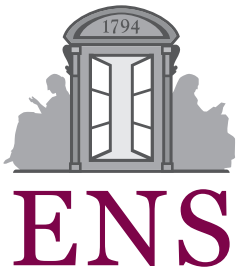

On the Provable Security of Cryptographic Implementations

Thèse d'habilitation

présentée et soutenue publiquement le 21 juin 2022
pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches de l'École normale supérieure (spécialité Informatique)

par

Matthieu Rivain

devant le jury composé de

Rapporteurs :	Pierre-Alain Fouque	(Université de Rennes 1)
	Yuval Ishai	(Technion)
	Bart Preneel	(KU Leuven)
Examinateurs :	Anne Canteaut	(INRIA)
	Jean-Sebastien Coron	(University of Luxembourg)
	Elisabeth Oswald	(University of Klagenfurt)
	David Pointcheval	(ENS, CNRS, PSL University)
	Emmanuel Prouff	(Sorbonne Université)
	Francois-Xavier Standaert	(UC Louvain)

Travaux effectués au sein de la société CryptoExperts

Remerciements

C'est confortablement installé sur un transat à l'ombre d'un parasol depuis la côte ouest crétoise observant par interstice la mer Méditerranée que j'écris ces quelques lignes afin de rendre hommage aux personnes sans qui cette thèse d'habilitation n'existerait pas.

En premier lieu, j'adresse mes remerciements à David Pointcheval pour avoir rendu possible cette habilitation à diriger des recherches au sein de l'École normale supérieure. Je le remercie pour son temps, ses conseils et son aide quant à la concrétisation de ce projet.

J'aimerais ensuite exprimer ma gratitude à Anne Canteaut, Jean-Sébastien Coron, Pierre-Alain Fouque, Yuval Ishai, Elisabeth Oswald, David Pointcheval, Bart Preneel, Emmanuel Prouff et François-Xavier Standaert qui me font le plaisir et l'honneur de constituer mon jury d'HDR. Je suis notamment touché par la présence de Yuval dont les résultats fondateurs ont inspiré plusieurs de mes travaux. Je remercie Bart, Pierre-Alain et Yuval d'avoir pris le temps de relire ce mémoire et d'écrire leurs rapports soutenant cette habilitation.

Outre leur participation au jury, je tiens à spécialement remercier Jean-Sébastien Coron et Emmanuel Prouff. Jean-Sébastien a joué un rôle déterminant dans ma carrière de chercheur, en m'offrant l'opportunité de faire une thèse CIFRE chez Oberthur dont il a été le directeur. Je le remercie pour son soutien et ses conseils tout au long de ma carrière. Je remercie Emmanuel pour avoir accompagné mes premiers pas de chercheur, pour ses conseils, nos discussions, et bien sûr, pour son exigence en matière œnologique (toujours en l'absence de sulfite), sans oublier nos aventures communes de la confrérie de rois du cryptage avec avec Guénaël Renault.

La recherche est avant tout un effort collectif. Elle avance par la confrontation des points de vue, l'échange des idées et l'union des forces vers un but commun. Je tiens donc à témoigner ma reconnaissance envers les différentes personnes avec qui j'ai eu la chance de m'adonner à ce processus d'échange, de confrontation et de collaboration. Je remercie chaleureusement mes différents coauteurs, désormais un peu nombreux pour être tous nommés ici. Je remercie tout particulièrement Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff et Damien Vergnaud pour nos nombreuses et fructueuses collaborations sans lesquelles cette thèse d'habilitation ne serait pas ce qu'elle est. Merci notamment à Sonia et Damien pour nos co-encadrements de stagiaires et thésards.

J'ai ici une pensée particulière pour quatre jeunes gens, plein de motivation et de talent, dont j'ai (eu) le privilège de co-encadrer les thèses ; j'ai nommé Thibauld Feneuil, Dahmun Goudarzi, Abdel Rahman Taleb et Junwei Wang. Je les remercie de m'avoir octroyé leur confiance dans cette entreprise. Merci notamment à Dahmun et Abdel pour nos collaborations dont les résultats nourrissent ce mémoire. Merci à Junwei avec qui nous nous sommes amusés à ouvrir et fermer la boîte blanche (à défaut de celle de Pandore). Et merci à Thibauld avec qui j'explore de nouveaux horizons sans nulle divulgation de connaissance (comme on dit en bon français).

Comme tout être vivant, le chercheur gagne à évoluer dans un environnement naturel propice à son épanouissement. Mon parcours de chercheur depuis la fin de mon doctorat est intimement lié au projet et à l'aventure que représente CryptoExperts. J'y ai trouvé –et espère avoir contribué à y construire– un environnement cultivant le savoir et l'innovation scientifiques. Je remercie donc la personne morale que constitue CryptoExperts de m'offrir un tel environnement depuis 12 ans. Mais plus personnellement, je tiens à exprimer ma reconnaissance envers mes associés Pascal Paillier et Louis Goubin qui, à l'origine accompagnés d'Aline Gouget et Christophe Clavier, ont enfanté le projet CryptoExperts. Merci particulièrement à Pascal qui m'y a accueilli en 2010 et au contact de qui j'ai beaucoup appris. Sa culture crypto, sa curiosité et son optimisme légendaire ont été une grande

source d'inspiration pour moi. Un grand merci également à Louis pour son écoute, sa bienveillance et pour nos différentes collaborations.

Cette petite douzaine d'années passées chez CryptoExperts, alias CRX, m'a donné l'occasion de rencontrer et de travailler avec des personnes dont les qualités personnelles, scientifiques et professionnelles m'ont beaucoup influencé. Merci donc à Thomas Baignères, Sonia Belaïd, Cécile Delerablée, Thibauld Feneuil, Matthieu Finiasz, Louis Goubin, Dahmun Goudarzi, Aline Gouget, Antoine Joux, Tancrède Lepoint, Darius Mercadier, Viet Sang Nguyen, Pascal Paillier, Abdul Rahman Taleb, Aleksei Udovenko et Junwei Wang ! Avec une pensée particulière pour Cécile et Thomas qui étaient là au début de l'aventure.

Pour différents échanges et autres collaborations qui ont également marqué mon parcours de chercheur, je remercie Thomas Roche et Victor Lomné (mes amis les ninjas), Darius Mercadier (et ses talents de compilateur), Aleksei Udovenko (et ses talents de white boxeur), Antoine Joux, Dan Page et François-Xavier Standaert.

Il n'y a pas que le travail dans la vie ! Mais quand travail rime avec passion, il est parfois difficile de tracer une limite ferme et de sacraliser les moments de repos et de partage avec les siens. Je remercie Claire qui me fait le bonheur de partager ma vie et qui m'aide sans relâche à préserver ces moments crypto-free (avec plus ou moins de succès dira-t-elle). Merci à mes parents, Françoise et Vincent, à ma sœur Elsa, à ma famille et mes amis pour leur présence et leur soutien.

Enfin, merci à toi, cher lecteur, qui t'apprête à persister au delà de cette page de remerciements et grâce à qui je l'espère ce travail de synthèse, au delà de sa stricte nécessité, n'aura pas été vain.

Contents

Part I.

Introduction

Chapter 1

Introduction

We live in a world in which cryptography has become ubiquitous. Devices around us are constantly processing cryptographic computations to ensure the confidentiality and the authenticity of our communications. As of today, cryptography is widely deployed to secure payment transactions, phone calls, wireless communications between devices, video and music streaming, and of course many communications on the Internet (including chat, video calls, user authentication for web services, etc.). Tomorrow, cryptography might increasingly be involved when accessing our homes and work offices, unlocking our cars, proving our identities to third parties (with legal value), sending instructions to our domestic appliances, using privacy-preserving artificial intelligence services, performing cryptocurrency/blockchain transactions, and much more.

Cryptography has experienced an impressive development in the last forty years. It has become a full-fledged branch of computer science and has provided the world with practical, reliable and well-understood tools to build secure communications (such as encryption, authentication, digital signature). Paradoxically, a strongly imbalanced conception of security persists depending on the level of abstraction of a cryptographic mechanism: one requires very high assurance for an algorithm or a protocol but very low assurance for its actual implementation. The scientific community and the industry (through standardization committees) have converged towards the paradigm of *provable security* for cryptographic algorithms and protocols. Under this paradigm, a cryptographic algorithm or protocol should come with a security proof (or security reduction) that, under some well-studied computational hardness assumptions, breaking the mechanism requires some considerable computing power (e.g. 2^{128} CPU instructions). Such a proof is usually limited to the so-called *black-box model* in which the adversary is assumed to have an input-output access to the cryptographic mechanism, the latter is hence considered as a black box answering some *e.g.* encryption requests.

In the late 1990's, Kocher, Jaffe and Jun have demonstrated that information leaking through so-called *side channels* could be exploited to practically break (black-box secure) cryptographic implementations [Koc96, KJJ99]. A side-channel adversary can for instance measure the running time of an implementation to mount what is known as a *timing attack*. Many cryptographic implementations have been (and are still frequently) shown vulnerable to this type of attacks. While designing constant-time algorithms has become the norm in cryptography, the current challenge resides in ensuring that timing attacks are systematically avoided even by inexperienced developers (*e.g.* by selecting cryptographic standards avoiding these pitfalls and/or developing formal verification tools or compilers for cryptographic implementations).

Besides the running time, a side-channel adversary might take advantage of a (partial) physical access to the computing device to monitor the power consumption and/or electromagnetic emanations of the device a.k.a. the *side-channel leakage*. This physical access is non-invasive in the sense that the adversary does not need to tamper with the device but only to passively observe its physical behavior. When measurable, this leakage enables devastating key-recovery attacks against unprotected implementations. Moreover, recent advances tend to ease the practice of physical side-channel attacks: turnkey side-channel equipment is more accessible (and at lower cost) than before. For example, the progress of far-field electromagnetic attacks relaxes the physical access requirement for potential

attackers, the advent of deep-learning based attacks relaxes the necessary expertise of potential attackers.

To face this threat, we present in this thesis some contributions toward the provable security of cryptographic implementations in the presence of side-channel leakage. Our approach relies on *masking* whose principle is to apply secret sharing at the computation level. The goal of masking is to randomize the intermediate variables processed by the computation in order to mitigate the side-channel information leakage. While the first masking countermeasures were *ad hoc* and of limited order, the results presented in this thesis have contributed to the formalization of masking security, the construction of masking schemes secure at any orders, the formalization of practically-relevant side-channel leakage models, and the construction of masking schemes achieving provable security under these models.

This thesis is organized as follows: Chapter 2 provides the necessary technical preliminaries. Chapter 3 introduces the concept of masking and presents the design of masking schemes at any order in the *probing security paradigm*. Chapter 4 introduces the *noisy leakage model*, a formal model which captures the physical reality of power and electromagnetic leakages. Chapter 5 and Chapter 6 address the composition of elementary *masking gadgets* into globally secure masking schemes under different probing security flavors. Chapter 7 and Chapter 8 finally describe concrete masking schemes achieving provable security in the noisy leakage model.

Chapter 2

Preliminaries

Contents

2.1. Basic notions and notations

Along this thesis, $\mathbb{K}$ shall denote a finite field while $\mathbb{F}_q$ shall denote the finite field with q elements. A $\mathbb{K}$-*linear map* shall refer to any function $f : \mathbb{K}^\ell \to \mathbb{K}^m$, for some $\ell, m \in \mathbb{N}$, such that $f(\boldsymbol{x}+\boldsymbol{y}) = f(\boldsymbol{x})+f(\boldsymbol{y})$ for every $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}^\ell$. 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 $\boldsymbol{x}|_I = (x_i)_{i \in I}$.

For a probability distribution $\mathcal{D}$, the notation $x \leftarrow \mathcal{D}$ means that x is sampled from $\mathcal{D}$. For a finite set S, the notation $x \leftarrow S$ means that x is uniformly sampled at random from S. For an algorithm $\mathcal{A}$, $out \leftarrow \mathcal{A}(in)$ further means that out is obtained by a call to $\mathcal{A}$ on input in (using uniform random coins whenever $\mathcal{A}$ is probabilistic).

The *statistical distance* between any two probability distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ (with same sample space), denoted $\Delta(\mathcal{D}_1; \mathcal{D}_2)$, is defined as

$$\Delta(\mathcal{D}_1; \mathcal{D}_2) := \frac{1}{2} \sum_x |p_{\mathcal{D}_1}(x) - p_{\mathcal{D}_2}(x)| \ ,$$

where $p_{\mathcal{D}_1}(\cdot)$ and $p_{\mathcal{D}_2}(\cdot)$ denote the probability mass functions of $\mathcal{D}_1$ and $\mathcal{D}_2$. Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are said ε-*close*, denoted $\mathcal{D}_1 \approx_\varepsilon \mathcal{D}_2$, if their statistical distance is upper bounded by ε, while they are identically distributed, denoted $\mathcal{D}_1 \overset{id}{=} \mathcal{D}_2$ if their statistical distance is null.

2.2. Arithmetic circuits

An *arithmetic circuit* over a field $\mathbb{K}$ is a labeled directed acyclic graph whose edges are *wires* and vertices are *arithmetic gates* representing operations over $\mathbb{K}$. We consider the following arithmetic gate:

- an addition gate, of fan-in 2 and fan-out 1, computes an addition over $\mathbb{K}$,
- an subtraction gate, of fan-in 2 and fan-out 1, computes an addition over $\mathbb{K}$,
- a multiplication gate, of fan-in 2 and fan-out 1, computes a multiplication over $\mathbb{K}$,
- a scalar multiplication gate, of fan-in 1 and fan-out 1, computes a multiplication by a constant over $\mathbb{K}$,

- a copy gate, of fan-in 1 and fan-out 2, outputs two copies of its input.

A *randomized arithmetic circuit* is equipped with an additional type of gate:

- a random gate, of fan-in 0 and fan-out 1, outputs a fresh uniform random value of $\mathbb{K}$.

A (randomized) arithmetic circuit is further formally composed of input gates of fan-in 0 and fan-out 1 and output gates of fan-in 1 and fan-out 0.

Along this thesis, the set of wires of a circuit C shall be denoted $\mathrm{Wires}(C)$. The size of a circuit C, denoted $|C|$, shall refer to the number of wires of C, *i.e.* $|C| = |\mathrm{Wires}(C)|$ (which is always greater than the number of gates). We shall call *output wires* the wires of C which are incoming an output gate. All the other wires are called the *internal wires* of C.

Evaluating an ℓ-input m-output circuit C consists in writing an input $\boldsymbol{x} \in \mathbb{K}^\ell$ in the input gates, processing the gates from input gates to output gates, then reading the output $\boldsymbol{y} \in \mathbb{K}^m$ from the output gates. Defining $\boldsymbol{y}$ as the output corresponding to an input $\boldsymbol{x}$ is denoted by $\boldsymbol{y} = C(\boldsymbol{x})$. During the evaluation process, each wire in the circuit is assigned with a value on $\mathbb{K}$. We call the tuple of all these wire values a *wire assignment* of C (on input $\boldsymbol{x}$). Along this thesis, we will consider the wire assignment of a circuit C on input $\boldsymbol{x}$ for a subset $\mathcal{W}$ of its wires, which we shall denote

$$\mathsf{AssignWires}(C, \mathcal{W}, \boldsymbol{x}) \in \mathbb{K}^{|\mathcal{W}|} \ .$$

We stress that for a randomized arithmetic circuit, which includes random gates, AssignWires can be thought of as a probabilistic algorithm (which randomly samples the output of random gates).

2.3. Sharing and gadgets

Let us recall that for some tuple $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n$ and for some set $I \subseteq [n]$, the tuple $(x_i)_{i \in I}$ is denoted $\boldsymbol{x}|_I$.

Definition 1 (Linear Sharing)**.** *Let $n \in \mathbb{N}$ and $\boldsymbol{v} \in \mathbb{K}^n$. For any $x \in \mathbb{K}$, an* $\boldsymbol{v}$-linear sharing *of x is a vector $\boldsymbol{x} \in \mathbb{K}^n$ such that $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = x$. A random vector $\boldsymbol{x}$ distributed over $\mathbb{K}^n$ is said to be a* uniform $\boldsymbol{v}$*-linear sharing of a variable x if for any set $I \subset [n]$, with $|I| < n$, the tuple $\boldsymbol{x}|_I$ is uniformly distributed over $\mathbb{K}^{|I|}$.*

The $\boldsymbol{v}$*-linear decoding mapping* is the function mapping $\boldsymbol{x} \in \mathbb{K}^n$ to $\langle \boldsymbol{v}, \boldsymbol{x} \rangle \in \mathbb{K}$. A $\boldsymbol{v}$*-linear encoding* is a probabilistic algorithm which on input $x \in \mathbb{K}$ outputs a uniform $\boldsymbol{v}$-linear sharing of x. Such an encoding typically samples $n-1$ random values $x_1, \ldots x_{n-1}$ over $\mathbb{K}$ and computes x_n in order to verify the $\boldsymbol{v}$-linear decoding relation $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = x$.

Definition 2 (Gadget)**.** *Let $g : \mathbb{K}^\ell \to \mathbb{K}^m$. We shall call an ($n$-*share*, ℓ-*to*-m) $\boldsymbol{v}$-*gadget *for g, a randomized arithmetic circuit G that maps an input $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell) \in (\mathbb{K}^n)^\ell$ to an output $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) \in (\mathbb{K}^n)^m$ while satisfying*

$$\big(\langle \boldsymbol{v}, \boldsymbol{y}_i \rangle\big)_{1 \leqslant i \leqslant m} = g\Big[\big(\langle \boldsymbol{v}, \boldsymbol{x}_i \rangle\big)_{1 \leqslant i \leqslant \ell}\Big]$$

with probability 1 over the internal randomness of G.

Along this thesis, we shall sometimes drop the $\boldsymbol{v}$ prefix when the vector $\boldsymbol{v}$ is clear from the context. Most of the time, we shall focus on the particular case $\boldsymbol{v} = (1, 1, \ldots, 1)$ for which the decoding function is simply $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = x_1 + \cdots + x_n$. In this case, we shall use the terminology of n*-linear sharing*, n*-linear encoding*, and n*-linear decoding* to refer to the above concepts, while making the number of shares explicit.

2.4. Circuit compilers

A circuit compiler transforms an input circuit C into a functionally equivalent circuit $\widehat{C}$ working on encoded variables in order to satisfy some security properties. In the following definitions, $\mathrm{poly}(\cdot)$

stands for asymptotically polynomial in the argument(s), *i.e.* $\mathrm{poly}(\alpha_1, \alpha_2) = \mathcal{O}(\alpha_1^{e_1}\alpha_2^{e_2})$ for some constants e_1, e_2.

Definition 3 (Circuit Compiler)**.** *A* circuit compiler *is a triplet of algorithms* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows:*

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

These three algorithms satisfy the following properties:

- ***Correctness:*** *For every arithmetic circuit* C *of input length* ℓ*, and for every* $\boldsymbol{x} \in \mathbb{K}^{\ell}$*, we have*
$$\Pr\big(\mathsf{Dec}\big(\widehat{C}(\mathsf{Enc}(\boldsymbol{x}))\big) = C(\boldsymbol{x})\big) = 1 \ , \ \text{where } \widehat{C} = \mathsf{CC}(C).$$
- ***Efficiency:*** *For some security parameter* $\kappa \in \mathbb{N}$*, the running time of* $\mathsf{CC}(C)$ *is* $\mathrm{poly}(\kappa, |C|)$*, the running time of* $\mathsf{Enc}(\boldsymbol{x})$ *is* $\mathrm{poly}(\kappa, |\boldsymbol{x}|)$ *and the running time of* $\mathsf{Dec}\big(\boldsymbol{y}\big)$ *is* $\mathrm{poly}(\kappa, |\boldsymbol{y}|)$*.*

The ratio $|C|/|\widehat{C}|$ between the size of the original circuit and the size of the output of the circuit compiler shall be referred to as the *complexity overhead* of the compiler.

We shall focus on a special kind of circuit compilers which relies on linear sharing of the variables and replace each gate from the original circuit by a gadget. Such *standard circuit compilers* are formally defined hereafter.

Definition 4 (Standard Circuit Compiler)**.** *Let* $\kappa \in \mathbb{N}$ *be some security parameter and let* $n = \mathrm{poly}(\kappa)$*. Let* $\{G\}$ *be a family of* n*-share gadgets for the different arithmetic gates (addition, multiplication, copy, etc.) over* $\mathbb{K}$*. The* standard circuit compiler *with* sharing order n *and* base gadgets $\{G\}$ *is the circuit compiler* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *satisfying the following:*

1. *The input encoding* Enc *applies a* n*-share linear encoding to each input of the circuit.*
2. *The output decoding* Dec *applies the linear decoding mapping to each output of the circuit.*
3. *The circuit compilation* CC *consists in replacing each gate in the original circuit by an* n*-share gadget with corresponding functionality from the base* $\{G\}$*, and each wire by a set of* n *wires carrying a linear sharing of the original wire. If the input circuit is a randomized arithmetic circuit, each of its random gates is replaced by* n *random gates (thus producing an* n*-linear sharing of a random value).*

For such a circuit compiler, the correctness and efficiency directly follows from the correctness and efficiency of the gadgets from $\{G\}$*. In particular we have* $|\widehat{C}| \leqslant |C| \cdot \max |G|$*, namely the complexity overhead of the standard circuit compiler is the complexity of its largest base gadget.*

We shall further consider standard circuit compiler making use of so-called *refresh gadgets*. Formally, a refresh 1-to-1 gadget is a gadget for the identity function. One usually expect some security properties from a refresh gadget such as the *uniformity* which holds when the output sharing is uniform and independent of the input sharing. But as we will see along this thesis, further security notions are desirable for the refresh gadget.

