n & \boldsymbol{T}_{\gamma,1} & \dots \; \boldsymbol{T}_{\gamma,n} \end{array} \right]$$

Then we say that $\boldsymbol{\gamma}$ satisfies Condition 3 if and only if

- *for any vector $\boldsymbol{v} \in \mathbb{F}_q^z$ of Hamming weight $\mathsf{hw}(\boldsymbol{v}) \leq n$,*
- *for any $i_1, \dots, i_K \in [z]$ such that $v_{i_1} \neq 0, \dots, v_{i_K} \neq 0$ and the corresponding columns $i_1, \dots, i_K$ in $\boldsymbol{L}$ and in $\boldsymbol{M}$ have no zero coefficient (i.e there are K probes of the form $a_0 + \sum_{i=1}^{n-1}(r_i + a_i)$ or $a_0 + \sum_{j=1}^{n-1}(\gamma_{i,j} r_j + a_j)$ for any $i = 0, \dots, n-1$),*

if $\boldsymbol{M} \cdot \boldsymbol{v} = 0$, then we have $\mathsf{hw}(\boldsymbol{L} \cdot \boldsymbol{v}) \leq \mathsf{hw}(\boldsymbol{v}) - K$.

Under this new condition, we obtain the following result, whose proof is available in the full version.

Lemma 8. *Let $t \leq n-1$ and $d = \min(t+1, n-t)$. Let $G_{submult}$ as defined above with n-share inputs. If both matrices $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ satisfy Condition 3, then $G_{submult}$ is $(d-1)$-NI and $(d-1, 2d-1)$-partial NI.*

Proof. The proof of the Lemma is in fact the same as the proof of Lemma 7. The only difference is that in this lemma, we also cover the special case of an odd value for the number of shares n and $t = \lfloor \frac{n-1}{2} \rfloor = \frac{n-1}{2}$. In the latter case, we consider in the proof up to n probes on the gadget G_{submult}, while in Lemma 7, we could only have up to $n-1$ probes on the gadget. Since Condition 3 covers the case of having up to n probes on G_{submult}, then we can follow the exact same procedure of the proof of Lemma 7 to prove the Lemma by considering the new condition. □

Remark 3. The number of output shares $m = 2n+1$ of G_{submult} satisfies the constraint required by G_{compress} in Algorithm 1 ($m \geq 2n$). We can thus use the compression gadget G_{compress} exactly as described in the algorithm on the input sharing $(c_0, \dots, c_{2n})$, instantiated with the $\mathcal{O}(n \log n)$ refresh gadget from Section 4. Since the multiplication sub-gadget G_{submult} requires $\mathcal{O}(n)$ random values and G_{compress} requires $\mathcal{O}(n \log n)$ random values from the refresh gadget, the overall multiplication gadget G_{mult} also requires a quasi-linear number of random values $\mathcal{O}(n \log n)$.

5.4 Instantiations

We first state the existence of a matrix γ which satisfies Condition 3 over any finite field $\mathbb{F}_q$ for q large enough (with $\log(q) = \Omega(n \log n)$)[8]. The proof technique follows closely the proof of [7, Theorem 4.5] and makes use of the non-constructive "probabilistic method' and states that if one chooses γ uniformly at random in $\mathbb{F}_q^{n \log n}$, the probability that the matrix γ satisfies Condition 3 is strictly positive, when q is large enough. It is important to note that the proof relies on probability but the existence of a matrix γ which satisfies Condition 3 (for q large enough) is guaranteed without any possible error.

Theorem 4. *For any $n \geq 1$, for any prime power q, if γ is chosen uniformly in $\mathbb{F}_{q^{n \times n}}$, then*

$$\Pr[\gamma \textit{ satisfies Condition 3}] \geq 1 - 2 \cdot (12n)^n \cdot n \cdot q^{-1} \ .$$

In particular, for any $n \geq 1$, there exists an integer $Q = O(n)^{n+1}$, such that for any prime power $q \geq Q$, there exists a matrix $\gamma \in \mathbb{F}_q{}^{n \times n}$ satisfying Condition 3.

As when $\boldsymbol{\gamma}$ is uniformly random, so is $\boldsymbol{\delta}$, Theorem 4 immediately follows from the following proposition and the union bound.

Proposition 1. *For any $n \geq 1$, for any prime power q, if $\boldsymbol{\gamma}$ is chosen uniformly in $\mathbb{F}_q^{n \times n}$, then*

$$\Pr[\boldsymbol{\gamma} \textit{ satisfies Condition 3}] \geq 1 - (12n)^n \cdot n \cdot q^{-1} \ .$$

In particular, for any $n \geq 1$, there exists an integer $Q = O(n)^{n+1}$, such that for any prime power $q \geq Q$, there exists a matrix $\boldsymbol{\gamma} \in \mathbb{F}_q^{n \times n}$ satisfying Condition 3.

The proof of this proposition is very technical but follows essentially the proof of the analogous [7, Proposition 4.6]. For the sake of completeness, it is provided in the full version of this paper.

In [7], Belaïd et al. presented examples of matrices which satisfy their condition for 2 shares and 3 shares. Karpman and Roche [17] proposed afterwards new explicit instantiations up to order $n = 6$ over large finite fields and up to $n = 4$ over practically relevant fields such as $\mathbb{F}_{256}$. It is worth mentioning that the matrices proposed in [17] are actually incorrect (due to a sign error) but this can be easily fixed and we check that matrices obtained following [17] also achieves our Condition 3. These matrices for 3, 4 and 5 shares are provided in the full version of this paper.