Definition 5 (Refreshing)**.** *A standard circuit compiler with* refreshing *is defined for a base of* n*-share gadgets* $\{G\}$ *together with an additional refresh gadget:*

- *In a standard circuit compiler with* partial refreshing*, the compilation* CC *further inserts refresh gadgets at carefully chosen location in* $\widehat{C}$*.*
- *In a standard circuit compiler with* full refreshing*, the compilation* CC *further inserts a refresh gadget at the output of* every *gadget of* $\widehat{C}$*.*

2.5. Simulation-based security notions

Along this thesis, a *simulator* should formally refer to a probabilistic algorithm. We use this terminology when the purpose of this algorithm is to produce an output which simulates a given distribution.

Simulation-based security notions consider an adversary that can gain access to a certain number of leaking wires in a circuit $\widehat{C}$ taking an encoded input $\mathsf{Enc}(x)$ and working on encoded values. While the topology of those leaking wires depend on the considered leakage model, all the simulation-based security notions introduced hereafter rely on the fact that those leaking wires can be (perfectly) simulated without any knowledge on the plain output. Formally, given any distribution $\mathcal{D}_{\boldsymbol{x}}$ of the plain input $\boldsymbol{x}$, those security notions require the existence of a simulator which outputs a distribution identical (or at least statistically close to) the assignment of the leaking wires of $\widehat{C}$ on input $\mathsf{Enc}(\boldsymbol{x})$ (where $\boldsymbol{x}$ is not input to the simulator).

In the following definitions, we shall consider an encoding Enc mapping $\mathbb{K}^{\ell}$ to $\mathbb{K}^{\ell'}$ for some $\ell, \ell' \in \mathbb{N}$, a randomized arithmetic circuit $\widehat{C}$ with input space $\mathbb{K}^{\ell'}$ (output space of the encoding), and a distribution $\mathcal{D}_{\boldsymbol{x}}$ defined over $\mathbb{K}^{\ell}$ (input space of the encoding). Those definition spaces are left implicit for the sake of simplicity.

Definition 6 (Probing Security)**.** *A randomized arithmetic circuit $\widehat{C}$ is* t-probing secure *w.r.t. an encoding* Enc *if there exists a simulator* Sim *which, for every set $\mathcal{W} \subseteq \mathrm{Wires}(\widehat{C})$, with $|\mathcal{W}| \leqslant t$, and for every distribution $\mathcal{D}_{\boldsymbol{x}}$, satisfies*

$$\mathsf{Sim}(\mathcal{W}) \stackrel{id}{=} \mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x}))$$

where $\boldsymbol{x}$ is randomly sampled from $\mathcal{D}_{\boldsymbol{x}}$.

Definition 7 (Region Probing Security)**.** *A randomized arithmetic circuit $\widehat{C}$ is* r-region probing secure *w.r.t. an encoding* Enc *if there exist a circuit partition $(C_1, C_2, \ldots, C_m) \equiv \widehat{C}$ and a simulator* Sim *which, for every sequence of sets $\mathcal{W}_1 \subseteq \mathrm{Wires}(C_1)$, $\mathcal{W}_2 \subseteq \mathrm{Wires}(C_2)$, $\ldots$, $\mathcal{W}_m \subseteq \mathrm{Wires}(C_m)$, with $|\mathcal{W}_i| \leqslant \lceil r|C_i| \rceil$, and for every distribution $\mathcal{D}_{\boldsymbol{x}}$, satisfies*

$$\mathsf{Sim}(\mathcal{W}) \stackrel{id}{=} \mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})) \ ,$$

where $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2 \cup \ldots \cup \mathcal{W}_m$ and $\boldsymbol{x}$ is randomly sampled from $\mathcal{D}_{\boldsymbol{x}}$. The parameter r is the probing rate *tolerated by $\widehat{C}$.*

While the above definitions apply to a randomized arithmetic circuit (with respect to an encoding), they naturally extend to circuit compilers as follows.

Definition 8. *A circuit compiler* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *is {t-, r-region} probing secure if for every arithmetic circuit C, the circuit $\widehat{C} = \mathsf{CC}(C)$ is {t-, r-region} probing secure with respect to* Enc*.*

Note that the above definitions consider *perfect* (region) probing security in the sense that the simulator is required to output a perfect simulation of the wire assignment. They naturally generalize to *statistical* (region) probing security for which the simulation is only required to be statistically close to the wire assignment, *i.e.*

$$\mathsf{Sim}(\mathcal{W}) \approx_{\varepsilon} \mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})) \ ,$$

for some parameter ε (which should be negligible in the security parameter). In that case, we shall refer to these notions as (t, ε)-*probing security* and (t, ε)-*region probing security*.

The random probing security notion introduced hereafter is statistical in nature. Informally speaking, in the random probing model, each wire of a circuit $\widehat{C}$ leaks its value with probability p (the *leakage probability*), all these leakage events being mutually independent. To formally define the random probing security notion, let us first introduce the *leaking-wires sampler*. For a leakage probability $p \in [0, 1]$ and a randomized arithmetic circuit $\widehat{C}$, it outputs a set $\mathcal{W}$, denoted as

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

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

Definition 9 (Random Probing Security)**.** *A randomized arithmetic circuit $\widehat{C}$ is* (p, ε)-random probing secure *w.r.t. encoding* Enc *if there exists a simulator* Sim *which, for every distribution $\mathcal{D}_{\boldsymbol{x}}$, satisfies*

$$\mathsf{Sim}() \approx_{\varepsilon} \left(\mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})), \mathcal{W}\right) ,$$

where $\mathcal{W}$ is randomly sampled from $\mathsf{LeakingWires}(\widehat{C}, p)$ *and $\boldsymbol{x}$ is randomly sampled from $\mathcal{D}_{\boldsymbol{x}}$.*

Remark 1. For the constructions described in this thesis, the simulator shall always sample $\mathcal{W}$ from $\mathsf{LeakingWires}(\widehat{C}, p)$, and thus get a perfect distribution for $\mathcal{W}$, from which it then attempts to simulate $\mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x}))$. For the sake of simplicity and without loss of generality, we shall therefore make the $\mathcal{W}$ output of the simulation implicit and aim for a simulator achieving

$$\mathsf{Sim}() \approx_{\varepsilon} \mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})) .$$

Here also, we obtain a random probing security definition for compilers as follows.

Definition 10. *A circuit compiler* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *is* (p, ε)*-random probing secure if for every arithmetic circuit C, the circuit $\widehat{C} = \mathsf{CC}(C)$ is* $(p, |C| \cdot \varepsilon)$*-probing secure with respect to* Enc*.*

Remark 2. The above security definitions consider the *stateless model* in which the compiled circuit has no memory. They can be naturally extended to the *stateful model* by considering circuits with memory (see formal definition in [ISW03]). In the stateful model, the circuit has a persistent memory from one invocation to another, and the compiled circuit is required to remain secure in the presence of $\{t$-probing, r-region probing, (p, ε)-random probing, ...$\}$ leakage for each invocation. This model, is relevant to the practical context of a cryptographic implementation for which a key must be stored in memory. We note that secure compilers for the stateless model can generally be extended to the stateful model. In a nutshell, one applies refreshing of the state encoding from one invocation to another (see for instance [Cor14, Section 4]). This might be at the cost of a slight degradation of the achieved security: for instance the probing secure schemes of [ISW03, Cor14] loose a factor 2 in terms of tolerated probes while turning to the stateful model.

Part II.

Provable security for masked implementations

Chapter 3

Masking schemes in the probing security paradigm

Contents

3.1. Introduction

Masking is one of the most (if not the most) widely deployed countermeasure against power and electromagnetic side-channel attacks. Many masking schemes have been designed to efficiently protect cryptographic implementations following the discovery of these attacks in the late nineties. However, for more than a decade, the proposed schemes were restricted to first (or low) masking orders which only provide limited security guaranties. In [RP10] we have shown how the *private circuits* construction of Ishai, Sahai and Wagner [ISW03] could be instrumental to design efficient masking schemes with arbitrary security orders. We applied this approach to the specific case of AES and showed in a subsequent work how to generalize it to further ciphers [CGP+12].

This chapter revisits these works. We first provide some background on the concept of masking as side-channel countermeasure in Section 3.2 and outline the Ishai-Sahai-Wagner (ISW) construction in Section 3.3. We then present in Section 3.4 the masking approach suggested in [RP10] which consists in generalizing and tightening the ISW scheme in view of its efficient implementation. This approach notably raises the issue of *mask refreshing* which is further discussed with an overview of relevant

follow-up works. Finally, Section 3.5 describes the efficient application of ISW-based masking to AES [RP10] and to further block ciphers [CGP+12].

3.2. Masking

3.2.1. Principle

Soon after the publication of the differential power analysis (DPA) [KJJ99], masking was suggested as a possible countermeasure in different works. It was patented by Kocher, Jaffe and Jun [KJJ98a, KJJ98b] and further independently published by Chari, Jutla, Rao and Rohatgi in [CJRR99] as well as Goubin and Patarin in [GP99].

The principle of masking is to randomize the intermediate variables processed by a cryptographic algorithm in order to break the statistical dependence between these data and the (mean) side channel leakage. To be specific, DPA exploits the difference of means $\mathrm{E}(T \mid x = 0) \neq \mathrm{E}(T \mid x = 1)$, where T is a leakage trace and x is a bit processed by the target implementation which can be predicted by guessing a small part of the key (*e.g.* a byte). Note that this difference is not expected to occur in every sample of T but in the leakage points corresponding to the processing of x. Now masking the implementation consists in replacing every processing of such bit x by a pair of variables $x \oplus r$ and r, where r is a uniformly drawn random bit. Some points of T will then depend on $x \oplus r$, other points will depend on r, but the randomness of r ensures that no point of T depends on x anymore which yields $\mathrm{E}(T \mid x = 0) = \mathrm{E}(T \mid x = 1)$ and prevents DPA.

A shortcoming of *first-order masking*, which relies on a single mask r, is its vulnerability to *second-order attacks*. Although no point in the leakage trace T depends on some predictable bit x anymore, some pairs of points are jointly dependent of x. In other terms, the (multivariate) second-order moment of T depends on x. This dependency can be exploited through a second-order attack combining the leakage on $x \oplus r$ together with the leakage on r (see for instance [Mes00, OMHT06, PRB09]).

3.2.2. Higher-order masking

In order to reach *higher-order security*, a natural idea is to increase the number of masks or equivalently to rely on *secret sharing* [Sha79]. Specifically, every sensitive intermediate variable x processed during the cryptographic computation is randomly split into n *shares* $x_1, \ldots, x_n$ which satisfy a completeness relation:

$$x = x_1 + \cdots + x_n \ , \tag{3.1}$$

over some additive finite group or field. To share a variable, $n-1$ shares are randomly drawn while the remaining one is derived to satisfy the above relation. Such linear sharing is the most common choice but other masking relation might be preferred in some contexts (see for instance Chapter 7).

Although such an n-sharing (a.k.a. $(n-1)$-order masking) might be vulnerable to a side-channel attack of order n, it can be argued that the sharing order n provides a sound security parameter against side-channel attacks.

3.2.3. Soundness of masking

The security of higher-order masking in the presence of noisy leakage was formally argued in the pioneering work of Chari, Jutla, Rao and Rohatgi in [CJRR99]. They assume a simplified but still relevant leakage model in which a bit x is masked as n random shares $x_1, \ldots, x_n$ satisfying Equation 3.1, and an adversary is given a noisy leakage $L_i = x_i + N_i$ on each share, where $N_i \leftarrow \mathcal{N}(0, \sigma)$ is a random Gaussian noise of standard deviation σ. Under this leakage model, the authors show that the number of samples required by any adversary to distinguish the distribution $(L_1, \ldots, L_n \mid x = 0)$ from the distribution $(L_1, \ldots, L_n \mid x = 1)$ is lower bounded by $\sigma^{n+\delta}$ where $\delta = 4 \log \alpha / \log \sigma$ for a success probability α. In other words, extracting some information from such (static) noisy leakage of the shares is exponentially difficult in the sharing order n (where the base of the exponentiation is

related to the amount of noise in the leakage). One can further argue about the practical difficulty of locating the n different leakage points corresponding to the target shares among the many samples composing a leakage trace, and which are subject to jittering. The combination of these theoretical and practical arguments make the sharing order a sound security parameter for a masked implementation.

3.2.4. Masking schemes

We call *masking scheme* an algorithm which applies the masking principle to some specific (cryptographic) comoputation, namely which performs the computation while solely manipulating shared variables. The goal of such a masking scheme is to be *complete*, *i.e.* ensuring that the masked process computes the correct output (once unmasked), while preserving some security property. In particular, a masking scheme is said *t-th order side-channel secure* if it ensures that any t-tuple of intermediate variables is statistically independent of the algorithm data (*e.g.* input and secret key). We stress that this security property is equivalent to the *probing security* in the circuit model formalized by Ishai, Sahai and Wagner in [ISW03] which is recalled hereafter.

Following the publication of the DPA attack and the principle of masking, many first-order masking schemes have designed targeting various algorithms and implementation contexts, see for instance [GP99, Mes01, AG01, BGK04, OMPR05]. A first attempt to design a generic (higher-order) masking scheme was made by Schramm and Paar in [SP06] but we showed in a joint work with Coron and Prouff that it was actually vulnerable to attacks of order 2 and 3 [CPR07]. We further designed a masking scheme achieving second-order security in a joint work with Dottax and Prouff [RDP08] but its generalization to higher orders was still open at that time, which was addressed by Coron meanwhile [Cor14].

In the rest of this chapter, we show how we filled this gap in our works [RP10, CGP+12] by relying on the ISW construction [ISW03].

3.3. Ishai-Sahai-Wagner (ISW) construction

In their pioneering work [ISW03], Ishai, Sahai and Wagner designed the first circuit compiler achieving what they formalized as *t-probing security*. In a nutshell, they showed how to compile any Boolean circuit into a new randomized circuit tolerating up to t *probes* on its wires without leaking any information about the processed data.

3.3.1. The ISW circuit compiler

The ISW scheme applies to Boolean circuits composed of AND and NOT gates, which is enough to represent any Boolean circuit. Each single wire of the original circuit is replaced by n wires carrying an n-sharing of the original variable. Each NOT gate is replaced by a NOT gadget which simply consists in applying a NOT gate to one of the input shares. Correctness simply follows since $\mathsf{NOT}(x) = \mathsf{NOT}(x_1) \oplus x_2 \cdots \oplus x_n$ for any sharing $(x_1, \ldots, x_n)$ of x. Each AND gate is replaced by an AND gadget defined as follows.

Let $x, y \in \mathbb{F}_2$ and let $z = \mathsf{AND}(x, y) = xy$. From two sharings $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ of x and y, we want to compute a sharing $(z_1, \ldots, z_n)$ of z. The AND gadget consists of the following steps:

1. For every $1 \leqslant i < j \leqslant n$, draw a random bit $r_{i,j}$.
2. For every $1 \leqslant i < j \leqslant n$, compute $r_{j,i} = (r_{i,j} \oplus x_i y_j) \oplus x_j y_i$.
3. For every $1 \leqslant i \leqslant d$, compute $z_i = x_i y_i \oplus \bigoplus_{j \neq i} r_{i,j}$.

In the above process, the use of brackets indicates the order in which the operations are performed, which is mandatory for security of the scheme.

The completeness of the solution follows from:

$$\begin{aligned}\bigoplus_i z_i &= \bigoplus_i \left(x_i y_i \oplus \bigoplus_{j\neq i} r_{i,j}\right) = \bigoplus_i \left(x_i y_i \oplus \bigoplus_{j>i} r_{i,j} \oplus \bigoplus_{j<i} (r_{j,i} \oplus x_i y_j \oplus x_j y_i)\right) \\ &= \bigoplus_i \left(x_i y_i \oplus \bigoplus_{j<i} (x_i y_j \oplus x_j y_i)\right) = \left(\bigoplus_i x_i\right)\left(\bigoplus_i y_i\right) \ .\end{aligned}$$

3.3.2. Probing security

Consider a Boolean circuit C and its encoded version $\widehat{C}$ produced by the ISW compiler with parameter $n = 2t + 1$. Ishai *et al.* show that $\widehat{C}$ achieves t-probing security (see formal definition in Section 2.5). Let $w \in [s]$ denote a wire index, where $s = |C|$ is the number of wires in C, let x^w denote the variable carried by the wire of index $w \in [s]$ in C, and let $\boldsymbol{x}^w = (x_1^w, \ldots, x_n^w)$ be the sharing of x^w which is an input/output sharing of some gadget in $\widehat{C}$. The proof of Ishai, Sahai and Wagner consists in showing that any wire on $\widehat{C}$ can be perfectly simulated from the shares $(x_i^w)_{w\in[s]}$ and $(x_j^w)_{w\in[s]}$ for two indices $i, j \in [n]$. Any tuple of t wires can hence be jointly simulated from $(x_i^w)_{w\in[s],i\in I}$ for a set $I \subseteq [n]$ containing at most $2t < n$ share indices. The uniform property of the sharings $\boldsymbol{x}^w$ implies that all these shares can be perfectly simulated with fresh and independent random bits. In other words, any tuple of t wires can be perfectly simulated independently of the plain variables $(x_i^w)_{w\in[s]}$.

3.4. Efficient ISW-based masking schemes

3.4.1. Generalization to arithmetic circuits

We first note that although described for Boolean circuits made of AND and NOT gates, the ISW scheme can be generalized to work on any field $\mathbb{K}$ with further arithmetic gates. Specifically, we consider arithmetic circuits over $\mathbb{K}$ made of addition gates, subtraction gates, multiplication gates and constant gates. Additionally, we consider gates computing $\mathbb{K}$-linear maps, namely functions $f : \mathbb{K}^\ell \to \mathbb{K}^m$ such that $f(\boldsymbol{x} + \boldsymbol{y}) = f(\boldsymbol{x}) + f(\boldsymbol{y})$ (addition and subtraction are particular cases of linear $\mathbb{K}$-linear maps from $\mathbb{K}^2$ to $\mathbb{K}$).

The generalized compiler represents each variable x of the original circuit by a random n-sharing $(x_1, \ldots, x_n) \in \mathbb{K}^n$ and each gate is replaced by a corresponding gadget defined as follows:

- *Addition gadget.* Given two n-sharings $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, the addition gadget outputs
$$\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, \ldots, x_n + y_n) \ .$$
This is done *via* n addition gates processing each share separately.
- *Subtraction gadget.* Given two n-sharings $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, the subtraction gadget outputs
$$\boldsymbol{x} - \boldsymbol{y} = (x_1 - y_1, \ldots, x_n - y_n) \ .$$
This is done *via* n subtraction gates processing each share separately.
- *Linear gadget.* More generally, given a $\mathbb{K}$-linear map $f : \mathbb{K}^\ell \to \mathbb{K}^m$ and ℓ input sharings $\boldsymbol{x}^j = (x_i^j)_{i\in[n]}$, with $j \in [\ell]$, the linear gadget for f outputs m sharings $\boldsymbol{y}^k = (y_i^k)_{i\in[n]}$, with $k \in [\ell]$, such that
$$(y_i^1, \ldots, y_i^m) = f(x_i^1, \ldots, x_i^\ell)$$
for every $i \in [n]$. This is done *via* n linear gates for f processing each share index separately.
- *Constant gadget.* Given a constant $c \in \mathbb{K}$, the constant gadget outputs $(c, 0, \ldots, 0) \in \mathbb{K}^n$. This is done *via* n constant gates.
- *Random gadget.* The random gadget is simply made of n random gates and outputs a uniform random n-tuple $(r_1, \ldots, r_n)$, or equivalently a uniform n-sharing of a random value r.

- *Multiplication gadget.* The multiplication gadget is similar to the AND gadget described above. This ISW multiplication gadget is depicted in Algorithm 1.

Algorithm 1 ISW multiplication gadget

Input: Sharings $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n$ and $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{K}^n$
Output: Sharing $\boldsymbol{z} = (z_1, \ldots, z_n)$ s.t. $\sum_i z_i = (\sum_i x_i)(\sum_i y_i)$

1: **for** $i = 1 \ldots n$ **do**
2: **for** $j = i + 1 \ldots n$ **do**
3: $r_{i,j} \leftarrow \mathbb{K}$
4: $r_{j,i} \leftarrow (x_i \cdot y_j - r_{i,j}) + x_j \cdot y_i$
5: **for** $i = 1 \ldots n$ **do**
6: $z_i \leftarrow x_i y_i$
7: **for** $j = 1 \ldots n$ with $j \neq i$ **do** $z_i \leftarrow z_i + r_{i,j}$
8: **return** $(z_1, \ldots, z_n)$

3.4.2. Tighter proof for the ISW multiplication gadget

Besides the above generalization, we also suggest in [RP10] that the probing security of the ISW scheme could be made tighter and actually optimal. Namely, we aim at achieving t-probing security using n-sharings with $n = t + 1$. We show that the ISW multiplication gadget achieves this tight probing security assuming that the two input sharing $\boldsymbol{x}$ and $\boldsymbol{y}$ are mutually independent.

Theorem 1. *Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ be two n-sharings. For any tuple of wires $\boldsymbol{w} = (w_1, \ldots, w_t)$ of the ISW multiplication gadget (Algorithm 1) on input $\boldsymbol{x}$ and $\boldsymbol{y}$, there exist two sets $I \subseteq [n]$ and $J \subseteq [n]$ with $|I|, |J| \leqslant t$ such that $\boldsymbol{w}$ can be perfectly simulated from $\boldsymbol{x}|_I$ and $\boldsymbol{y}|_J$.*

The proof given in [RP10] (see Appendix A) follows the outlines of the proof of Ishai *et al.* but instead of building a single set I which gets two indexes per probe, we build two sets I and J (one per input sharing) each of which gets (at most) one index per probe.

We stress that the above theorem does not prove the tight probing security of the generalized ISW compiler. Without further precaution, we could face the case of an ISW multiplication gadget taking as input a sharing $\boldsymbol{x} = (x_1, \ldots, x_n)$ and a sharing $\boldsymbol{y} = (f(x_1), \ldots, f(x_n))$ for some linear map $f : \mathbb{K} \to \mathbb{K}$ (or simply $\boldsymbol{x} = \boldsymbol{y}$ as a special case). In such a context, a perfect simulation of t probes on the multiplication gadget would require t shares from $\boldsymbol{x}$ and t shares from $\boldsymbol{y}$, which is $2t$ shares from $\boldsymbol{x}$. Then we would get the same non-tight probing security as the original ISW scheme. To circumvent this issue, we suggested in [RP10] to use gadgets that *refresh* one of the input sharings to deal with cases of dependency between the input sharings.

3.4.3. Refresh gadgets and the composition issue

We propose in [RP10] the refresh gadget depicted in Algorithm 2. It simply consists in remasking $n-1$ out of n shares with $n-1$ fresh random masks and accumulating the sum of masks in the last share. It is not hard to show that for any input sharing $\boldsymbol{x}$, Algorithm 2 outputs a fresh uniform sharing of the same value. Thanks to this property, and as a corollary of Theorem 1, the ISW multiplication of $\boldsymbol{x}$ and $G_{\mathsf{R}}(\boldsymbol{y})$ is tightly probing secure (*i.e.* t-probing secure with $t = n - 1$) whatever the previous relation between the sharings $\boldsymbol{x}$ and $\boldsymbol{y}$.

Although this tight probing security holds at the scale of the multiplication gadget, it unfortunately falls at the global scale. In other words, the refresh gadget depicted in Algorithm 2 is insufficient to ensure the composition security of the proposed approach. We indeed showed in a joint work with Coron, Prouff and Roche [CPRR14] that the composition $G_{\otimes}(\boldsymbol{x}, G_{\mathsf{R}}(G_f(\boldsymbol{x})))$ is not tightly probing secure, with $G_{\otimes}$ being the ISW multiplication gadget, G_{R} being our simple refresh gadget

Algorithm 2 Simple refresh gadget

Input: Sharing $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n$
Output: Sharing $\boldsymbol{y} = (y_1, \ldots, y_n)$ s.t. $\sum_i y_i = \sum_i x_i$

$(y_1, \ldots, y_n) \leftarrow (x_1, \ldots, x_n)$
for $i = 1 \ldots n-1$ **do**
  $r_i \leftarrow \mathbb{K}$
  $y_i \leftarrow y_i + r_i$
  $y_n \leftarrow y_n - r_i$
return $(y_1, \ldots, y_n)$

(Algorithm 2) and G_f being a linear gadget. Specifically, by probing $\lceil \frac{n-1}{2} \rceil$ variables from $G_\otimes$ as well as 1 variable from from G_R, one can recover some information about the plain variable. We further show that this flaw can be avoided by defining a new ISW-like gadget for a function of the form $x \mapsto x \cdot f(x)$, where f is a $\mathbb{K}$-linear map. This gadget then comes as a replacement of the composition $G_\otimes(\boldsymbol{x}, G_\mathsf{R}(G_f(\boldsymbol{x})))$ while achieving tight probing security. But the tight probing security of a more global (and possibly complex) composition was left open in our work [CPRR14].

A more general alternative consists in using the ISW multiplication gadget as a refresh gadget. The principle which was originally suggested in [DDF14] consists in evaluating $G_\mathsf{R} : \boldsymbol{x} \mapsto G_\otimes(\boldsymbol{x}, (1, 0, \ldots, 0))$ where $G_\otimes$ is the ISW multiplication. By definition, the output $G_\mathsf{R}(\boldsymbol{x})$ is a fresh sharing of the variable encoded by $\boldsymbol{x}$. Note that this refresh gadget does not formally require the evaluation of an ISW multiplication but can instead be computed as depicted in Algorithm 3.

Algorithm 3 ISW refresh gadget

Input: Sharing $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n$
Output: Sharing $\boldsymbol{y} = (y_1, \ldots, y_n)$ s.t. $\sum_i y_i = \sum_i x_i$

$(y_1, \ldots, y_n) \leftarrow (x_1, \ldots, x_n)$
for $i = 1 \ldots n$ **do**
  for $j = i + 1 \ldots n$ **do**
    $r_{i,j} \leftarrow \mathbb{K}$
    $y_i \leftarrow y_i + r_{i,j}$
    $y_j \leftarrow y_j - r_{i,j}$
return $(y_1, \ldots, y_n)$

The issue of secure composition of masking gadgets (with tight probing security) was addressed in the subsequent work of Barthe, Belaïd, Dupressoir, Fouque, Grégoire, Strub and Zucchini [BBD+16]. They notably introduce the notion of *strong non-interference* (SNI) for a masking gadget, which makes it inherently composable to achieve tight probing security. They show that several ISW-based gadgets (specifically, the ISW multiplication gadget, the ISW refresh gadget, and our ISW-like gadget for $x \mapsto x \cdot f(x)$ functions) all satisfy the SNI property. They further describe a tool maskComp which automatically places refresh gadgets in order to ensure the tight probing security of the composition. However, this placement is overconservative and might insert more refresh gadgets than necessary.

Another conservative and systematic approach that we suggested in a joint work with Goudarzi [GR17] consists in applying an ISW refresh gadget to one input sharing of every multiplication gadget in the circuit. We conjectured that such an approach provide a tight probing secure composition. We were able to formally prove this conjecture in a follow-up work with Belaïd and Goudarzi [BGR18] and to further develop an exact verification tool for the placement of refresh gadgets in an ISW-based masked circuit.

3.5. Efficient application to block ciphers

3.5.1. Masking block ciphers

We consider SPN-like block ciphers[1] which iterate a round function made of a key addition, an s-box layer and a linear layer. Such a structure is a widely used to design symmetric ciphers: among others, DES [FIP99b], AES [AES01] and the SHA3 permutation [FIP99a] are SPN-like block ciphers in this sense. Such a cipher transforms an input plaintext into an output ciphertext through the repetition of a key-dependent permutation, called the *round transformation*. Such a round transformation looks like $\rho_{\boldsymbol{k}} : \boldsymbol{x} \mapsto \lambda \circ \gamma(\boldsymbol{x} + \boldsymbol{k})$, where $\boldsymbol{x}$ in the input state, $\boldsymbol{k}$ is the round key, λ is a $\mathbb{K}$-linear map for some field $\mathbb{K}$, $\boldsymbol{x} + \boldsymbol{k}$ is the addition on the same field, and γ is a non-linear layer. The latter usually consists of the parallel application of a function S called the *s-box*: $\gamma : (x_1, \ldots, x_s) \mapsto (\mathrm{S}(x_1), \ldots, \mathrm{S}(x_s))$. The s-box is the only non-linear ingredient in such a cipher. Usually, the base field $\mathbb{K}$ is an extension of $\mathbb{F}_2$, say $\mathbb{K} = \mathbb{F}_{2^m}$ for some m and the s-box S is an invertible m-bit mapping.

While applying the general ISW-based masking approach described in Section 3.4 to such a cipher, the linear layer and the key addition are easily dealt with using sharewise linear gadgets. The efficient implementation of the s-box on the other hand is more tricky. We address this issue hereafter, first for the special case of AES and then for any s-boxes.

3.5.2. Application to AES

The AES cipher [AES01] follows the above structure over the base field $\mathbb{F}_{256}$. Specifically, the AES state is composed of 16 elements of $\mathbb{F}_{256}$ and the linear layer is defined as a linear map $(\mathbb{F}_{256})^{16} \to (\mathbb{F}_{256})^{16}$. We can therefore naturally apply the ISW-based masking approach described in Section 3.4.

Masking the s-box. The AES S-box is defined as the composition of an affine transformation with the power function $x \mapsto x^{254}$ over $\mathbb{F}_{2^8}$. The affine transformation is an $\mathbb{F}_{2^8}$-linear map with a constant addition, which simply gives rise to a linear sharewise gadget (and a constant gadget).

For the exponentiation, a possible idea is to decompose $x \mapsto x^{254}$ into a sequence of multiplications by applying an exponentiation algorithm (*e.g.* the square-and-multiply algorithm). Such an approach was previously used to design a first-order masking scheme for AES [BGK04]. However, we can do better here by observing that squaring is a linear map on $\mathbb{F}_{2^8}$. In other words, any sharing $(x_1, \ldots, x_n)$ of $x \in \mathbb{F}_{2^8}$ satisfies:

$$x_1^2 + \ldots + x_n^2 = x^2 \ .$$

The square operation can hence be masked using a linear sharewise gadget. This observation further holds for any power of 2 since $x_1^{2^m} + \ldots + x_n^{2^m} = x^{2^m}$ for every $m \in \mathbb{N}$.

Our masking of the AES s-box hence relies on sharwise gadgets for squares, and more generally for functions $x \mapsto x^{2^m}$, and on ISW multiplication gadgets for standard non-linear $\mathbb{F}_{2^8}$ multiplications. From a complexity viewpoint, and for an increasing order n, most of the computation is devoted to the multiplication gadgets, which involve $\mathcal{O}(n^2)$ operations and random generations, whereas the linear gadgets only involve n applications of the original operation. For the sake of efficiency, we aim at minimizing the number of multiplication gadgets in our computation of $x \mapsto x^{254}$ while tolerating a broader used of squaring.

It can be checked that an exponentiation to the power 254 on the field $\mathbb{F}_{2^8}$ requires at least 4 non-linear multiplications (this is argued in Section 3.5.3). Algorithm 4 achieves this lower bound and requires few additional squares. Algorithm 5 depicts the corresponding masked algorithm where $G_\otimes$ denotes the ISW multiplication gadget, G_{R} denotes a refresh gadget (see Section 3.4.3), and $G_{(\cdot)^{2^m}}$ denotes a sharewise gadget for the $x \mapsto x^{2^m}$ linear map. We note that this algorithm achieves tight probing security when G_{R} is the ISW-based refresh gadget (Algorithm 3) which was shown in [BBD+16].

[1] SPN meaning *substitution-permutation network*.

Algorithm 4 Exponentiation to the 254

Input: $x \in \mathbb{F}_{256}$
Output: x^{254}

1: $z \leftarrow x^2$ ▷ $z = x^2$
2: $y \leftarrow z \cdot x$ ▷ $y = x^3$
3: $w \leftarrow y^4$ ▷ $w = x^{12}$
4: $y \leftarrow y \cdot w$ ▷ $y = x^{15}$
5: $y \leftarrow y^{16}$ ▷ $y = x^{240}$
6: $y \leftarrow y \cdot w$ ▷ $y = x^{252}$
7: $y \leftarrow y \cdot z$ ▷ $y = x^{254}$
8: **return** y

Algorithm 5 Masked exponentiation

Input: Sharing $\boldsymbol{x}$ of $x \in \mathbb{F}_{256}$
Output: Sharing $\boldsymbol{y}$ of $y = x^{245}$

1: $\boldsymbol{z} \leftarrow G_{(\cdot)^2}(\boldsymbol{x})$
2: $\boldsymbol{y} \leftarrow G_{\otimes}(\boldsymbol{x}, G_{\mathsf{R}}(\boldsymbol{z}))$
3: $\boldsymbol{w} \leftarrow G_{(\cdot)^4}(\boldsymbol{y})$
4: $\boldsymbol{y} \leftarrow G_{\otimes}(\boldsymbol{y}, G_{\mathsf{R}}(\boldsymbol{w}))$
5: $\boldsymbol{y} \leftarrow G_{(\cdot)^{16}}(\boldsymbol{y})$
6: $\boldsymbol{y} \leftarrow G_{\otimes}(\boldsymbol{y}, \boldsymbol{w})$
7: $\boldsymbol{y} \leftarrow G_{\otimes}(\boldsymbol{y}, \boldsymbol{z})$
8: **return** y

3.5.3. Efficient decomposition of any s-boxes

As explained above, the linear layer and key addition in an SPN-like cipher can be easily masked with sharewise gadgets. The s-box on the other hand might be a random-looking m-bit mapping without particular structure (unlike the AES s-box). We explain hereafter how to derive an efficient masking for any s-boxes.

We first note that a function S from $\{0,1\}^m$ to $\{0,1\}^m$ has a polynomial representation on the field $\mathbb{F}_{2^m}$, that is

$$\mathrm{S} : x \in \mathbb{F}_{2^m} \rightarrow \sum_{i=0}^{2^m-1} a_i x^i \ . \tag{3.2}$$

The coefficients a_i of this polynomial representation can be obtained from the look-up table of S by applying Lagrange interpolation theorem. The polynomial representation of the s-box can be straightly evaluated based on four kinds of operations over $\mathbb{F}_{2^m}$: additions, scalar multiplications (*i.e.* multiplications by constants), squares, and regular multiplications (*i.e.* of two different variables). In the masked setting, the latter operation gives rise to an ISW multiplication gadget, while all the former ones are $\mathbb{F}_{2^m}$-linear which can be efficiently evaluated through sharewise gadgets.

While the above observation already provides a masking scheme for any s-box, a trivial evaluation of Equation 3.2 results in a high number of (inefficient) non-linear multiplications. This leads us to the definition of the *masking complexity* of an s-box [CGP+12] (a.k.a. *non-linear complexity* [CRV14]) as the minimal number of nonlinear multiplications required to evaluate its polynomial representation. We note that the masking complexity of an s-box is invariant when composed with $\mathbb{F}_2^m$-affine bijections in input and/or in output. In other words, affine equivalent s-boxes have the same masking complexity.

We address hereafter the issue of finding polynomial evaluations of an s-box that aim at minimizing the number of nonlinear multiplications. Those constructions will enable us to deduce upper bounds on the masking complexity of an s-box. We first study the case of power functions whose polynomial representation has a single monomial (*e.g.* the AES s-box). For these functions, we exhibit the exact masking complexity by deriving addition chains with minimal number of nonlinear multiplications. We then address the general case and provide efficient heuristics to evaluate any s-box with a low number of nonlinear multiplications.

The case of power functions. We first consider s-boxes for which the polynomial representation is a single monomial. These s-boxes are usually called *power functions* in the literature. We describe a generic method to compute the masking complexity of such s-boxes. Our method involves the notion of *cyclotomic class*.

Definition 11. *Let $\alpha \in [0; 2^m - 2]$. The* cyclotomic class *of α is the set C_α defined by:*

$$C_\alpha = \{\alpha \cdot 2^i \bmod 2^m - 1;\ i \in [0; m-1]\}.$$

The study of cyclotomic classes is interesting in our context since a power x^α can be computed from a power x^β without any nonlinear multiplication if and only if α and β lie in the same cyclotomic

class. Hence, all the power functions with exponents within a given cyclotomic class have the same masking complexity.

To compute the masking complexity for an element in a cyclotomic class, we use the following observation: determining the masking complexity of a power function $x \mapsto x^\alpha$ amounts to find the addition chain for α with the least number of additions which are not doublings (see [Knu88] for an introduction to addition chains). This kind of addition chain is usually called a 2-*addition chain*. Let $(\alpha_i)_i$ denote some addition chain. At step i, it is possible to obtain any element within the cyclotomic classes $(C_{\alpha_j})_{j \leqslant i}$ using doublings only. As we are interested in finding the addition chain with the least number of additions which are not doublings, the problem we need to solve is the following: given some $\alpha \in C_\alpha$, find the shortest chain $C_{\alpha_0} \to C_{\alpha_1} \to \cdots \to C_{\alpha_k}$ where $C_{\alpha_0} = C_1$, $C_{\alpha_k} = C_\alpha$ and for every $i \in [1; k]$, $\alpha_i \in \left(\cup_{j<i} C_{\alpha_j}\right) + \left(\cup_{j<i} C_{\alpha_j}\right)$.

We shall denote by $\mathcal{M}_k^m$ the class of exponents α such that $x \mapsto x^\alpha$ has a masking complexity equal to k. The family of classes $(\mathcal{M}_k^m)_k$ is a partition of $[0; 2^m - 2]$ and each $\mathcal{M}_k^m$ is the union of one or several cyclotomic classes. For a small dimension m, we can proceed by exhaustive search to determine the shortest 2-addition chain(s) for each cyclotomic class. We implemented such an exhaustive search from which we obtained the masking complexity classes $\mathcal{M}_k^m$ for $m \leqslant 11$ (note that in practice most s-boxes have dimension $m \leqslant 8$). The results are provided in [CGP+12] (see Appendix B).

Interestingly, the *inverse function* $x \mapsto x^{2^m-2}$ related to the cyclotomic class $C_{2^{n-1}-1}$ always has the highest masking complexity. In particular, the inverse function $x \mapsto x^{254}$ (for $m = 8$) used in the AES has a masking complexity of 4 (as claimed in Section 3.5.2).

Heuristics for general s-boxes. We describe hereafter some heuristics for minimizing the number of non-linear multiplications in the evaluation of an s-box without assuming anything about its mathematical structure (*i.e.* with possibly random coefficients in its polynomial representation).

Cyclotomic method. Let q denote the number of distinct cyclotomic classes of $[0; 2^m - 2]$. The polynomial representation of S can be written as:

$$\mathrm{S}(x) = a_0 + a_{2^m-1} x^{2^m-1} + \sum_{i=1}^{q} Q_i(x) \ , \tag{3.3}$$

where the Q_i's are polynomials each composed of powers from a single cyclotomic class C_{α_i}, namely we can write $Q_i(x) = \sum_j a_{i,j} x^{\alpha_i 2^j}$ for some coefficients $a_{i,j}$ in $\mathbb{F}_{2^m}$. Let us then denote L_i the linearized polynomial $L_i(x) = \sum_j a_{i,j} x^{2^j}$ which is a $\mathbb{F}_2^m$-linear function of x. The *cyclotomic method* simply consists in deriving the powers x^{α_i} for each cyclotomic class C_{α_i} as well as x^{2^m-1} if $a_{2^m-1} \neq 0$, and in evaluating Equation 3.3 with $Q_i(x) = L_i(x^{\alpha_i})$. The powers x^{α_i} can each be derived with a single nonlinear multiplication. This is obvious for the α_i lying in $\mathcal{M}_1^m$. Then it is clear that every power x^{α_i} with $\alpha_i \in \mathcal{M}_{k+1}^m$ can be derived with a single multiplication from the powers $(x^{\alpha_i})_{\alpha_i \in \mathcal{M}_k^m}$. The power x^{2^m-1} can then be derived with a single nonlinear multiplication from the power x^{2^m-2}. The cyclotomic method hence involves a number of nonlinear multiplications equal to the number of cyclotomic classes, minus 2 (as x^0 and x^1 are obtained without nonlinear multiplication), plus 1 (to derive x^{2^m-1}).

Parity-split method. This method is composed of two stages. The first stage derives a set of powers $(x^j)_{j \leqslant q}$ for some q. The second stage essentially consists in a recursive application of the following equation (known as Knuth-Eve polynomial evaluation [Knu62, Eve64]):

$$Q(x) = Q_1(x^2) + Q_2(x^2)x \ , \tag{3.4}$$

where for any polynomial Q of degree t, there exist two polynomials Q_1 and Q_2 of degree at most $\lfloor t/2 \rfloor$ satisfying the above equation. The parity-split method consists in applying Equation 3.4 recursively r times. Those recursive calls involve $2^r - 1$ non-linear multiplications and the obtained polynomials are linear combinations of the powers $(x^{2^r j})_{j \leqslant 2^{m-r-1}}$ which can be derived with $2^{m-r-1} - 1$ non-linear multiplications. A detailed description of the method is provided in our paper [CGP+12] (see

Appendix B). We obtain a total number of non-linear multiplications of:

$$\min_{0\leqslant r\leqslant n}(2^{n-r-1}+2^r)-2=\begin{cases}3\cdot 2^{(n/2)-1}-2 & \text{if } n \text{ is even,}\\ 2^{(n+1)/2}-2 & \text{if } n \text{ is odd.}\end{cases}$$

where the minimum is reached for $r=\lfloor\frac{n}{2}\rfloor$.

CRV method. In a follow-up work [CRV14], Coron, Roy and Vivek introduce an improvement of the cyclotomic method. The so-called CRV method is today the most efficient heuristic evaluation method to minimize the number of nonlinear multiplications in a random s-box. In a nutshell, the CRV method consists in representing an s-box S as

$$\mathrm{S}(x)=\sum_{i=1}^{t-1}P_i(x)\cdot Q_i(x)+P_t(x)\ , \tag{3.5}$$

where the P_i's and Q_i's are polynomials whose monomials belong to $x^L:=\{x^\alpha\ ;\ \alpha\in L\}$ with L being the union of several cyclotomic classes. This union set is defined as $L=C_{\alpha_1=0}\cup C_{\alpha_2=1}\cup C_{\alpha_3}\cup\ldots\cup C_{\alpha_\ell}$ such that for every $i\geqslant 3$, $\alpha_i=\alpha_j+\alpha_k \bmod 2^m-1$ for some $j,k<i$. This makes it possible to compute all the monomials in x^L with $\ell-2$ non-linear multiplications. The total number of nonlinear multiplications in the evaluation of Equation 3.5 is hence of $(\ell-2)+(t-1)$. It can be shown that minimizing the number of nonlinear multiplications while ensuring the existence of such a representation leads to parameters $t\approx\sqrt{2^m/m}$ and $\ell\approx\sqrt{2^m/m}$.

Table 3.1 summarizes the masking complexity (a.k.a. nonlinear complexity) achieved by the above methods on the base field $\mathbb{F}_{2^m}$ for different values of m.

Table 3.1.: Masking complexity of different evaluation methods.

$m=$	3	4	5	6	7	8	9	10
Cyclotomic	1	3	5	11	17	33	53	105
Parity-Split	2	4	6	10	14	22	30	46
CRV	-	2	4	5	7	10	14	19

Extensions of the CRV method. In some follow-up works, we have extended the CRV method to different notions of masking complexity arising from different implementation contexts. In particular, we have considered in [CPRR15] the decomposition of an s-box into a few evaluations of low-degree functions (for which we define a probing-secure gadgets). We have further proposed in [GR16] efficient CRV-like heuristics to minimize the Boolean multiplicative complexity of an s-box (with application to bitslice masked implementations). In [GRVV17] we further generalized this approach to minimize the number of multiplications on any base field from $\mathbb{F}_2$ to $\mathbb{F}_{2^m}$. Table 3.2 summarizes the achievements of these different methods.