[8] Such large finite fields may actually be useful to build efficient symmetric primitives (see for instance MiMC [**?**]).

6 Improved Asymptotic Complexity

In the previous sections, we exhibit the construction of a multiplication gadget G_{mult} which performs a linear number of multiplications between variables, and a quadratic number of multiplications by a constant operations. Using the results of Lemmas 5, 8 and 4, the constructed multiplication gadget is RPE and achieves the maximum amplification order $\lfloor \frac{n+1}{2} \rfloor$ for any number of shares n. Using the three linear gadgets proposed in Section 4 G_{add}, G_{copy}, G_{cmult} with the $\mathcal{O}(n \log n)$ base refresh gadgets, and the proposed construction of the multiplication gadget G_{mult}, we get an expanding compiler with a complexity matrix M_{CC} with the eigenvalues:

$$(\lambda_1, \lambda_2) = (n, -2n + 6n\log(n)) \ , \quad \lambda_3 = n \ , \quad \lambda_4 = 2n + 1 \quad \text{and} \quad \lambda_5 = n.$$

Hence we have $N_{\max} = \mathcal{O}(n \log n)$. In addition, since all of the base gadgets use a quasi-linear number of random values $\mathcal{O}(n \log n)$, we get an expanding compiler which is quasi-linear in terms of randomness complexity.

Figure 3 illustrates the evolution of the complexity exponent with respect to the number of shares n, for the best construction provided in [9] with quadratic complexity for an expanding compiler (orange curve), and our new arithmetic construction with quasi-linear complexity (pink curve). While the best construction from [9] yields a complexity in $\mathcal{O}(|C| \cdot \kappa^e)$ for e close to 3 for reasonable numbers of shares, the new expanding compiler quickly achieves a sub-quadratic complexity in the same settings.

7 Conclusion

Our dynamic expanding compiler instantiated with our optimized gadgets tolerates a leakage probability of $2^{-7.5}$ with an asymptotic complexity in $\mathcal{O}(\kappa^2)$. When the working finite field meets the requirement of our multiplication gadget, this asymptotic complexity becomes arbitrary close to linear, which is optimal.

As for concrete instantiations, our small example on the AES demonstrates the benefits of our dynamic compiler. Namely, it provides the best tolerated probability (from the best suited compiler) while optimizing the complexity using higher numbers of shares. Using two compilers with 3 and 5 shares instead of a single one already reduces the complexity by a factor 10.

To go further in the concrete use of our expanding compiler, future works could exhibit explicit constructions of matrices with (quasi)constant field size for our multiplication gadget. Another direction would be to build other constructions of RPE multiplication gadgets with also a linear number of multiplications on smaller fields. Eventually, one must optimize the tolerated probability of our compiler by investigating RPE gadgets which specifically maximize it.

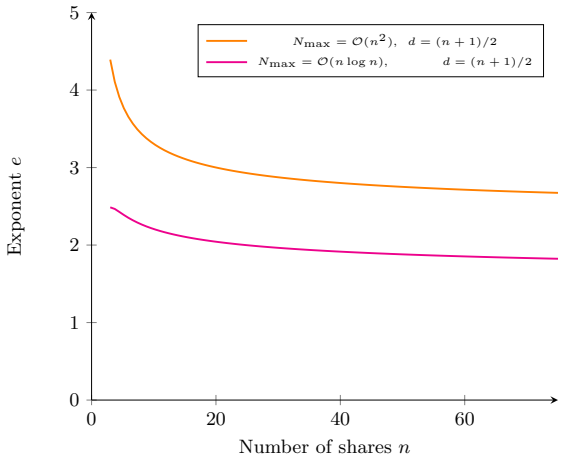

Fig. 3: Evolution of the complexity exponent $e = \log(N_{\max})/\log(d)$ with respect to the number of shares n. The orange curve matches the instantiation from[9] with quadratic asymptotic complexity ($N_{\max} = \mathcal{O}(n^2)$); the pink curve matches the new construction with quasi-linear asymptotic complexity ($N_{\max} = \mathcal{O}(n \log n)$).

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. Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 23–39. Springer, Heidelberg, August 2016.
5. 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. `https://eprint.iacr.org/2016/540`.
6. 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.

7. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Private multiplication over finite fields. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 397–426. Springer, Heidelberg, August 2017.
8. 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.
9. Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. *IACR Cryptol. ePrint Arch.*, 2021:434, 2021.
10. Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. Cryptology ePrint Archive, Report 2016/321, 2016. `https://eprint.iacr.org/2016/321`.
11. 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.
12. 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.
13. Jean-Sebastien Coron, Franck Rondepierre, and Rina Zeitoun. High order masking of look-up tables with common shares. Cryptology ePrint Archive, Report 2017/271, 2017. `https://eprint.iacr.org/2017/271`.
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. 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.
16. 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.
17. Pierre Karpman and Daniel S. Roche. New instantiations of the CRYPTO 2017 masking schemes. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 285–314. Springer, Heidelberg, December 2018.
18. 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.
19. Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 208–225. Springer, Heidelberg, February 2006.