Table 3.2.: Masking complexities of different CRV-like heuristics.

$m=$	4	5	6	7	8	9	10
$\mathbb{F}_{2^m}$ multiplications [CRV14]	2	4	5	7	10	14	19
Quadratic $\mathbb{F}_{2^m}\to\mathbb{F}_{2^m}$ eval. [CPRR15]	3	4	5	8	11	17	26
Cubic $\mathbb{F}_{2^m}\to\mathbb{F}_{2^m}$ eval. [CPRR15]	2	3	3	4	4	-	-
$\mathbb{F}_2$ multiplications [GR16]	6	11	19	33	56	93	143
$\mathbb{F}_4$ multiplications [GRVV17]	4	-	12	-	31	-	73
$\mathbb{F}_{16}$ multiplications [GRVV17]	2	-	-	-	17	-	-

3.6. Conclusion and related works

In this chapter, we have shown how the ISW private circuit framework could be used to design arbitrary-order masking schemes for cryptographic implementations. Following our first application of this approach to AES [RP10] and its generalization to any SPN-like block ciphers [CGP+12], many subsequent works have followed a similar approach to design masking schemes for various algorithms and contexts. In relation of the issue of refreshing and tight probing security discussed in Section 3.4.3, a particular line of works has focused on secure masking composition, see in particular [CPRR14, BBD+16, BGR18, CS20]. Another line of works has focused on the efficient s-box decomposition and different masking operations, see in particular [RV13, CRV14, CPRR15, GR16, GRVV17], while dedicated designs of cryptographic primitives amenable to high-order masking have been proposed in [PRC12, GGNS13, GLSV15, BBB+20, GJK+20]. Alternative probing-secure multiplication gadgets have further been investigated in [BBP+16, BBP+17, KR18], and probing security has been extended to capture glitches with the *robust probing model* [FGP+18, CS21a]. Another important line of works has focused on building tools to formally verify the probing security of masking gadgets, see in particular [BBD+15, Cor18, BGI+18, BBC+19, KSM20, CFOS21, BK21]. Efficient low-level masked implementations and their automatic generation have further been investigated in [MOPT12, WVGX15, GR17, JS17, GJRS18, BDM+20].

This (non-exhaustive) list of works witnesses that the field of probing-secure masked implementations has been thoroughly investigated and has reached a certain maturity. Today, several tools have been developed for the generation of optimized and formally-verified masked implementations achieving some levels of provable security (in the probing model). From this state of affair, one of the main challenge that remains to be addressed is to close the gap between the probing model abstraction and the physical reality of side-channel leakage. This question is explored in the next chapter with the introduction of a more realistic leakage model, and in the rest of this thesis with the design of secure masking schemes in this model.

Chapter 4

The noisy leakage model

Contents

4.1. Introduction

We have seen in the previous chapter how to design masking schemes achieving arbitrary security orders in the probing model. However this model (as well as other existing leakage models) does not give full satisfaction to capture the physical reality of power and electromagnetic leakages. On the other hand, the later leakages are often *noisy* in practice and the soundness of masking in the presence of noise was formally argued in the seminal work of Chari, Jutla, Rao and Rohatgi in [CJRR99]. In order to capture this intuition in a more general leakage model, we formalized the *noisy leakage model* in a joint work with Prouff at Eurocrypt 2013 [PR13]. Compared to the univariate Gaussian model considered by Chari *et al.*, our model encompasses any type of (noisy) leakage distribution. In this work, we further demonstrated the soundness of masking in this more general leakage model and exhibited a proof of security (under some assumptions) for a masked implementation in this model. Although our proof suffered some limitations, such as the necessity of a leakage-free component and a linear scaling of the amount of noise, these limitations could be overcome in follow-up works. In particular, a major result due to Duc, Dziembowski and Faust [DDF14] subsequently shown that the noisy leakage security of an implementation could be reduced to its (region) probing security.

This chapter survey our definition of the noisy leakage model. After motivating the necessity of this model in Section 4.2, we recall its formal definition and discuss it relevance in Section 4.3. We then provide some useful bounds from our initial paper [PR13] in Section 4.4 and overview the Duc-Dziembowski-Faust reduction in Section 4.5.

4.2. Motivation

As overviewed in the previous chapter, many works have focused on designing masking schemes achieving formal security in the t-probing model. In this model, the adversary gets perfect leakage of exactly t intermediate variables in the computation. Although such a model enjoys some level of practical relevance (see discussion in Section 3.2) it does not give full satisfaction. In practice a side-channel adversary gets leakage traces which contain several leakage points from all the intermediate variables processed in the computation. Such an adversary is clearly out of the scope of the t-probing model. A particular shortcoming of the probing model is to fail to capture attacks known as *multivariate* or *horizontal* side-channel attacks in which the adversary take advantage of repeated manipulations of a variable (*e.g.* a share) in order to average the leakage noise. An implementation might be perfectly t-probing secure (with possibly high t) while being practically insecure against this type of attacks. We therefore need a model considering some leakage for all the intermediate variables in order to capture this type of attacks.

A pioneering work towards the formalization of leaking computation was the *physically observable cryptography* framework introduced by Micali and Reyzin in [MR04]. In particular, they formalize the assumptions that a cryptographic device can at least keep some secrets and that *only computation leaks information* (OCLI). In this setting, a cryptographic computation is split into a sequence of elementary operations that each leaks information on the accessed part of the device state. A few years later, Dziembowski and Pietrzak introduced the *leakage resilient cryptography* model [DP08] (inspired from the *bounded retrieval model* [DLW06]) in which the leakage function f is assumed to have a bounded range (*i.e.* taking values in $\{0,1\}^\lambda$ for some parameter λ). In this model, an operation taking an intermediate variable $x \in \{0,1\}^\ell$ as input leaks $f(x) \in \{0,1\}^\lambda$ with $\lambda < \ell$. The work of Dziembowski and Pietrzak had a huge impact and many follow-up works focused on the design of leakage resilient cryptographic primitives in this model as well as generic circuit compilers (see for instance [DP08, Pie09, DP10, DF12, GR12]).

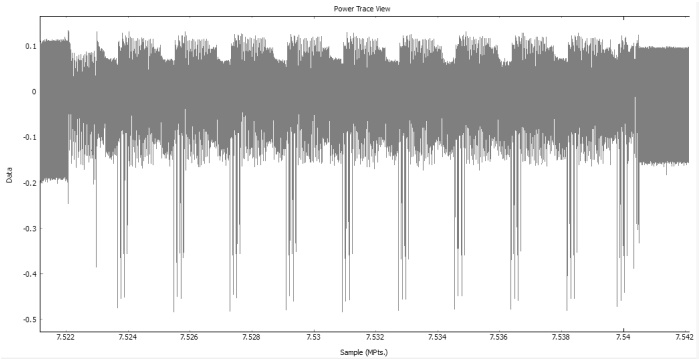

Figure 4.1.: A power consumption trace of an AES computation.
(Source: ChipWhisperer tutorial – `https://wiki.newae.com/`).

This model might seem conservative in terms of security since it encompasses leakage functions with complex structures. Unfortunately, this model also fails in capturing the physical reality of power and electromagnetic leakages for which the leakage is usually substantially larger than the secret state (but hopefully does not contain its entire entropy). This is illustrated on Figure 4.1 which depicts a power consumption trace corresponding to an AES encryption. This trace is composed of $\sim$20.000 points each encoded on several bits. Although the information could be compressed, it would hardly fit into less that 256 bits (the input size of AES counting the plaintext and the key), thus invalidating the $\lambda < \ell$ constraint of the bounded range leakage model. Moreover, such leakage traces can be made more accurate by increasing the sampling rate of the measurement equipment, resulting in even heavier traces. This further holds at a smaller granularity: a power or electromagnetic trace

corresponding to a single CPU instruction might also be made of several points and clearly exceed the size of the underlying intermediate variable.

In contrast, power and electromagnetic leakages are known to be noisy and several *hiding* techniques exist to increase this amount of noise, see for instance [CCD00, SÖP04, PM05, HOM06, MOP07, MSQ07, CK10]. This motivates the definition of a leakage model characterizing this noisy feature. While the pioneering work of Chari, Jutla, Rao and Rohatgi on the soundness of masking [CJRR99] (see Section 3.2.3) already considered noisy leakage, it was limited to a very narrow type of leakage functions of the form $f : x \in \{0,1\} \mapsto x + \mathcal{N}(0,\sigma^2)$ (*i.e.* with binary input range, univariate leakage, and Gaussian noise). In the following, we present the noisy leakage definition that we proposed in [PR13] which encompasses *any* leakage distribution. The leakage function f, which can take any form, is characterized through a single parameter (the *bias*) which captures the amount of noise in the leakage.

4.3. Noisy leakage definition

4.3.1. Intuition

Our noisy leakage model is a specialization of the physically observable cryptography framework [MR04] (and the OCLI assumption) with leakage functions belonging to the class of noisy leakage functions (as formally defined hereafter). In this model, every step of the processing reveals a leakage function of the touched part of the device state. The computation is split into several *elementary calculations* (in practice, a sequence of few CPU instructions) that each accesses a subpart x of the device state and leaks a function of x. Starting from the observation that masking is sound when combined with noise and that many practical solutions exist to amplify leakage noise, we assume the leakage functions to be *noisy*. The noisy feature of a leakage function f is captured by assuming that an observation of $f(x)$ only implies a *bounded bias* in the probability distribution of x. Namely the statistical distance between the distributions $\Pr(x)$ and $\Pr\big(x|f(x)\big)$ is bounded by some *noise parameter* δ.

Remark 3. We note that a generalization of the bounded range leakage model exists for which the loss in terms of min-entropy resulting from the leakage is bounded by λ, that is $\mathrm{H}_\infty(X|f(X)) - \mathrm{H}_\infty(X) \leqslant \lambda$.[1] This generalization introduced in [NS09] has the advantage of being more general (it encompasses bounded-range leakage functions) and it does not suffer from the limitations exposed above. Moreover, it was recently shown that this model is essentially equivalent to the bounded range model [BFO+21]. We note that this model is sometimes refer to as the *noisy leakage model* in the literature, because it does capture a form of noisy leakage. However, relying on the min-entropy still makes the underlying leakage assumption strong and imply a separation with our noisy leakage model (as recently demonstrated in [BFO+21]).

4.3.2. Formal definition

A leaking computation is modeled by a sequence of *elementary calculations* $(g_i)_i$ accessing a common memory called *internal state*. Each elementary calculation reads its input and writes its output on the internal state. When processed on some input x, an elementary calculation g_i reveals $f_i(x)$ to the adversary for some *noisy leakage function* f_i. A noisy leakage function is defined as a function that takes two arguments: the value x held by the accessed part of the internal state and a random string ρ long enough to model the leakage noise. Each execution leaks the values $\big(f_i(x_i,\rho_i)\big)_i$ where the x_i's are the successive intermediate values (from the internal state) in input of the elementary calculations g_i's and ρ_i's are fresh random strings. We stress that all the ρ_i's involved in successive executions are uniformly and independently drawn (*independent noise assumption*).

For the sake of simplicity, we shall omit the random string parameter, which leads to the notation $f_i(x)$ where x is the accessed value. Note that $f_i(x)$ is not the result of a function but it can be seen

[1] Here $\mathrm{H}_\infty(X|f(X))$ is the *average* conditional min-entropy: $\mathrm{H}_\infty(X|f(X)) = \sum_y \Pr(f(X)=y)\mathrm{H}_\infty(X|f(X)=y)$.

as the output of a probabilistic algorithm. In particular, $f_i(x)$ can take several values with a given probability distribution, and can therefore be considered as a random variable. The noisy property of f is captured by assuming that the bias introduced in the distribution of a uniform random variable X given the leakage $f(X)$ is bounded. This is formalized in the next definition:

Definition 12 (Noisy Function)**.** *Let $\mathcal{X}$ be a finite set and let $\delta \in \mathbb{R}$. A δ-noisy leakage function f on $\mathcal{X}$ is a function of domain $\mathcal{X} \times \{0,1\}^{|\rho|}$ for some $|\rho| \in \mathbb{N}$ such that*

$$\beta(X|Y) := \sum_{y \in \mathrm{Range}(f)} \Pr(Y = y) \cdot \Delta(X; (X \mid Y = y)) \leqslant \delta \ , \tag{4.1}$$

where Δ is a statistical distance measure, X is a uniform random variable over $\mathcal{X}$ and where $Y = f(X, R)$ for a uniform random variable R over $\{0,1\}^{|\rho|}$.

On the notion of statistical distance. The above definition depends on the notion of statistical distance. In our original definition [PR13], we suggested to use a statistical distance based on the Euclidean norm (or L_2 norm), which is:

$$\Delta_2(X_1; X_2) := \Big(\sum_{x \in \mathcal{X}} (\Pr(X_1 = x) - \Pr(X_2 = x))^2 \Big)^{\frac{1}{2}}$$

In a follow-up work [DDF14], Duc, Dziembowski and Faust suggested to use L_1 norm, normalized by $\frac{1}{2}$, which is:

$$\Delta_1(X_1; X_2) := \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr(X_1 = x) - \Pr(X_2 = x)|$$

This choice has the advantage to match the standard definition of statistical distance used in cryptography and it offers additional advantages discussed in [DDF14]. Let us denote by β_1 and β_2 the bias measures respectively induced by the statistical distances Δ_1 and Δ_2. From the L_1-L_2 norm inequality, these notions are related as follows:

$$\frac{1}{2}\, \beta_2(X|Y) \leqslant \beta_1(X|Y) \leqslant \frac{1}{2} \sqrt{|\mathcal{X}|}\, \beta_2(X|Y) \ . \tag{4.2}$$

We deduce that a δ_1-noisy leakage function w.r.t Δ_1-distance, is also a δ_2-noisy leakage function w.r.t Δ_2-distance, with $\delta_2 = (\sqrt{|\mathcal{X}|}/2) \cdot \delta_1$. On the other hand, a δ_2-noisy leakage function w.r.t Δ_2-distance, is also a δ_1-noisy leakage function w.r.t Δ_1-distance, with $\delta_1 = 2\delta_2$. We note that for a small definition space $\mathcal{X}$, the two notions are essentially equivalent.

It was further suggested by Prest, Goudarzi, Martinelli and Passelègue in [PGMP19] to use a statistical distance notion based on the *relative error*, specifically:

$$\Delta_{\mathrm{RE}}((X \mid Y = y); X) = \max_{x \in \mathcal{X}} \left| \frac{\Pr(X = x \mid Y = y)}{\Pr(X = x)} - 1 \right| \ . \tag{4.3}$$

We note that the above notion of distance is not symmetric and the order of the argument matters: $\Delta_{\mathrm{RE}}((X \mid Y = y); X) \neq \Delta_{\mathrm{RE}}(X; (X \mid Y = y))$. Noisy functions based on this distance are referred to as *average relative error* (ARE) noisy leakage functions in [PGMP19] since the relative error in Equation 4.3 is averaged over the distribution of the leakage Y according to Definition 12. The authors of [PGMP19] argue that using the above notion does not imply a stronger assumption on the leakage in practice, compared to the definitions based on L_1 and L_2 norms, whereas it has some advantages for tight security proofs.

On the choice of the uniform distribution. For the above definition of noisy leakage functions to be sound, we need to specify the distribution of X while bounding $\beta(X|f(X))$, and the uniform distribution is a natural choice. Of course, this does not constrain the leakage function to only apply

to uniformly distributed inputs. For a non-uniform variable X and a δ_1-noisy leakage function f (for Δ_1-distance), we obtain the following bound of the bias:[2]

$$\beta_1(X|f(X)) \leqslant |\mathcal{X}|\,\beta_1(U_{\mathcal{X}}|f(U_{\mathcal{X}})) = |\mathcal{X}|\,\delta_1\ , \tag{4.4}$$

where $U_{\mathcal{X}}$ denotes a uniform random variable on $\mathcal{X}$. If f is δ_2-noisy leakage function for the Δ_2-distance, using Equation 4.2, this bound translates to

$$\beta_2(X|f(X)) \leqslant |\mathcal{X}|^{\frac{3}{2}}\,\beta_2(U_{\mathcal{X}}|f(U_{\mathcal{X}})) = |\mathcal{X}|^{\frac{3}{2}}\,\delta_2\ . \tag{4.5}$$

4.3.3. Discussion

Practical relevance. Our model captures power and electromagnetic leakages since it does not impose any restriction on the form of the leakage distribution. In all generality, an elementary calculation processing an value x gives rise to a leakage trace $Y(x)$ which is a multivariate random variable (a.k.a random vector) following a distribution whose parameters depend on x. In most practical contexts, this distribution is well approximated by a multivariate Gaussian $\mathcal{N}(\boldsymbol{m}_x, \Sigma_x)$ for some parameters $\boldsymbol{m}_x$ (mean vector) and Σ_x (covariance matrix). Such parameters can be inferred in practice through a profiling of the device, from which we obtain the bias $\beta(X|Y)$ of the leakage $Y(x)$ by evaluating Equation 4.1. Of course, this bias should be small enough if we aim to build a secure implementation tolerating this leakage but this constraint matches a practical requirement: the noise in the leakage should be high enough to prevent a side-channel adversary from recovering the processed variable x with overwhelming probability.

We still need to stress that our model makes two implicit assumptions which might not be verified in practice without further care:

- *Data isolation.* The OCLI assumption implies that a leakage $f_i(x_i)$ corresponding to the elementary calculation $g_i(x_i)$ only takes as input the touched part of the state x_i and not the previously accessed parts of the state: ..., x_{i-2}, x_{i-1}. In other words, the leakage is assumed to respect some *data isolation* between successive elementary calculations. However, physical effects exist which are likely to break this implicit assumption. In particular, transitions occurring on memory buses or CPU registers between a previously processed value x_{prev} and the current one x usually leak some information correlated to $x_{prev} \oplus x$, which clearly invalidates the data isolation principle. At the hardware level, glitches further make the leakages of different successive gates mutually dependent of their respective inputs [MPG05]. At the software level, the CPU synchronization tackles the issue of glitches (which can further be avoided in hardware by the addition of synchronization elements. On the other hand, one should take a special care to transitions that may occur at many places in the CPU and somehow enforce data isolation for our model to be valid.
- *Independent noise assumption.* Our model assumes an independence of the leakage noises from the successive elementary calculations. Formally, the random tape ρ_i in each $f_i(x_i, \rho_i)$ is sampled as a fresh uniform string. Though in practice, if one would cut a leakage trace into several subtraces corresponding to successive elementary calculations, the noises in the successive subtraces would hardly be mutually independent. It is indeed known that correlation exists between successive leakage points, which makes multivariate statistics particularly useful for side-channel attacks [CRR03].

Computation granularity. In the OCLI model, a computation is divided into several sub-computations and the leakage function applies to the input of each sub-computation. In leakage-resilient cryptography constructions [DP08, Pie09, DP10], such a sub-computation is usually a cryptographic

[2] To see this, note that $\beta_1(X|f(X)) = \sum_x \Pr(X = x)\delta_x \leqslant \sum_x \delta_x \leqslant |\mathcal{X}| \sum_x \Pr(U_{\mathcal{X}} = x)\delta_x$ where we let $\delta_x := \frac{1}{2}\sum_y |\Pr(Y = y) - \Pr(Y = y|X = x)|$.

primitive, such as a (weak) pseudo random function (*e.g.* the AES cipher in practice). In contrast, our noisy leakage model is more suited for a finer granularity: the computation of one cryptographic primitive is divided into several elementary calculations that each leaks a function of its input. In other words, while leakage-resilient cryptography addresses the issue of constructing secure protocols from a cryptographic primitive with limited leakage, our motivation with the noisy model is to address the provably secure implementation of cryptographic primitives composed of leaking elementary calculations.

RAM *vs.* circuit model. Note that in our description, we implicitly assumed that the indexes (or addresses) of the accessed parts of the internal state throughout the computation are fixed and data-independent. This notably captures a software program avoiding any data-dependent memory access (a desirable feature for cryptographic implementations). We stress that in terms of information leakage, this can equivalently be captured by a circuit model with *gate leakage* (each gate leaks a noisy leakage function of its inputs). Our model can be extended to deal with random access memory (RAM) programs in which the accessed part of the internal state may depend on the data (*e.g.* a table look-up). But in that case, one should also consider the *address leakage* (*i.e.* the leakage function should apply to both the address and the read/written data) in order to fully capture the information leakage of such implementations.

4.4. Some security bounds

We provide hereafter some useful security bounds for noisy leakages. Unless otherwise stated, these results consider our original definition of noisy leakage functions based on the Δ_2-distance (*i.e.* $\Delta = \Delta_2$ in Definition 12). The proofs of the following statements are provided in our paper [PR13] (see Appendix C).

4.4.1. Relation to mutual information

Our notion of bias can be thought of as a measure for the information of X contained in $f(X)$. We show in the next proposition that it provides an upper bound of the mutual information between X and $f(X)$.

Proposition 1. *Let X be a random variable uniformly distributed over a set $\mathcal{X}$. Let f be a δ-noisy leakage function. The mutual information between X and $f(X)$ satisfies*

$$\mathrm{MI}(X;Y) \leqslant \frac{|\mathcal{X}|}{\ln 2}\beta_2(X|f(X)) \leqslant \frac{\delta\,|\mathcal{X}|}{\ln 2}\ . \tag{4.6}$$

4.4.2. Noisy leakage of a shared variable

The next theorem provides a bound of the bias on a variable induced by the noisy leakage of a sharing of this variable.

Theorem 2. *Let X be a uniform random variable over some field $\mathbb{K}$, let $(X_1, \ldots, X_n)$ be a uniform sharing of X. Let $f_1, \ldots, f_n$ be δ-noisy leakage functions defined over $\mathbb{K}$. We have:*

$$\beta_2\big(X\big|f_1(X_1), \ldots, f_n(X_n)\big) \leqslant |\mathbb{K}|^{\frac{n}{2}}\delta^n\ .$$

The above theorem shows that the bias of X given the leakages on its shares decreases exponentially with the order n, provided that the initial bias is sufficiently low, namely lower than $1/\sqrt{|\mathbb{K}|}$. Theorem 2 can be seen as a generalization of the result of Chari *et al.* [CJRR99] to any type of noisy leakage function. It shows that masking provides exponential security in the presence of noisy leakage for any leakage distribution with sufficient noise, seeing the *amount of noise* as the inverse of the bias.

Our result can also be interpreted in terms of hardness of distinguishing leakage distributions, as considered in [CJRR99]. Let F be the randomized function mapping some $x \in \mathbb{K}$ to the random vector

$(f_1(X_1), \ldots, f_n(X_n))$ where $(X_1, \ldots, X_n)$ is a uniform sharing of x. From Theorem 2, F is a ε-noisy leakage function with $\varepsilon = |\mathbb{K}|^{\frac{n}{2}}\delta^n$. Now consider the game in which an adversary aims to distinguish $F(x_0)$ from $F(x_1)$ for two chosen values $x_0, x_1 \in \mathbb{K}$ (this is the game considered in [CJRR99] for $\mathbb{K}$ being $\mathbb{F}_2$ and for the specific case of univariate Gaussian leakages). We know that the advantage of such an adversary is upper-bounded by

$$\begin{aligned}\Delta_1(F(x_0), F(x_1)) &\leqslant \Delta_1(F(x_0), F(U_\mathbb{K})) + \Delta_1(F(U_\mathbb{K}), F(x_1)) \\ &\leqslant 2|\mathbb{K}|\beta_1(U_\mathbb{K}|F(U_\mathbb{K})) \\ &\leqslant 4|\mathbb{K}|\varepsilon = 4|\mathbb{K}|^{1+\frac{n}{2}}\delta^n \ .\end{aligned}$$

where $\Delta_1(F(x_i), F(U_\mathbb{K})) \leqslant |\mathbb{K}|\beta_1(U_\mathbb{K}|F(U_\mathbb{K}))$ is obtained from Equation 4.4. We hence get a negligible distinguishing advantage as the masking order grows.

4.4.3. Noisy leakage of a repeated variable

The next theorem deals with the case of repeated leakages on a variable X.

Theorem 3. *Let X be a uniform random variable defined over a finite set $\mathcal{X}$. Let $f_1, \ldots, f_t$ be δ-noisy leakage functions defined over $\mathcal{X}$ with $\delta \leqslant \alpha/(t|\mathcal{X}|)$ for some $\alpha \in [0, 1]$. Then we have:*

$$\beta_2(X|f_1(X), f_2(X), \ldots, f_t(X)) \ \leqslant \ \left(\left(\frac{\mathrm{e}^\alpha - 1}{\alpha}\right)t + \mathrm{e}^\alpha\right)\delta \ .$$

The bound in Theorem 3 shows that the bias of X given t leakages increases linearly with t. A requirement is that the bias given a single leakage, namely δ, is at least t times lower than $\frac{1}{|\mathcal{X}|}$ or less, namely $\delta \leqslant \frac{\alpha}{t|\mathcal{X}|}$ for some $\alpha \in [0, 1]$. Then the bias of X given t leakages is smaller that $\lambda(t) \cdot \delta$ where λ is an affine function. The value α provides a trade-off between the constraint on δ and the coefficients of λ. If $\alpha = 1$ then $\lambda(t) = (\mathrm{e} - 1)t + \mathrm{e} \approx 1.72\,t + 2.72$, while $\lambda(t)$ tends towards $t + 1$ as α approaches 0.

4.5. From probing to noisy leakage security

One year after our formalization of the noisy leakage model, Duc, Dziembowski, and Faust accomplished a major step towards the design of provably secure schemes in this model [DDF14]. They could show that proving the security of a scheme in the noisy leakage model is essentially equivalent as proving its security in the *random probing model*, which is conceptually much simpler. Thanks to the Chernoff bound, random probing security can further be obtained from the simpler *region probing security* (see Section 2.5 for the formal definition of these probing security notions). We outline the DDF reduction hereafter.

The random probing model was first used in [ISW03, Ajt11] and formalized by Duc *et al.* in [DDF14]. It can be seen as a restriction of the noisy leakage model in which leakage functions leak their entire input with a given probability. These *δ-random-probing functions* (*δ-identity functions* [DDF14]) are formally defined as follows.

Definition 13. *The* p-random-probing function *is the randomized function $\phi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ satisfying*

$$\phi(x) = \begin{cases} \perp & \text{with probability } 1 - p \\ x & \text{with probability } p \end{cases} \tag{4.7}$$

Remark 4. Such a function is a special case of δ-noisy leakage function with $\delta = p$ (for the Δ_1-distance). To be tighter, a p-random-probing function is a δ-noisy leakage function with $\delta = p\,(1 - \frac{1}{|\mathcal{X}|}) \leqslant p$. For the Δ_2-distance notion of leakage function, a p-random-probing function is a δ-noisy leakage function with $\delta = p\,(2 - \frac{3}{|\mathcal{X}|} + \frac{1}{|\mathcal{X}|^2}) \leqslant 2p$.

The key result of Duc, Dziembowski, and Faust [DDF14] is to show that every noisy leakage function f can be expressed as a composition $f = f' \circ \phi$ where ϕ is a random-probing function. This enables to reduce noisy-leakage security to random-probing security.

Lemma 1 ([DDF14])**.** *Let $f : \mathcal{X} \to \mathcal{Y}$ be a δ-noisy leakage function with $\delta < \frac{1}{|\mathcal{X}|}$. There exists a p-random-probing function $\phi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ and a randomized function $f' : \mathcal{X} \cup \{\perp\} \to \mathcal{Y}$ such that for every $x \in \mathcal{X}$ we have*

$$f(x) = f'(\phi(x)) \quad \text{and} \quad p \leqslant \delta \cdot |\mathcal{X}| \ . \tag{4.8}$$

The reduction. Consider a computation giving rise to a sequence of intermediate variables $x_1, \ldots, x_s$. This computation is ε-secure against δ-noisy leakage if there exists a simulator $\mathsf{Sim}_{\mathrm{nl}}$ such that

$$\Delta_1(\mathsf{Sim}_{\mathrm{nl}}(), (f_1(x_1), \ldots, f_s(x_s))) \leqslant \varepsilon \ , \tag{4.9}$$

for any set of δ-noisy leakage functions $\{f_i\}$. This computation is ε-secure against p-random probing leakage if there exists a simulator $\mathsf{Sim}_{\mathrm{rp}}$ such that

$$\Delta_1(\mathsf{Sim}_{\mathrm{rp}}(), (\phi(x_1), \ldots, \phi(x_s))) \leqslant \varepsilon \ , \tag{4.10}$$

where ϕ is the p-random-probing function.

We can show that any computation which is ε-secure against p-random probing leakage is also ε-secure against δ-noisy leakage with $\delta = p/|\mathcal{X}|$. The reduction consists in constructing $\mathsf{Sim}_{\mathrm{nl}}$ from $\mathsf{Sim}_{\mathrm{rp}}$ as follows. $\mathsf{Sim}_{\mathrm{rp}}$ is first called which outputs a vector $(y_1, \ldots, y_s)$. Then $\mathsf{Sim}_{\mathrm{nl}}$ simply outputs $(f'_1(y_1), \ldots, f'_s(y_s))$ where the f'_i's are defined such that $f_i = f'_i \circ \phi$ which is possible by Lemma 1. By definition, $(y_1, \ldots, y_s)$ is ε-close to $(\phi(x_1), \ldots, \phi(x_s))$ which implies that $(f'_1(y_1), \ldots, f'_s(y_s))$ is ε-close to $(f'_1(\phi(x_1)), \ldots, f'_s(\phi(x_s)))$ which is in turn identically distributed as $(f_1(x_1), \ldots, f_s(x_s))$.

To sum-up we have:

1. δ-noisy leakage security $\Rightarrow$ p-random probing security, for any $p \leqslant \delta/2$,
2. p-random probing security $\Rightarrow$ δ-noisy leakage security, for any $\delta \leqslant p/|\mathcal{X}|$,

where $\mathcal{X}$ is input space of elementary calculations decomposing the computation. Some ways to mitigate the loss factor $1/|\mathcal{X}|$ have been suggested in [DFS15, GJR18, PGMP19].

Remark 5. In the original version of Lemma 1, the authors make an additional (weak) assumption on f so that f' is efficiently computable in the above simulation.[3] We note that this requirement is only necessary for computational security (which requires the simulation to be efficient) and can be relaxed for statistical security (considering adversaries with unlimited computational power).

One step further. In the random-probing model, the total number of leaking operations can be statistically bounded using the Chernoff bound as suggested in [ISW03, DDF14].

Theorem 4 (Chernoff Bound [Che52])**.** *Consider the δ-random probing leakage of a computation composed of s elementary calculations. The probability that this leakage reveals more than $\ell > \delta s$ variables is upper bounded by*

$$\psi(\ell, s) = \exp\Big(- \frac{(\ell - \delta\, s)^2}{\ell + \delta\, s} \Big) \ . \tag{4.11}$$

Taking $\ell = 3\delta s$ we get

$$\psi(\ell, s) = \exp\Big(- \frac{1}{3} \delta s \Big) \ . \tag{4.12}$$

[3] Precisely, they require that f is *efficiently decidable* by which they mean that f can be computed in polynomial time as well as $\Pr(f(x) = y)$ for any pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Using the above theorem, any computation ε_{p}-secure against t-probing leakage is further $\varepsilon_{\mathrm{rp}}$-secure against p-random probing leakage with

$$p = \frac{t}{3s} \quad \text{and} \quad \varepsilon_{\mathrm{rp}} = \varepsilon_{\mathrm{p}} + \exp\Big(-\frac{1}{3}\delta s\Big) .$$

Let us stress that a direct application of the Chernoff bound to a t-probing secure circuit $\widehat{C}$ results in a p-random probing security with $p = \Theta(t/|\widehat{C}|)$. Namely, the tolerated leakage probability linearly scales down with the size of $\widehat{C}$, which is not satisfactory. This is the motivation to consider *region probing security* which requires that each gadget G in the circuit can tolerate up to t probes (see formal definition in Section 2.5). We then get a p-random probing security with $p = \Theta(r)$ where $r = t/|G|$ is known as the *probing rate*, which is not affected by the total size of $\widehat{C}$ and hence provides a much tighter reduction. We thus have:

$$\begin{array}{l} r\text{-region probing security} \\ \quad \Rightarrow p\text{-random probing security, with } p = \Theta(r) \\ \quad\quad \Rightarrow \delta\text{-noisy leakage security, with } \delta = \Theta(p) = \Theta(r) \end{array}$$

We observe that, up to constant factors, the probing rate r translates the leakage probability p, which in turns translates to the bias δ in the noisy model. For this reason, we shall jointly refer to those parameters as the tolerated *leakage rate* of the implementation. Ideally, we would like this leakage rate to be constant and as close as possible to 1.

4.6. Conclusion and related works

In this chapter, we have introduced a formal leakage model which arguably captures the physical reality of power an electromagnetic leakages. In a remarkable follow-up work [DDF14] overviewed in Section 4.5, Duc, Dziembowski and Faust have shown that the security of an implementation in this noisy model can be reduced to its security in the conceptually simpler random probing and region probing models. This reduction motivates the design of new masking schemes achieving these stronger variants of probing security. This is explored in the next chapters of the present thesis, by first focusing on secure masking composition in these models and then proposing actual masking schemes achieving (quasi)constant leakage rates. While Duc *et al.*'s work unifies our noisy model and probing-like models, it was recently shown in [BFO+21] that our noisy model is separated from the bounded range leakage model [DP08] as well as the "noisy" generalization of the bounded range model [NS09] (which bounds the loss in min-entropy due to the leakage). Another interesting follow-up work due to Prest, Goudarzi, Martinelli and Passelègue [PGMP19] has investigated alternative definitions of the bias metric for noisy leakage functions based on the Rényi divergence. While the noisy leakage model introduce in this chapter and its generalizations [DDF14, PGMP19] have reduced the gap between the theory and practice of side-channel security, more research is still to be made to close this gap. In particular, more investigation is needed to evaluate the noisy leakage functions (and their bias) observed in practice for common elementary calculations on a wide range of devices. Another related research direction is to relax and/or practically enforce the ideal hypothesis implicitly made by the noisy leakage model in terms of data isolation and noise independence.

Part III.

Secure masking composition

Chapter 5

Secure composition in the region probing model

Contents

5.1. Introduction

As exposed in the previous chapters, the probing security model is widely used to formally prove the security of masking schemes. Whenever a masked implementation can be proven secure in this model with a reasonable *leakage rate*, it is also provably secure in the realistic noisy leakage model introduced in Chapter 4. In a joint work with Goudarzi, Prest and Vergnaud [GPRV21], we have introduced a new framework for the composition of probing-secure circuits. Specifically, we have introduced the security notion of *input-output separation* (IOS) for a refresh gadget. From this notion, one can easily compose gadgets satisfying the classical probing security notion –which does not ensure composability on its own– to obtain a *region probing secure* circuit. Such a circuit is secure against an adversary placing up to t probes in each gadget composing the circuit, which ensures a tight reduction to the more realistic noisy leakage model. This chapter presents the IOS composition framework. We first introduce the IOS notion as well as the secure composition result in Section 5.2. In Section 5.3, we then show that a variant of the Battistello, Coron, Prouff and Zeitoun (BCPZ) refresh gadget [BCPZ16a] achieves IOS security in quasilinear complexity $\mathcal{O}(n \log n)$.

5.2. Composition through input-output separation

5.2.1. Input-output separation

We introduce hereafter the *input-output separation* (IOS) security notion for a refresh gadget, which is a variant of the *input-output linear separability* that we originally introduced [GJR18]. We first define

the notion of uniformity for a gadget which will be a requirement for the IOS notion. These notions are introduced for $\boldsymbol{v}$-linear sharings (for which the encoding relation is $x = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$) and $\boldsymbol{v}$-gadgets (which process $\boldsymbol{v}$-linear sharings) as formally introduced in Section 2.3.

Definition 14 (Uniformity)**.** *Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$. A $\boldsymbol{v}$-refresh gadget G is* uniform*, if for every $\boldsymbol{x} \in \mathbb{K}^n$, the output $G(\boldsymbol{x})$ is a uniform $\boldsymbol{v}$-linear sharing of $\langle \boldsymbol{v}, \boldsymbol{x} \rangle$.*

In the following, we shall say that a pair of vector $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathbb{K}^n)^2$ is *admissible* for a gadget G if there exists a random tape $\boldsymbol{\rho}$ (*i.e.* an assignment of the random gates' outputs) such that $\boldsymbol{y} = G^{\boldsymbol{\rho}}(\boldsymbol{x})$. For an admissible pair $(\boldsymbol{x}, \boldsymbol{y})$ and a set $\mathcal{W}$ of wires of G, the wire assignment distribution of G in $\mathcal{W}$ *induced* by $(\boldsymbol{x}, \boldsymbol{y})$, denoted $\mathsf{AssignWires}(G, \mathcal{W}, \boldsymbol{x}, \boldsymbol{y}) \in \mathbb{K}^{|\mathcal{W}|}$, is the random vector $\mathsf{AssignWires}(G, \mathcal{W}, \boldsymbol{x})$ (as introduced in Section 2.2) constrained to $\boldsymbol{y} = G^{\boldsymbol{\rho}}(\boldsymbol{x})$, *i.e.* the wire assignment distribution obtained for a uniform drawing of $\boldsymbol{\rho}$ among $\{\boldsymbol{\rho}\,;\, G^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x}) = \boldsymbol{y}\}$. We note that for a uniform $\boldsymbol{v}$-refresh gadget, an admissible pair is any $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathbb{K}^n)^2$ such that $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = \langle \boldsymbol{v}, \boldsymbol{y} \rangle$.

Definition 15 (Input-Output Separation)**.** *Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$ and let G be a $\boldsymbol{v}$-refresh gadget. G is said* t-input-output separative (t-IOS)*, if it is uniform and if for every distribution $\mathcal{D}_{(\boldsymbol{x},\boldsymbol{y})}$ sampling admissible pairs for G and for every set of wires $\mathcal{W}$ of G with $|\mathcal{W}| \leqslant t$, there exists a (two-stage) simulator $\mathsf{Sim} = \big(\mathsf{Sim}_1, \mathsf{Sim}_2\big)$ such that*

1. *given $\mathsf{Sim}_1(\mathcal{W}) = (I, J)$ where $I, J \subseteq [n]$, with $|I| \leqslant |\mathcal{W}|$ and $|J| \leqslant |\mathcal{W}|$,*
2. *and for $(\boldsymbol{x}, \boldsymbol{y}) \leftarrow \mathcal{D}_{(\boldsymbol{x},\boldsymbol{y})}$, we have:*

$$\mathsf{Sim}_2(\mathcal{W}, \boldsymbol{x}|_I, \boldsymbol{y}|_J) \overset{id}{=} \mathsf{AssignWires}(G, \mathcal{W}, \boldsymbol{x}, \boldsymbol{y})$$

A $\boldsymbol{v}$-refresh gadget is simply said to be IOS *if it is n-IOS.*

5.2.2. Composition intuition

The intuition behind the IOS notion can be understood as follows. Any probing leakage from an IOS refresh gadget can be simulated given a subset of its input shares and output shares. We can therefore reduce the standard region probing security game to a game in which the refresh gadget does not leak anything but its surrounding gadgets leak more. The uniformity property then implies that the leakages from two gadgets separated by a refresh gadget are mutually independent. One can then achieve a perfect simulation of the full leakage through independent simulations of the *separated* leakages from the two gadgets.

This is illustrated on Figure 5.1. The full probing leakage $(\boldsymbol{w}_1, \boldsymbol{w}_\mathsf{R}, \boldsymbol{w}_2)$ can be simulated from $(\boldsymbol{w}_1, \boldsymbol{y}|_I, \boldsymbol{y}'|_J, \boldsymbol{w}_2)$. Moreover, the refresh uniformity implies that, given x, the separated leakages $(\boldsymbol{w}_1, \boldsymbol{y}|_I)$ and $(\boldsymbol{w}_2, \boldsymbol{y}'|_J)$ are mutually independent. Therefore, if one can simulate $(\boldsymbol{w}_1, \boldsymbol{y}|_I)$ on the one hand and $(\boldsymbol{w}_2, \boldsymbol{y}'|_J)$ on the other hand, then one can simulate the full leakage.

5.2.3. Composition theorem

The IOS composition approach relies on standard circuit compilation with *full refreshing*, *i.e.* interleaving a refresh gadget in output of every operation gadget (see formal definition in Section 2.4). Specifically, we consider a set $\{G_g\}$ of base gadgets, one per type of gate g, as well as a refresh gadget G_R. The compilation process replaces each gate of type g by a gadget G_g followed by a refresh gadget G_R. Copy gadgets (which have fan-out 2) give rise to 2 refresh gadgets, one per output copy.

Let us first recall that a circuit composed of gadgets $\{G_i\}$ that jointly tolerates $\{t_i\}$ probes is r-region probing secure with $r = \min(t_i/|G_i|)$ (see Section 2.5 for a formal definition of region probing security). The following composition theorem shows that a standard circuit compiler based on $\{G_g\}$ and G_R achieves region probing security if G_R is IOS and if the gadgets $\{G_g\}$ are probing secure.

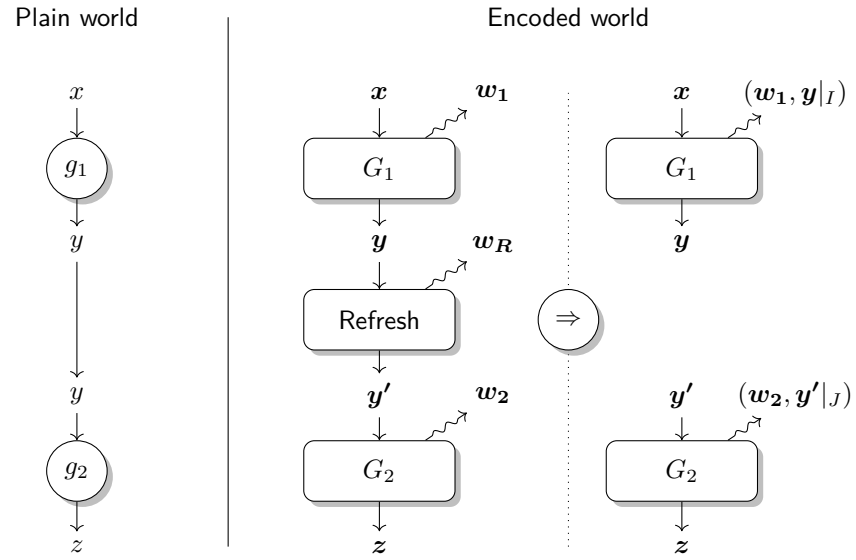

Figure 5.1.: Illustration of the IOS property.

Theorem 5 (IOS Composition)**.** *Let $\{G_g\}$ be t_g-probing secure and let G_{R} be t_{R}-IOS. Then the standard circuit compiler with full refreshing defined from the base gadgets $\{G_g\}$ and the refresh gadget G_{R} satisfies r-region probing security with*

$$r = \max_{t \leqslant t_{\mathsf{R}}} \min \left(\left(\frac{t_g - 3t}{|G_g|} \right)_g , \frac{t}{|G_{\mathsf{R}}|} \right) . \tag{5.1}$$

The proof follows the intuition given above. Note that for each operation (or copy) gadget, the sum of fan-in and fan-out is three. This implies that after applying the IOS property to replace internal probes of refresh gadgets by probes on input/output sharings, each operation gadget G_g has to tolerate $3t$ probes on its input/output shares (t per surrounding refresh gadget) plus its own probes. The number of its own probes can hence be at most $t_g - 3t$ if the gadget is t_g-probing secure, which leads to a probing rate of $(t_g - 3t)/|G_g|$ for this gadget. On the other hand the probing rate of the refresh gadget is of $t/|G_{\mathsf{R}}|$. One can then choose t in order to maximize the minimum rate among the different gadgets. The formal proof of Theorem 5 can be found in our paper [GPRV21] (see Appendix E).

5.2.4. Comparison with previous composition approaches

It is well-known that composition of probing secure gadgets is not always probing secure, which was a motivation to introduce refresh gadgets as discussed in Chapter 3. Stronger security definitions have been proposed to achieve secure composition in the probing model. In particular, the notion of *(strong) non-interference*, or (S)NI, was proposed in [BBD+16] and the notion of *probe isolating non-interference* (PINI) was also recently introduced in [CS20].

While the composition approaches considered in these previous works aim at achieving standard t-probing security for the composition, our aim here is to achieve region probing security, *i.e.* to tolerate some amount of probes in all the gadgets (rather than a total of t probes on the circuit). We provide a detailed comparison of the composition approaches in our paper [GPRV21] (see Appendix E). In particular, while SNI refresh gadgets composed with NI operation gadgets can provide region probing security, PINI gadgets are limited to standard probing security (but without requiring any refresh gadgets). In contrast our composition approach achieves region probing security by composing IOS refresh gadgets with any probing secure operation gadgets.

5.3. An input-output separative refresh gadget

5.3.1. BCPZ refresh gadget

Battistello, Coron, Prouff and Zeitoun (BCPZ) introduced in [BCPZ16a] (long version of [BCPZ16b]) a refresh gadget which achieves SNI in complexity $\mathcal{O}(n \log n)$. This refresh gadget is defined recursively as:

$$G_{\mathsf{R}}(\boldsymbol{x}) = \big(G_{\mathsf{R}}(\boldsymbol{x}_1 + \boldsymbol{r}) + \boldsymbol{s} \parallel G_{\mathsf{R}}(\boldsymbol{x}_2 - \boldsymbol{r}) - \boldsymbol{s}\big) \tag{5.2}$$

for any $\boldsymbol{x} = (\boldsymbol{x}_1 \parallel \boldsymbol{x}_2) \in \mathbb{K}^n$ with $n > 1$, with randomly sampled $\boldsymbol{r} \leftarrow \mathbb{K}^{n/2}$ and $\boldsymbol{s} \leftarrow \mathbb{K}^{n/2}$, and $G_{\mathsf{R}}(\boldsymbol{x}) = \boldsymbol{x}$ for $n = 1$. While this definition implicitly assumes that n is a power of 2, this refresh gadget is defined more generally for any $n \in \mathbb{N}$ (see description in [BCPZ16a]).

We show hereafter that a variant of this gadget –which is defined for any $\boldsymbol{v}$-linear sharing and which requires half the randomness of the original gadget– achieves the IOS property.

5.3.2. Proposed variant

The modified refresh gadget G_{R} is described in Algorithm 7. It consists in a simple generalization of BCPZ to the case of $\boldsymbol{v}$-linear sharings (for any vector $\boldsymbol{v} \in \mathbb{K}^n$) and which further saves half of the randomness compared to the original version. Moreover, this variant is used to generate an $\boldsymbol{v}$-linear sharing of 0, denoted $\boldsymbol{z}$, which is then added to $\boldsymbol{x}$ to refresh it.

Specifically, our IOS refresh gadget is defined as:

$$G_{\mathsf{R}}(\boldsymbol{x}) \mapsto \boldsymbol{y} = \boldsymbol{x} + \boldsymbol{z} \quad \text{with} \quad \boldsymbol{z} \leftarrow \mathsf{ZeroEncoding}(\boldsymbol{v}) \ , \tag{5.3}$$

where $\mathsf{ZeroEncoding}(\boldsymbol{v})$ outputs a fresh $\boldsymbol{v}$-linear sharing of 0 using a variant of BPCZ gadget applied to the all-0 vector. This variant is defined recursively as follows: for $n = 2$, it outputs $\boldsymbol{z} = (r, -r \cdot (v_1 v_2^{-1}))$ such that $\langle \boldsymbol{z}, \boldsymbol{v} \rangle = 0$ and for $n \geqslant 4$ a power of 2, $\mathsf{ZeroEncoding}$ is called recursively to produce two halves of the sharing (Steps 4-5) and a post-processing layer is applied to the whole sharing (Steps 6-9). Note that the original refresh gadget proposed in [BCPZ16b] makes use of an additional and similar pre-processing layer before the two recursive calls. It results that our variant is twice more efficient in terms of computation and randomness generation.

Algorithm 6 $\mathsf{ZeroEncoding}$ (variant of BCPZ)

Input: $\boldsymbol{v} = (v_1, \ldots, v_n)$
Output: $\boldsymbol{z} = (z_1, \ldots, z_n)$ such that $\langle \boldsymbol{z}, \boldsymbol{v} \rangle = 0$
1: **if** $n = 2$ **then**
2: $\quad r \leftarrow \mathbb{K}$
3: $\quad$ **return** $(r, -(v_1 v_2^{-1}) \cdot r)$
4: $(s_1, \ldots, s_{\frac{n}{2}}) \leftarrow \mathsf{ZeroEncoding}(v_1, \ldots, v_{\frac{n}{2}})$ ▷ Recursive call
5: $(s_{\frac{n}{2}+1}, \ldots, s_n) \leftarrow \mathsf{ZeroEncoding}(v_{\frac{n}{2}+1}, \ldots, v_n)$ ▷ Recursive call
6: **for** $i = 1, \ldots, \frac{n}{2}$ **do**
7: $\quad r_i \leftarrow \mathbb{K}$
8: $\quad z_i \leftarrow s_i + r_i$
9: $\quad z_{i+\frac{n}{2}} \leftarrow s_{i+\frac{n}{2}} - (v_i v_{i+\frac{n}{2}}^{-1}) \cdot r_i$

Algorithm 7 IOS refresh gadget

Input: $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{v} = (v_1, \ldots, v_n)$,
Output: $\boldsymbol{y} = (y_1, \ldots, y_n)$ such that $\langle \boldsymbol{y}, \boldsymbol{v} \rangle = \langle \boldsymbol{x}, \boldsymbol{v} \rangle$
1: $\boldsymbol{z} \leftarrow \mathsf{ZeroEncoding}(\boldsymbol{v})$
2: $\boldsymbol{y} \leftarrow \boldsymbol{x} + \boldsymbol{z}$

Let us denote $R(n)$, $A(n)$ and $M(n)$ the randomness complexity, the number of additions and the number of scalar multiplications of the ZeroEncoding algorithm for length-n linear sharing. We have $R(2) = 1$, $A(2) = 0$ and $M(2) = 1$ and $R(n) = 2R(\frac{n}{2}) + \frac{n}{2}$, $A(n) = 2A(\frac{n}{2}) + n$ and $M(n) = 2M(\frac{n}{2}) + \frac{n}{2}$ for all $n \geqslant 2$. By induction, we thus have for any $n \geqslant 2$, a power of 2,

$$R(n) = M(n) = \frac{n}{2}\log(n) \quad \text{and} \quad A(n) = n \log \frac{n}{2} \,. \tag{5.4}$$

We have n further additions in Algorithm 7.

5.3.3. Input-output separation

The following theorem states the IOS security of the above refresh gadget. The proof is provided in our paper [GPRV21] (see Appendix E).

Theorem 6. *The refresh gadget from Algorithm 7 is input-output separative.*

We thus obtain a quasilinear-complexity refresh gadget achieving the IOS property for any $\boldsymbol{v}$-linear sharings. We note that the leakage rate (*i.e.* the ratio $t/|G|$) tolerated by such a gadget is $\Theta(1/\log n)$ which is not constant but only *quasiconstant*.

5.4. Conclusion and related works

In this chapter, we have introduced a simple composition approach to obtain region probing security. The advantage of our approach resides in the fact that given a refresh gadget satisfying our IOS security property, all the other gadgets composing the scheme only require to satisfy the weak notion of probing security (as opposed to a stronger composition notion such as *e.g.* SNI). We have further shown that a variant of the BCPZ refresh gadget achieves IOS security in quasilinear complexity $\mathcal{O}(n \log n)$ and with a *quasiconstant* leakage rate $\Theta(1/\log n)$. In Chapter 7, we will present a specific instantiation of the IOS composition framework which results in a region probing secure scheme with quasilinear complexity overhead and *quasiconstant* leakage rate.

Chapter 6

Secure composition in the random probing model

Contents

6.1. Introduction

While many works have addressed the issue of secure masking composition in the probing model, until recently, no composition notions have been proposed for the random probing model. The latter is arguably more tricky to deal with since any set of wires in a gadget might leak with a given probability. At Crypto 2020, we filled this gap in a joint work with Belaïd, Coron, Prouff and Taleb by formilizing the notion of *random probing composability*. This notion is reminiscent to composition notions for the probing model, like (S)NI, but integrates the probabilistic nature of the random probing model.

This chapter presents this formalization. We first introduce in Section 6.2 the notion of *simulation with abort* and *simulation failure probability* which are at the core of our definition. Then we formally define the random probing composability (RPC) notion and state the associated composition theorem in Section 6.3. We further exhibit a relation between the RPC notion and the strong non-interference (SNI) notion which is widely used for secure composition in the probing model.

6.2. Background notions

6.2.1. Simulation with abort

As formally introduced in Section 2.5, a randomized circuit $\widehat{C}$ is (p, ε)-*random probing secure* (w.r.t. encoding Enc) if there exists a simulator Sim which satisfies

$$\mathsf{Sim}() \approx_\varepsilon \mathsf{AssignWires}(\widehat{C}, \mathsf{LeakingWires}(\widehat{C}, p), \mathsf{Enc}(\boldsymbol{x})) \ , \tag{6.1}$$

where $\boldsymbol{x} \leftarrow \mathcal{D}_{\boldsymbol{x}}$ (for any given distribution $\mathcal{D}_{\boldsymbol{x}}$) and where $\mathsf{LeakingWires}(\widehat{C}, p)$, the so-called leaking-wires sampler, outputs a set $\mathcal{W}$ constructed by including each wire from the circuit $\widehat{C}$ with probability p to $\mathcal{W}$ (with all the probabilities being mutually independent).

For our composition notion, we shall consider a particular simulation strategy called *simulation with abort* [AIS18]. In this approach, the simulator first calls the leaking-wires sampler to get a set $\mathcal{W}$ and then either aborts with probability ε or outputs the exact distribution of the wire assignment corresponding to $\mathcal{W}$. Formally, for any leakage probability $p \in [0, 1]$, the simulator Sim is defined as

$$\mathsf{Sim}() = \mathsf{Sim}_{\mathsf{AW}}(\mathsf{LeakingWires}(\widehat{C}, p)) \tag{6.2}$$

where $\mathsf{Sim}_{\mathsf{AW}}$, the *wire assignment simulator*, either returns $\perp$ (simulation failure) or a perfect simulation of the requested wires. More precisely, the experiment

$$\begin{aligned} \mathcal{W} &\leftarrow \mathsf{LeakingWires}(\widehat{C}, p) \\ out &\leftarrow \mathsf{Sim}_{\mathsf{AW}}(\widehat{C}, \mathcal{W}) \end{aligned}$$

leads to

$$\varepsilon = \Pr[out = \perp] \tag{6.3}$$

and

$$\left(out \mid out \neq \perp\right) \overset{id}{=} \left(\mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})) \mid out \neq \perp\right) .$$

It is not hard to see that if we can construct such a simulator $\mathsf{Sim}_{\mathsf{AW}}$ for a randomized circuit $\widehat{C}$, then the simulator Sim defined in Equation 6.2 satisfies Equation 6.1 and hence the circuit is (p, ε)-random probing secure.

6.2.2. Simulation failure probability

The probability in Equation 6.3 is called the *simulation failure probability*, which we characterize hereafter. We consider a randomized circuit $\widehat{C}$ composed of s wires labeled from 1 to s and a wire assignment simulator $\mathsf{Sim}_{\mathsf{AW}}$. For any sub-set $\mathcal{W} \subseteq [s]$ we denote by $\delta_{\mathcal{W}}$ the value defined as:

$$\delta_{\mathcal{W}} = \begin{cases} 1 & \text{if } \mathsf{Sim}_{\mathsf{AW}}(\mathcal{W}) = \perp, \\ 0 & \text{otherwise.} \end{cases}$$

The simulation failure can then be explicitly expressed as a function of p. Namely, we have $\varepsilon = f(p)$ with f defined for every $p \in [0, 1]$ by:

$$f(p) = \sum_{\mathcal{W} \subseteq [s]} \delta_{\mathcal{W}} \cdot p^{|\mathcal{W}|} \cdot (1 - p)^{s - |\mathcal{W}|} \ . \tag{6.4}$$

Letting c_i be the number of sub-sets $\mathcal{W} \subseteq [s]$ of cardinality i for which $\delta_{\mathcal{W}} = 1$, namely for which the simulation fails, we have $c_i = \sum_{|\mathcal{W}|=i} \delta_{\mathcal{W}}$ and hence Equation 6.4 simplifies to

$$f(p) = \sum_{i=1}^{s} c_i \cdot p^i \cdot (1 - p)^{s-i} \ \leqslant \ \sum_{i=1}^{s} c_i \cdot p^i \ . \tag{6.5}$$

We note that for any circuit $\widehat{C}$ achieving t-probing security, the values $\delta_{\mathcal{W}}$ with $|\mathcal{W}| \leqslant t$ are equal to zero which implies $c_1 = c_2 = \cdots = c_t = 0$. Moreover, by definition, the coefficients c_i satisfy the upper bound $c_i \leqslant \binom{s}{i}$ which leads to the following upper-bound for t-probing secure circuits:

$$f(p) \ \leqslant \ \sum_{i=t+1}^{s} \binom{s}{i} \cdot p^i \cdot (1 - p)^{s-i} \ \leqslant \ \sum_{i=t+1}^{s} \binom{s}{i} \cdot p^i$$

More generally the bound $c_i \leqslant \binom{s}{i}$ can be used to derive an upper bound of $f(p)$ given the knowledge of a limited number of low-degree coefficients c_i.

6.3. Random probing composability

6.3.1. Formal definition

We introduce hereafter the *random probing composability* notion for a gadget. In the following definition, for an n-share, ℓ-to-m gadget, we denote by $\boldsymbol{I}$ a collection of sets $\boldsymbol{I} = (I_1, \ldots, I_\ell)$ with $I_1 \subseteq [n]$, $\ldots$, $I_\ell \subseteq [n]$ where $n \in \mathbb{N}$ refers to the number of shares.

Definition 16 (Random Probing Composability)**.** *Let $n, \ell, m \in \mathbb{N}$. An n-share gadget $G : (\mathbb{K}^n)^\ell \to (\mathbb{K}^n)^m$ is (t, p, ε)-*random probing composable (RPC) *for some $t \in \mathbb{N}$ and $p, \varepsilon \in [0, 1]$ if there exists a deterministic simulator Sim_1 and a probabilistic simulator Sim_2 such that for every distribution $\mathcal{D}_{\boldsymbol{x}}$ over $(\mathbb{K}^n)^\ell$ and for every set collection $J_1 \subseteq [n]$, $\ldots$, $J_m \subseteq [n]$ of cardinals $|J_1| \leqslant t$, $\ldots$, $|J_m| \leqslant t$, the random experiment*

$$\begin{aligned} &\mathcal{W} \leftarrow \mathsf{LeakingWires}(G, p) \\ &\boldsymbol{I} \leftarrow \mathsf{Sim}_1(\mathcal{W}, \boldsymbol{J}) \\ &(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell) \leftarrow \mathcal{D}_{\boldsymbol{x}} \\ &\mathit{out} \leftarrow \mathsf{Sim}_2\big(\boldsymbol{x}_1|_{I_1}, \ldots, \boldsymbol{x}_\ell|_{I_\ell}\big) \end{aligned}$$

yields

$$\Pr\big((|I_1| > t) \vee \ldots \vee (|I_\ell| > t)\big) \leqslant \varepsilon \tag{6.6}$$

and

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell))\ ,\ (\boldsymbol{y}_1|_{J_1}, \ldots, \boldsymbol{y}_m|_{J_m})\big)$$

where $\boldsymbol{J} = (J_1, \ldots, J_m)$ and $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) = G(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell)$. Let $f : \mathbb{R} \to \mathbb{R}$. The gadget G is (t, f)-RPC if it is $(t, p, f(p))$-RPC for every $p \in [0, 1]$.

In the above definition, the first-pass simulator Sim_1 determines the necessary input shares (through the returned collection of sets $\boldsymbol{I}$) for the second-pass simulator Sim_2 to produce a perfect simulation of the leaking wires defined by the set $\mathcal{W}$ together with the output shares defined by the collection of sets $\boldsymbol{J}$. Note that there always exists such a collection of sets $\boldsymbol{I}$ since $\boldsymbol{I} = ([n], \ldots, [n])$ trivially allows a perfect simulation whatever $\mathcal{W}$ and $\boldsymbol{J}$. However, the goal of Sim_1 is to return a collection of sets $\boldsymbol{I}$ with cardinals at most t. The idea behind this constraint is to keep the following composition invariant: for each gadget we can achieve a perfect simulation of the leaking wires plus t shares of each output sharing from t shares of each input sharing.

We shall call *failure event* the event that at least one of the sets I_1, $\ldots$, I_ℓ output of Sim_1 has cardinality greater than t. When (t, p, ε)-RPC is achieved, the failure event probability is upper bounded by ε according to Equation 6.6. We stress that this failure probability can be characterized in the same way as the failure probability for random probing simulation as detailed in Section 6.2.2 but with $\delta_{\mathcal{W}}$ defined as:

$$\delta_{\mathcal{W}} = \begin{cases} 1 & \text{if } \boldsymbol{I} = \mathsf{Sim}_1(\mathcal{W}, \boldsymbol{J}) \text{ with } \big((|I_1| > t) \vee \ldots \vee (|I_\ell| > t)\big), \\ 0 & \text{otherwise.} \end{cases}$$

6.3.2. Composition security

According to the RPC definition, a failure event occurs whenever Sim_2 requires more than t shares of one input sharing to be able to produce a perfect simulation of the leaking wires (*i.e.* the wires with label in $\mathcal{W}$) together with the output shares in $(\boldsymbol{y}_1|_{J_1}, \ldots, \boldsymbol{y}_m|_{J_m})$. Whenever such a failure occurs, the composition invariant is broken. In the absence of failure event, the RPC notion implies that a perfect simulation can be achieved for the full circuit composed of RPC gadgets.

This is formalized in the following theorem. The proof is available in the full version of [BCP+20] (see Appendix F).

Theorem 7 (Random Probing Composition)**.** *Let $t \in \mathbb{N}$, $p, \varepsilon \in [0,1]$, and CC be a standard circuit compiler with (t,p,ε)-RPC base gadgets. For every (randomized) arithmetic circuit C, the randomized circuit $\mathsf{CC}(C)$ is $(p, |C| \cdot \varepsilon)$-random probing secure. Equivalently, the standard circuit compiler CC is (p,ε)-random probing secure.*

6.3.3. Relation with strong non-interference

We recall the *strong non-interfering* (SNI) security notion that was introduced in [BBD+16] to make masking gadgets composable into probing-secure circuits. Informally, a gadget is t-strong non-interfering (t-SNI) if and only if any set of at most t probes, among which t_1 are on internal wires (*i.e.* all wires except the output ones) and t_2 are on output wires, can be perfectly simulated from at most t_1 shares of each input. The following definition formalizes the SNI notion for 2-input 1-output gadgets (see Chapter 2 for the formal definitions of gadgets and wires assignment).

Definition 17 (Strong Non-Interference)**.** *Let G be a 2-to-1 n-share gadgets. G is said t-Strong Non-Interferent (t-SNI), if for every distribution $\mathcal{D}_{(\boldsymbol{x}_1,\boldsymbol{x}_2)}$ over $\mathbb{K}^n \times \mathbb{K}^n$, every set $\mathcal{W}$ of internal wires of G such that $|\mathcal{W}| \leqslant t_1$, and every set $J \subseteq [n]$ of output share indices such that $|J| \leqslant t_2$ and $t_1 + t_2 \leqslant t$, there exists a (two-stage) simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that*

1. *given $\mathsf{Sim}_1(\mathcal{W}, J) = (I_1, I_2)$ where $I_1, I_2 \subseteq [n]$, with $|I_1|, |I_2| \leqslant t_1$,*
2. *and for $(\boldsymbol{x_1}, \boldsymbol{x_2}) \leftarrow \mathcal{D}_{(\boldsymbol{x}_1,\boldsymbol{x}_2)}$ and $\boldsymbol{y} \leftarrow G(\boldsymbol{x_1}, \boldsymbol{x_2})$, we have:*

$$\mathsf{Sim}_2(\mathcal{W}, J, \boldsymbol{x}_1|_{I_1}, \boldsymbol{x}_2|_{I_2}) \overset{id}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}, (\boldsymbol{x}_1, \boldsymbol{x}_2)),\ \boldsymbol{y}|_J\right)$$

A gadget is simply said to be SNI *if it is $(n-1)$-SNI.*

The following result exhibits a relation between the SNI notion and our random probing composition notion. We show that t-SNI gadgets are also RPC for specific parameters that we explicit in the following proposition.

Proposition 2. *Let n, ℓ and t be positive integers and let G be an ℓ-to-1 n-share gadget. If G is t-SNI, then it is also $(t/2, p, \varepsilon)$-RPC for any probability p and ε satisfying:*

$$\varepsilon = \sum_{i=\lfloor \frac{t}{2}+1 \rfloor}^{s} \binom{s}{i} p^i (1-p)^{s-i} \ , \tag{6.7}$$

where $s = |G|$ is the number of wires in G.

In a nutshell, the t-SNI definition implies the existence of a perfect simulation of the *out* distribution in the RPC definition (see Definition 16) for any sets J and $\mathcal{W}$ of cardinalities $|J| \leqslant t/2$ and $|\mathcal{W}| \leqslant t/2$ based on sets $I_1, \ldots I_\ell$ of cardinalities $|I_j| \leqslant t/2$ for every $j \in [\ell]$. The formal proof is available in the full version of [BCP+20] (see Appendix F).

6.4. Conclusion and related works

In this chapter, we have introduced the first composition notion for masking gadgets in the random probing model. Although a close formula for the simulation failure probability, the function $f(p)$, might be hard to derive for generic gadgets (which are defined for any number of shares n), its coefficients can be computed (or at least approximated) for small gadgets, which might be enough in some contexts. In particular, Chapter 8 presents an approach where arbitrary levels of random probing security can be obtained by bootstrapping small gadgets. In our paper [BCP+20], we further describe a tool VRAPS which can verify the random probing composability and characterize the function f for small masking gadgets. In a recent follow-up work [CFOS21], Cassier, Faust, Orlt

and Standaert refine our RPC notion with the concept of *probe distribution table* (PDT). Instead of using a composition invariant based on a threshold t (*i.e.* the leaking wires plus t output shares must be simulatable from t input shares), their PDT computes the failure probability for any pair of input set(s) of shares and output set(s) of shares. Their approach results in a much tighter random probing composition. One of its down side thought is that the PDT computation quickly explodes when several gadgets are composed or when the number of shares increases. Finding good trade-offs between the PDT accuracy and an efficient scaling is an interesting direction for future research.

Part IV.

Achieving noisy leakage security

Chapter 7

Noisy leakage security in quasilinear complexity

Contents

7.1. Introduction

Most probing secure schemes existing in the literature imply a quadratic or beyond quadratic complexity overhead of the number of gates in the protected circuit. In a joint work with Goudarzi and Joux [GJR18] we have introduced a scheme achieving a quasilinear complexity overhead. The original version of this scheme was further extended in a joint work with Goudarzi, Prest and Vergnaud [GPRV21] to make it applicable to any base field and to fit it in the IOS composition framework introduced in Chapter 5. When instantiated with n-sharings, our scheme has complexity overhead $\mathcal{O}(n \log n)$ (with fairly small constant factor) and achieves region probing security with leakage rate $\Theta(1/\log n)$.

This chapter presents this (generalized) quasilinear-complexity scheme. The scheme is described in Section 7.2 while it security is analyzed in Section 7.3.

7.2. A quasilinear-complexity masking scheme

Our scheme is an instantiation of the IOS composition framework (see Chapter 5) for arithmetic ciricuits on a base field $\mathbb{K}$. Specifically our scheme consists of a standard circuit compiler using full refreshing, *i.e.* interleaving a refresh gadget between any two gadgets, as introduced in Section 2.4. While linear gadgets (additions, subtractions, multiplications by constants, etc.) apply sharewisely in complexity $\mathcal{O}(n)$ and the IOS refresh gadget described in Section 5.3 has complexity $\mathcal{O}(n \log n)$, the core idea of our scheme is a way to further achieve $\mathcal{O}(n \log n)$ complexity for the multiplication gadget. This is obtained by relying on a special form of linear sharings which are compatible with fast polynomial evaluation methods.

7.2.1. Encoding

Let $\mathbb{K}$ be the base field of the considered arithmetic circuit and let $\omega \in \mathbb{K}$. Our scheme is based on $\boldsymbol{v}_\omega$-linear sharings (a.k.a. ω-*encodings* in [GJR18]) for a vector $\boldsymbol{v}_\omega$ defined as:

$$\boldsymbol{v}_\omega = (1, \omega, \ldots, \omega^{n-1}) \ .$$

For such a vector, a sharing $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ of a plain value $x \in \mathbb{K}$ satisfies

$$x = \langle \boldsymbol{v}_\omega, \boldsymbol{x} \rangle \ .$$

The sharing $\boldsymbol{x}$ can further be seen as the coefficients of a polynomial $P_{\boldsymbol{x}} = \sum_{i=1}^{n} x_i \theta^{i-1} \in \mathbb{K}[\theta]$ such that $P_{\boldsymbol{x}}(\omega) = x$. The quasilinear complexity can then be achieved by using efficient FFT-based polynomial multiplication.

Remark 6. Note that such encoding is close to –but different from– *Shamir's secret sharing* [Sha79]. In the latter the shares are defined as evaluations of a polynomial in fixed points and for which the plain value is the degree-0 coefficient.

Our scheme makes use of a Fast Fourier Transform (FFT) algorithm that, given any polynomial $P \in \mathbb{K}[\theta]$ of degree $< 2n$, maps the coefficients of P to the evaluations of P in $2n$ points of $\mathbb{K}$, with a complexity of $\widetilde{\mathcal{O}}(n)$ operations. That is:

$$\mathsf{FFT}_{\boldsymbol{\alpha}} : (x_1, x_2, \ldots, x_{2n}) \mapsto (u_1, u_2, \ldots, u_{2n}) \quad \text{with} \quad u_j = \sum_{i=1}^{2n} x_i \cdot \alpha_j^{i-1}$$

for every $j \in [2n]$, for some $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{2n}) \in \mathbb{K}^{2n}$. We further require that this FFT algorithm can be written as an arithmetic circuit on $\mathbb{K}$ solely composed of additions, subtractions and multiplication by constants in $\mathbb{K}$, and that it features an inverse FFT algorithm with the same properties (in terms of type and number of operations). Possible FFT algorithms matching those criteria are discussed in Section 7.2.4.

7.2.2. Multiplication gadget

Let $\boldsymbol{v}'_\omega \in \mathbb{K}^{2n}$ be the vector defined as

$$\boldsymbol{v}'_\omega = \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}(1, \omega, \omega^2, \ldots, \omega^{2n-1}) \ .$$

Let $\mathsf{Refresh}(\boldsymbol{v}'_\omega, \cdot)$ be a $\boldsymbol{v}_\omega$-sharing refresh gadget, such as the BCPZ variant introduced in Section 5.3. Let $\mathsf{Compress}$ be the $\mathbb{K} \times \mathbb{K}^{2n} \to \mathbb{K}^n$ mapping defined as

$$\mathsf{Compress}(\omega; t_1, t_2, \ldots, t_{2n}) = (t_1 + \omega^n \cdot t_{n+1},\ t_2 + \omega^n \cdot t_{n+2},\ \ldots,\ t_n + \omega^n \cdot t_{2n}) \ .$$

Given two $\boldsymbol{v}_\omega$-sharings $\boldsymbol{x}$ and $\boldsymbol{y}$, the multiplication gadget produces an output $\boldsymbol{v}_\omega$-sharing $\boldsymbol{z}$ as follows:

1. $\boldsymbol{r} \leftarrow \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{x} \parallel \boldsymbol{0})$
2. $\boldsymbol{s} \leftarrow \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{y} \parallel \boldsymbol{0})$
3. $\boldsymbol{u} \leftarrow \boldsymbol{r} \otimes \boldsymbol{s}$
4. $\boldsymbol{u}' \leftarrow \mathsf{Refresh}(\boldsymbol{v}'_\omega; \boldsymbol{u})$
5. $\boldsymbol{t} \leftarrow \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{u}')$
6. $\boldsymbol{z} \leftarrow \mathsf{Compress}(\omega; \boldsymbol{t})$

where $\boldsymbol{0}$ denotes the n-dimensional all-0 vector, $\parallel$ denote the concatenation operator and $\otimes$ denote the coordinatewise product. Note that $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{u}$, $\boldsymbol{u}'$, $\boldsymbol{t}$ are $(2n)$-dimensional sharings. Only the input/output sharings $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are n-dimensional vectors. The procedure is illustrated on Figure 7.1.

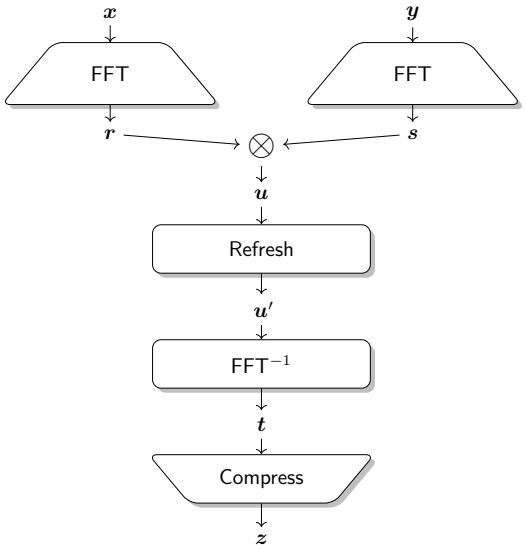

Figure 7.1.: Multiplication gadget.

Correctness. Let x and y be the values encoded by $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively and let $P_{\boldsymbol{x}} \in \mathbb{K}[\theta]$ and $P_{\boldsymbol{y}} \in \mathbb{K}[\theta]$ be polynomials of degree $n-1$ whose coefficients are the coordinates of $\boldsymbol{x}$ and $\boldsymbol{y}$, so that we have $P_{\boldsymbol{x}}(\omega) = x$ and $P_{\boldsymbol{y}}(\omega) = y$.

Let us first assume that Step 4 applies an identity mapping, *i.e.* $\boldsymbol{u}' = \boldsymbol{u}$. Then Steps 1–5 perform a classical FFT-based polynomial multiplication. Namely, the coordinates of $\boldsymbol{t}$ are the coefficients of the polynomial $P_{\boldsymbol{t}} \in \mathbb{K}[\theta]$ such that $P_{\boldsymbol{t}} = P_{\boldsymbol{x}} \cdot P_{\boldsymbol{y}}$, and in particular $P_{\boldsymbol{t}}(\omega) = x \cdot y$. Then Step 6 outputs a vector $\boldsymbol{z}$ such that $\langle \boldsymbol{v}_\omega, \boldsymbol{z} \rangle = P_{\boldsymbol{t}}(\omega) = x \cdot y$, *i.e.* a $\boldsymbol{v}_\omega$-sharing of $x \cdot y$.

Let $\boldsymbol{v}''_\omega = (1, \omega, \omega^2, \ldots, \omega^{2n-1})$, then we have

$$P_{\boldsymbol{t}}(\omega) = \langle \boldsymbol{v}''_\omega, \boldsymbol{t} \rangle = x \cdot y \quad \Leftrightarrow \quad \langle \boldsymbol{v}'_\omega, \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{t}) \rangle = x \cdot y \ .$$

By correctness of the FFT-based polynomial multiplication, we hence have that $\boldsymbol{u} = \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{t})$ is a $\boldsymbol{v}'_\omega$-sharing of $x \cdot y$. Let us now consider the actual multiplication gadget with refreshing at Step 4. By correctness of the refresh algorithm, $\boldsymbol{u}'$ is also a $\boldsymbol{v}'_\omega$-sharing of $x \cdot y$, and by the above relation we have that $\langle \boldsymbol{v}'_\omega, \boldsymbol{u}' \rangle = x \cdot y$ implies $\langle \boldsymbol{v}''_\omega, \mathsf{FFT}^{-1}_{\boldsymbol{\alpha}}(\boldsymbol{u}') \rangle = x \cdot y$, which is $\langle \boldsymbol{v}''_\omega, \boldsymbol{t} \rangle = x \cdot y$. We hence get the correctness of the multiplication gadget.

7.2.3. Overall circuit compiler

Our overall scheme is a standard circuit compiler (see Definition 4) using full refreshing (see Definition 5). It is based on sharewise gadgets for linear operations (addition, subtraction multiplication by constants, or any $\mathbb{K}$-linear mapping – see linear gadget description in Section 3.4.1), an IOS quasilinear refresh gadget of $\boldsymbol{v}_\omega$-linear sharings (see *e.g.* Algorithm 7), and the multiplication gadget depicted above.

The value ω might either be a constant parameter of the scheme or randomly sampled from $\mathbb{K}$, depending on the target security proof or argument (see next section). In any case, the value of ω is assumed to be known to the adversary. Note that in case of a random ω, the adversary probes are placed independently of the drawing of ω (since we do not consider an adaptive probing adversary).

7.2.4. Field extension and FFT algorithm

In order to instantiate our scheme with sharing order n over a finite field $\mathbb{K}$ we need an FFT algorithm which allows quasilinear multiplication of polynomials of degree at most n with coefficients in $\mathbb{K}$ and

which can be written as an arithmetic circuit on $\mathbb{K}$ solely composed of additions, subtractions and multiplications by constants.

A possible approach suggested in our original paper [GJR18] (see Appendix D) is to consider finite fields $\mathbb{K} = \mathbb{F}_q$ that contain the $(2n)$-th roots of unity (*i.e.* such that $2n \mid q-1$). We can then use the so-called *number theoretic transform* (NTT) which requires $3N \log N$ arithmetic gates for an input of size $N = 2n$. Any computation can then be embedded into such a field. See [GJR18] (Appendix D) for details.

In some contexts, we might want to protect a cryptographic primitive defined on a specific base field $\mathbb{K}$ while avoiding a (practically inefficient) embedding in a field $\mathbb{F}_q$ containing the $(2n)$th root of unity. To extend the original scheme to any finite field $\mathbb{K}$, we can use the general *additive* FFT due to Cantor [WZ88, Can89]. For a sharing order n and a base field $\mathbb{K} = \mathbb{F}_{p^m}$, this FFT can be instantiated over the extension $\mathbb{F}_{p^\ell}$ where ℓ is the minimum even value greater than m such that $p^\ell \geqslant 2n$.

For the particular case of a binary base field $\mathbb{K} = \mathbb{F}_{2^m}$ (which is of particular practical importance *e.g.* to protect AES), we can use the Gao-Mateer additive FFT [GM10] which is a variant of Cantor additive FFT that efficiently addresses binary fields. Using this transform, if m is even with $2^m \geqslant 2n$, then we can use directly our technique over $\mathbb{K} = \mathbb{F}_{2^m}$ and otherwise we can simply instantiate it over $\mathbb{K} = \mathbb{F}_{2^\ell}$ where ℓ is the smallest even integer for which $2^\ell \geqslant 2n$ and $m \mid \ell$ (see [GPRV21] – Appendix E for details).

7.3. Region probing security

7.3.1. Security reduction

This section provides a security reduction for our scheme. We show that under the probing security of the FFT, the scheme achieves region probing security. More formally, the reduction is based on the following hypothesis on the FFT algorithm.

Hypothesis 1 (FFT Probing Security). The circuits processing

$$\mathsf{FFT}_{\boldsymbol{\alpha}} : (\boldsymbol{x} \parallel \mathbf{0}) \mapsto \boldsymbol{r} \quad \text{and} \quad \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1} : \boldsymbol{u}' \mapsto \boldsymbol{t}$$

are t_{FFT}-probing secure w.r.t. the $\boldsymbol{v}_\omega$-encoding and the $\boldsymbol{v}'_\omega$-encoding respectively.

We can then state our reduction theorem. The meaning of Hypothesis 1 is further discussed in [GPRV21] – Appendix E.

Theorem 8. *Under Hypothesis 1 and the t_{R}-IOS property of the refresh gadget, our compiler is r-region probing secure with*

$$r = \max_{t \leqslant t_{\mathsf{R}}} \min \left(\frac{t_{\mathsf{FFT}} - 6t}{2 \cdot |\mathsf{FFT}|} , \frac{t}{|G_{\mathsf{R}}|} \right) \tag{7.1}$$

where $|\mathsf{FFT}|$ denotes the (maximum) number of wires in the FFT circuits for $2n$ input sharings.

Note that the IOS refresh gadget described in Section 5.3 satisfies $t_{\mathsf{R}} = n - 1$ and $|G_{\mathsf{R}}| = 3n \log n$. Assuming the FFT algorithm is quasilinear and that it can tolerate a linear number of probes (in the encoding order n) and denoting

$$\begin{aligned} |\mathsf{FFT}| &= \alpha \cdot n \log n \\ |G_{\mathsf{R}}| &= \beta \cdot n \log n \\ t_{\mathsf{FFT}} &= \gamma \cdot n \end{aligned}$$

for some constants α, β and γ (with $\gamma < 1$), one can check that the minimum in Equation 7.1 is reached for

$$t = \left(\frac{\beta\gamma}{2(\alpha + 3\beta)} \right) \cdot n \quad \Longrightarrow \quad r = \left(\frac{\gamma}{2(\alpha + 3\beta)} \right) \cdot \frac{1}{\log n} . \tag{7.2}$$

In particular, we obtain a probing rate $r = \Theta(1/\log n)$.

The proof of Theorem 8 is based on the two following lemmas (see proofs in [GPRV21] – Appendix E) and by applying the IOS composition theorem (see Theorem 5).

Lemma 2. *Under Hypothesis 1 the circuit processing*

$$(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{u} = \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{x} \parallel \boldsymbol{0}) \otimes \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{y} \parallel \boldsymbol{0})$$

is t_{FFT}-probing secure w.r.t. the $\boldsymbol{v}_\omega$-encoding.

Lemma 3. *Under Hypothesis 1 the circuit processing*

$$\boldsymbol{u}' \mapsto \boldsymbol{z} = \mathsf{Compress}(\omega; \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{u}'))$$

is $(t_{\mathsf{FFT}}/2)$-probing secure w.r.t. the $\boldsymbol{v}'_\omega$-encoding.

Theorem 8 formally shows that if probing security can be demonstrated for the FFT algorithm, then we obtain region probing security for our scheme. Unfortunately, it is not clear whether the classical FFT algorithms are probing secure or not. To some extent, this open issue is related to the choice of ω: some choices clearly lead to probing insecurity (*e.g.* $\omega = 0$ or to some nth power of unity when the NTT is used) while it is not clear whether some choices exist for which we can get the desired t-probing security (*i.e.* with $t = \mathcal{O}(n)$ to get a probing rate $\Theta(1/\log n)$).

While the question of a probing secure FFT is still open with respect to a given choice of ω, we can tackle this issue with a random choice of ω on a large-enough base field $\mathbb{K}$. We thus obtain (statistical) region probing security without relying on Hypothesis 1, as detailed hereafter.

7.3.2. Probing security of the FFT on large fields

The region-probing security of our scheme simply holds from the IOS property of the refresh gadget and assuming that the underlying FFT algorithm is somehow linear. This is captured by the following definition.

Definition 18 (Linear FFT)**.** *An FFT algorithm is said* linear *if the circuits processing*

$$\mathsf{FFT}_{\boldsymbol{\alpha}} : (\boldsymbol{x} \parallel \boldsymbol{0}) \mapsto \boldsymbol{r} \quad \textit{and} \quad \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1} : \boldsymbol{u}' \mapsto \boldsymbol{t}$$

are composed of additions, subtractions and multiplications by constants on $\mathbb{K}$.

The above definition implies that the value carried by each wire in the FFT circuit can be expressed as a linear combination of the coordinates of the input sharing. This property is necessary to apply our security argument. Note that this requirement is relatively weak since it is satisfied by classical FFT algorithms such as the NTT and the Gao-Mateer additive FFT [GM10].

For a linear FFT algorithm, every value v taken by a wire of $\mathsf{FFT}_{\boldsymbol{\alpha}}$ (resp. $\mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}$) on input a $\boldsymbol{v}_\omega$-sharing $\boldsymbol{x}$ (resp a $\boldsymbol{v}'_\omega$-sharing $\boldsymbol{u}'$) can be expressed as

$$v = \sum_{i=0}^{n-1} a_i x_i \tag{7.3}$$

where the a_i's are constant coefficients over $\mathbb{K}$. The lemmas use the following notation

$$[v] = (a_0, a_1, \ldots, a_{n-1})^{\mathsf{T}} \tag{7.4}$$

for the column vector of coefficients of such a wire value. Moreover, $[v_0, v_1, \ldots, v_\ell]$ shall denote the matrix with column vectors $[v_0]$, $[v_1]$, $\ldots$, $[v_\ell]$.

Lemma 4. *Let $v_1, v_2, \ldots, v_\ell$ be the values taken by $\ell < n$ wires of $\mathsf{FFT}_{\boldsymbol{\alpha}}$ on input a uniform $\boldsymbol{v}_\omega$-sharing of a variable x. The distribution of the tuple $(v_1, v_2, \ldots, v_\ell)$ is statistically independent of x iff*

$$\boldsymbol{v}_\omega \notin \mathrm{span}(v_1, \ldots, v_\ell) \ , \tag{7.5}$$

where span$(\cdot)$ *refers to the linear span of the input vectors. The same proposition holds for $\mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}$ with $\boldsymbol{v}'_\omega$ in place of $\boldsymbol{v}_\omega$.*

Lemma 5. *Let ω be a uniform random element in $\mathbb{K}^*$. And let $v_1, v_2, \ldots, v_\ell$ be a set of $\ell < n$ intermediate variables of $\mathsf{FFT}_{\boldsymbol{\alpha}}$. We have:*

$$\Pr\big(\boldsymbol{v}_\omega \in \mathrm{span}(v_1, \ldots, v_\ell)\big) \leqslant \frac{\ell}{|\mathbb{K}| - 1} < \frac{n}{|\mathbb{K}|} \ , \tag{7.6}$$

where the above probability is defined over a uniform random choice of ω. The same proposition holds for $\mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}$ with $\boldsymbol{v}'_\omega$ in place of $\boldsymbol{v}_\omega$.

The proofs of these lemmas are given in [GJR18] (see Appendix D).

From these two lemmas, the values taken by any set of $\ell < n$ wires of $\mathsf{FFT}_{\boldsymbol{\alpha}}$ or $\mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}$ can be perfectly simulated without knowledge of the underlying plain value. The simulation simply works by taking a random x, picking a random $\boldsymbol{v}_\omega$-sharing of x, and evaluating the wires $v_1, \ldots, v_\ell$ accordingly. By the above lemmas such a simulation fails with probability lower than $n/|\mathbb{K}|$.

Corollary 1. *If the FFT circuit is linear, our compiler is (r, ε)-region probing secure with r being the rate from Theorem 8 and $\varepsilon = \frac{n}{|\mathbb{K}|}$.*

From the above corollary, we see that to achieve κ-bit security, $\mathbb{K}$ should be such that $|\mathbb{K}| > 2^\kappa n$, or equivalently, elements of $\mathbb{K}$ should be of size greater than $\kappa + \log n$. If this is not ensured by the desired base field, one needs to consider a field extension satisfying this constraint.

7.4. Conclusion and related works

This chapter has overviewed the quasilinear-complexity masking scheme that we proposed in [GJR18, GPRV21]. This scheme achieves region probing security with *quasiconstant* leakage rate $\Theta(1/\log n)$ as long as the underlying base field is large enough, *i.e.* with elements of size $\kappa \log n$ (for κ-bit security). The region probing security of our scheme is based on the IOS framework with a quasilinear refresh gadget which is a variant of the Battistello, Coron, Prouff and Zeitoun (BCPZ) refresh gadget [BCPZ16a] (see Chapter 5). In [GPRV21] (see Appendix E), we further present some applications of our scheme to protect the AES and MiMC ciphers and show how its performance favorably scales compared to a standard quadratic-complexity probing-secure scheme (based on the ISW construction). Other masking schemes based on polynomial encoding, and in particular on Shamir's secret sharing, have been proposed in [GM11, PR11, CPR12, CRZ13]. However their multiplication gadgets, which are derived from multi-party computation protocols, have at least a quadratic complexity. In contrast, we could achieve a quasilinear complexity by relying on an FFT-based polynomial multiplication, with the key idea of using a random evaluation point ω for decoding. We further show that our scheme is secure for any field and (fixed) evaluation point ω provided that the underlying FFT achieves probing security with respect to the associated ω-encoding. Finding such probing-secure FFT for small field and/or fixed ω is an interesting open issue to allow more efficient instantiation of our scheme in some contexts. Another direction for improvement would be to achieve a constant leakage rate instead of the *quasiconstant* $\Theta(1/\log n)$. Another approach to probing security in quasilinear complexity was already proposed in the seminal work of Ishai, Sahai and Wagner [ISW03]. Their construction uses the principle of *wire shuffling* to achieve statistical t-probing security with complexity overhead $\mathcal{O}(t \log t)$ (with a constant factor κ^{10} where κ is the security parameter). This appraoch was recently improved by Coron and Spignoli in [CS21b]. They notably achieve t-probing security with a *linear* complexity overhead $\mathcal{O}(t)$ by relying on the *RAM model*.

Chapter 8

Noisy leakage security through random probing expansion

Contents

8.1. Introduction

Among the few schemes providing random probing security [ISW03, Ajt11, DDF14, ADF16, AIS18, GJR18], some tolerate a non-constant leakage probability, which needs to decrease with the number of shares, *e.g.* $p = \mathcal{O}(1/n)$ [ISW03, DDF14] or $p = \mathcal{O}(1/\log n)$ [GJR18], in order to reach exponential security $\varepsilon = \exp(\mathcal{O}(n))$, while others achieve the desirable feature of tolerating a constant leakage probability $p = \mathcal{O}(1)$ [Ajt11, ADF16, AIS18]. Prior to the work presented in this chapter, a single work had introduced such a scheme for which the tolerated (constant) leakage probability was made explicit. This scheme due to Ananth, Ishai and Sahai [AIS18] is based on an *expansion strategy*, which consists in applying a base compiler –using a fixed (small size) encoding– several times to a circuit until reaching the desired security level. According to the analysis we made in [BCP+20], the instantiation of this approach proposed in [AIS18] tolerates a leakage probability of 2^{-26} and has asymptotic complexity of $\mathcal{O}(\kappa^{8.2})$ to achieve a security $\varepsilon = 2^{-\kappa}$.

In [BCP+20, BRT21] we have revisited this expansion approach and introduce the framework of *random probing expandability* (RPE). We have shown that the *expanding compiler* can bootstrap simple base gadgets as long as they satisfy our new RPE security notion. We have further shown that

the obtained complexity rely on the so-called *amplification order* of the base RPE gadgets, and we have exhibited some bounds for this parameter. We have then introduced generic gadget constructions achieving RPE with optimal amplification order for any number of shares. We have further put forward concrete instantiations of our framework based on gadgets with small number of shares (3 and 5) which today achieve the highest leakage probability reported so far.

This chapter gives an overview of these works. We first introduce the random probing expansion framework in Section 8.2. Section 8.3 details our asymptotic analysis and provides related bounds. Section 8.4 presents our generic gadget constructions while Section 8.5 provides our concrete instantiation results.

8.2. Random probing expandability framework

Constructing random-probing-secure circuit compilers with an expansion strategy has been proposed by Ananth, Ishai and Sahai in [AIS18]. Such a strategy was previously used in the field of multi-party computation (MPC) with different but close security goals [HM00, CDI+13]. Note that this approach is called *composition* in [AIS18] since it roughly consists in composing a base circuit compiler several times. We prefer the terminology of *expansion* here to avoid any confusion with the notion of composition for masking gadgets as usually considered in the masking literature (and in Part III of the present thesis).

8.2.1. Expanding compiler

The basic principle of the expansion strategy is as follows. Assume that we have a family of n-share gadgets $\{G_g\}$ for some base of gates $\mathbb{B}$ and consider the underlying standard circuit compiler CC (as formally defined in Section 2.4). Let us recall that CC simply consists in replacing each gate g in the original circuit by the corresponding gadget G_g and replacing each wire by n wires carrying a sharing of the original value. We shall call CC the *base circuit compiler* in what follows. We can derive new n^2-share gadgets by simply applying CC to each gadget G_g: $G_g^{(2)} = \mathsf{CC}(G_g)$ for every $g \in \mathbb{B}$. 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 $G_g^{(k)}$ and each wire has been replaced by n^k wires carrying an (n^k)-linear sharing of the original wire. The underlying compiler is called *expanding compiler* which is formally defined hereafter (we refer to Section 2.4 for the definition of standard circuit compiler).

Definition 19 (Expanding Compiler)**.** *Let* CC *be the standard circuit compiler with* n*-share base gadgets. The* expanding compiler *with* expansion level k *and base compiler* CC *is the circuit compiler* $(\mathsf{CC}^{(k)}, \mathsf{Enc}^{(k)}, \mathsf{Dec}^{(k)})$ *satisfying the following:*

1. *The input encoding* $\mathsf{Enc}^{(k)}$ *is an* (n^k)*-linear encoding.*
2. *The output decoding* Dec *is the* (n^k)*-linear decoding mapping.*
3. *The circuit compilation is defined as*

$$\mathsf{CC}^{(k)}(\cdot) = \underbrace{\mathsf{CC} \circ \mathsf{CC} \circ \cdots \circ \mathsf{CC}}_{k \text{ times}}(\cdot)$$

The goal of the expansion strategy in the context of random probing security is to replace the leakage probability p of a wire in the original circuit by the failure event probability ε in the subsequent gadget

simulation. If this simulation fails then one needs the full input sharing for the gadget simulation, which corresponds to leaking the corresponding wire value in the base case. The security is thus amplified by replacing the probability p in the base case by the probability ε (assuming that we have $\varepsilon < p$). Let $p_{\max} < 1$ denote some maximal tolerated probability parameter such that the failure event probability ε can be upper bounded by some function of the leakage probability: $\varepsilon \leqslant f(p)$ for every leakage probability $p \in [0, p_{\max}]$. Then the expanding compiler with expansion level k shall result in a security amplification as

$$p = \varepsilon_0 \xrightarrow{f} \varepsilon_1 \xrightarrow{f} \cdots \xrightarrow{f} \varepsilon_k = f^{(k)}(p) \ , \tag{8.1}$$

which for an adequate function f (for instance $f : p \mapsto p^2$) provides exponential security. In order to get such a security expansion, the gadgets must satisfy a stronger notion than random probing security or the random probing composability notion introduced in Chapter 6. We call this notion *random probing expandability*.

8.2.2. Random probing expandability

The random probing expandability notion can be seen as a stronger version of the random probing composability notion (see Definition 16) that supports the expansion security and specifically Equation 8.1. In the context of random probing composability, the failure event occurs whenever more than t shares from an input sharing are necessary to complete a perfect simulation (see Chapter 6). For a gadget to be expandable we need further conditions. As a first requirement, a two-input gadget should have a failure probability which is independent for each input. This is because in the base case, each wire as input of a gate leaks independently. On the other hand, in case of failure event in the subsequent gadget, the simulator should be able to produce a perfect simulation of the full output (that is the full input for which the failure occurs). To do so, the simulator is given the clear output (which is obtained from the simulation of the base case) plus any set of $n-1$ output shares. This means that whenever the set J (output shares indices) is of cardinal greater than t, which means a failure in the subsequent gadget, the simulator can replace it by any set J' of cardinal $n-1$. Formally, we define random probing expandability as follows:

Definition 20 (Random Probing Expandability)**.** *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 simulator Sim_1 and a probabilistic simulator Sim_2 such that for every distribution $\mathcal{D}_{(\boldsymbol{x},\boldsymbol{y})}$ over $\mathbb{K}^n \times \mathbb{K}^n$, for every set $J \subseteq [n]$ and for every $p \in [0, 1]$, the random experiment*

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

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*

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{8.2}$$

   with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),
2. *J' is such that $J' = J$ if $|J| \leqslant t$ and $J' \subseteq [n]$ with $|J'| = n - 1$ otherwise,*
3. *the output distribution satisfies*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\boldsymbol{x}, \boldsymbol{y}))\ ,\ \boldsymbol{z}|_{J'}\big) \tag{8.3}$$

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

The parameter t in the above definition is referred to as the *RPE threshold*. Note that a gadget can achieve the notion for different RPE thresholds (each leading to different functions f).

The RPE notion can be simply extended to gadgets with 2 outputs: the Sim_1 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 simulator must then produce an output including $\boldsymbol{z}_1|_{J_1'}$ and $\boldsymbol{z}_2|_{J_1'}$ where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are the output sharings. The RPE notion can also be simply extended to gadgets with a single input: the Sim_1 simulator produces a single set I so that the failure event ($|I| > t$) occurs with probability lower than ε (and the Sim_2 simulator is then simply given $\boldsymbol{x}|_I$ where $\boldsymbol{x}$ is the single input sharing). A formal definition of RPE for 1-to-2 gadgets (*e.g.* copy gadgets) is provided in the full version of [BCP+20] (see Appendix F).

It is not hard to check that the above expandability notion is stronger that the composability notion introduced in Chapter 6. Formally, we have the following proposition:

Proposition 3. *Let $f = \mathbb{R} \to \mathbb{R}$ and $n \in \mathbb{N}$. Let G be an n-share gadget. If G is (t, f)-RPE then G is (t, f')-RPC, with $f'(\cdot) = 2 \cdot f(\cdot)$.*

Relaxation. The requirement of the RPE property that the failure events $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent might seem too strong. In practice it might be easier to show or verify that some gadgets satisfy a weaker notion. We say that a gadget is (t, f)-*weak random probing expandable* (wRPE) if the failure events verify $\Pr(\mathcal{F}_1) \leqslant \varepsilon$, $\Pr(\mathcal{F}_2) \leqslant \varepsilon$ and $\Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) \leqslant \varepsilon^2$ instead of Equation 8.2 in Definition 20. Although being easier to achieve and to verify, the latter is actually not much weaker than the original RPE notion. We have the following reduction of RPE to wRPE. The proof is available in the full version f [BCP+20] (see Appendix F).

Proposition 4. *Let $f = [0, 1] \to [0, 0.14]$. Let $G : \mathbb{K}^n \times \mathbb{K}^n \to \mathbb{K}^n$ be an n-share gadget. If G is (t, f)-wRPE then G is (t, f')-RPE with $f'(\cdot) = f(\cdot) + \frac{3}{2} f(\cdot)^2$.*

Assume that we can show or verify that a gadget is wRPE with the following failure event probabilities

$$\Pr(\mathcal{F}_1) = f_1(p) \ , \quad \Pr(\mathcal{F}_2) = f_2(p) \quad \text{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = f_{12}(p) \ ,$$

for every $p \in [0, 1]$. Then the above proposition implies that the gadget is (p, f)-RPE with

$$f : p \mapsto f_{\max}(p) + \frac{3}{2} f_{\max}(p)^2 \quad \text{with} \quad f_{\max} = \max(f_1, f_2, \sqrt{f_{12}}) \ .$$

8.2.3. Expansion security

In [BCP+20] we show that level-k gadgets $G^{(k)} = \mathsf{CC}^{(k-1)}(G)$ achieve a variant of RPE (in which the output set J must belong to the *adequate* subsets of $[n^k]$). While this variant is a restriction of the general RPE notion, it is still stronger than random probing composability. In particular, if the base gadgets are (t, f)-RPE then the level-k gadgets $G^{(k)}$ achieve $(t', 2f^{(k)})$-RPC for some $t' < n^k$. The random probing security of the expanding compiler can then be deduced from the random probing composition theorem (see Theorem 7). Formally, we get:

Theorem 9. *Let $n \in \mathbb{N}$ and $f : \mathbb{R} \to \mathbb{R}$. Let CC be the standard circuit compiler with base gadget $\{G_g\}_{g \in \mathbb{B}}$. Let $\mathsf{CC}^{(k)}$ be the expanding compiler with base compiler CC. If the base gadgets $\{G_g\}_{g \in \mathbb{B}}$ are (t, f)-RPE, then $\mathsf{CC}^{(k)}$ is $(p, 2 \cdot f^{(k)}(p))$-random probing secure.*

The proof is provided in [BCP+20] (see Appendix F).

8.3. Asymptotic analysis

In this section we show that the asymptotic complexity of a randomized circuit $\widehat{C} = \mathsf{CC}^{(k)}(C)$ is $|\widehat{C}| = \mathcal{O}\big(|C| \cdot \kappa^e\big)$, where κ is the security parameter we want to reach (*i.e.* $\widehat{C}$ is (p, ε)-random probing

secure with $\varepsilon = 2^{-\kappa}$) and where the exponent e is a constant that we make explicit hereafter. In particular, we show that e is determined by two parameters: the *amplification order* and *eigen-complexity* of the base compiler.

8.3.1. Amplification order

The complexity of the expanding compiler relates to a parameter called *amplification order* of the gadgets, which is formally define as follows.

Definition 21 (Amplification Order)**.**

- *Let $f : \mathbb{R} \to \mathbb{R}$ which satisfies*
$$f(p) = c_d\, p^d + \mathcal{O}(p^{d+\Theta(1)})$$
as p tends to 0, for some $c_d > 0$. 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 is defined with respect to its RPE threshold t. Namely, different RPE thresholds t are likely to yield different amplification orders d (or equivalently d can be thought of as a function of t).

8.3.2. Eigen-complexity

Let $\{G_g\}_{g\in\mathbb{B}}$ be the base gadgets of CC. For every gate $g \in \mathbb{B}$, let us define the gate-count vector of the gadget G_g as:
$$\boldsymbol{N}_g := (N_{g,g_1}, \ldots, N_{g,g_{|\mathbb{B}|}}, N_{g,r})^\mathsf{T}$$
where, given an indexing $\mathbb{B} = \{g_1, \ldots, g_{|\mathbb{B}|}\}$, N_{g,g_i} denotes the numbers of gates g_i in the gadget G_g, while $N_{g,r}$ denotes the number of random gates in the gadget G_g. Let us further define the *complexity matrix* $\boldsymbol{M}$ associated to the base compiler CC as
$$\boldsymbol{M} = \big(\boldsymbol{N}_{g_1} \mid \cdots \mid \boldsymbol{N}_{g_{|\mathbb{B}|}} \mid \boldsymbol{N}_r\big) \quad \text{with} \quad \boldsymbol{N}_r = (0, \ldots, 0, n)^\mathsf{T}\ ,$$
where $\boldsymbol{N}_r$ is the gate-count vector for the gadget replacing random gates when applying CC, which is simply composed of n random gates (by definition of the standard circuit compiler –see Section 2.4). In the following we shall assume that $\boldsymbol{M}$ is diagonalizable (which is always the case in practice for considered sets of gadgets).

It can be checked that applying the standard circuit compiler with base gadgets $\{G_g\}_{g\in\mathbb{B}}$ to a circuit C with gate-count vector $\boldsymbol{N}_C$ gives a circuit $\widehat{C}$ with gate-count vector $\boldsymbol{N}_{\widehat{C}} = \boldsymbol{M} \cdot \boldsymbol{N}_C$. It follows that the kth power of $\boldsymbol{M}$ gives the gate counts for the level-k gadgets as:
$$\boldsymbol{M}^k = \underbrace{\boldsymbol{M} \cdot \boldsymbol{M} \cdots \boldsymbol{M}}_{k \text{ times}} = \big(\boldsymbol{N}_{g_1}^{(k)} \mid \cdots \mid \boldsymbol{N}_{g_{|\mathbb{B}|}}^{(k)} \mid \boldsymbol{N}_r^{(k)}\big) \quad \text{with} \quad \boldsymbol{N}_r^{(k)} = (0, \ldots, 0, n^k)^\mathsf{T}$$
where $\boldsymbol{N}_{g_i}^{(k)}$ is the gate-count vector of the level-k gadget $G_{g_i}^{(k)}$ for every i. Let us denote the eigendecomposition of $\boldsymbol{M}$ as $\boldsymbol{M} = \boldsymbol{Q} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{Q}^{-1}$. We get
$$\boldsymbol{M}^k = \boldsymbol{Q} \cdot \boldsymbol{\Lambda}^k \cdot \boldsymbol{Q}^{-1} \quad \text{with} \quad \boldsymbol{\Lambda}^k = \begin{pmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_{|\mathbb{B}|+1}^k \end{pmatrix}$$
where $\lambda_1, \ldots, \lambda_{|\mathbb{B}|+1}$ are the eigenvalues of $\boldsymbol{M}$. We then obtain an asymptotic complexity of
$$|\widehat{C}| = \mathcal{O}\Big(|C| \cdot \sum\nolimits_i |\lambda_i|^k\Big) = \mathcal{O}\Big(|C| \cdot \lambda_{\max}^k\Big) \quad \text{with} \quad \lambda_{\max} := \max_i(|\lambda_i|) \tag{8.4}$$

for a randomized circuit $\widehat{C} = \mathsf{CC}^{(k)}(C)$. The parameter $\lambda_{\max}$ is called the *eigen-complexity* of the base compiler. We note that the constant in the above $\mathcal{O}(\cdot)$ solely depends on the matrix of eigenvectors $\boldsymbol{Q}$ and shall be fairly small in practice.

Characterization for arithmetic circuits. Let us consider the particular case of arithmetic circuits for fields of characteristic 2, which are composed of addition, copy and multiplication gates. And let us assume that the addition and copy gadgets do not involve multiplication gates. In this setting, the eigenvalues of the complexity matrix $\boldsymbol{M}$ are the following:

$$(\lambda_1, \lambda_2) = \mathsf{eigenvalues}(\boldsymbol{M}_{ac}) \ , \quad \lambda_3 = N_{m,m} \quad \text{and} \quad \lambda_4 = n$$

where $\boldsymbol{M}_{ac}$ is the 2×2 block matrix of $\boldsymbol{M}$ *i.e.*

$$\boldsymbol{M}_{ac} = \begin{pmatrix} N_{a,a} & N_{c,a} \\ N_{a,c} & N_{c,c} \end{pmatrix} \ ,$$

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

$$\lambda_{\max} = \max(\mathsf{eigenvalues}(\boldsymbol{M}_{ac}), N_{m,m}, n) \ .$$

Interestingly, the number of random gates used by the operation gadgets does not impact $\lambda_{\max}$, and hence does not impact the complexity of the expanding compiler. Similarly, the number of addition and copy gates in the multiplication gadget does not impact $\lambda_{\max}$ either. Those observations are helpful while searching for base gadgets yielding an efficient instantiation of the random probing expansion framework.

8.3.3. Complexity of the expanding compiler

In order to reach a security level $\varepsilon = 2^{-\kappa}$ for some target security parameter κ and assuming that we have a security expansion $p \to f^{(k)}(p)$, the expansion level k must be chosen so that $f^{(k)}(p) \leqslant 2^{-\kappa}$.

Let d be the amplification order of f, *i.e.* the (minimum) amplification order of the gadgets $\{G_g\}_{g \in \mathbb{B}}$. We have

$$f(p) = \sum_{i \geqslant d} c_i \, p^i \leqslant (c_d + o(1)) \, p^d \leqslant c'_d \, p^d \ ,$$

where $c'_d = c_d + o(1)$, as p tends to 0. In other words, for any $p < 1$, there exists a constant c'_d satisfying the above inequality. This upper bound implies $f^{(k)}(p) < (c'_d \, p)^{d^k}$. Hence, to satisfy the required security $f^{(k)}(p) \leqslant 2^{-\kappa}$ while assuming $c'_d \, p < 1$, the number k of expansion steps must satisfy:

$$k \geqslant \log_d(\kappa) - \log_d(-\log_2(c'_d \, p)) \ .$$

We can then rewrite Equation 8.4 as

$$|\widehat{C}| = \mathcal{O}\big(|C| \cdot \kappa^e\big) \quad \text{with} \quad e = \frac{\log \lambda_{\max}}{\log d} \ . \tag{8.5}$$

We thus obtain an explicit formula for the asymptotic complexity of the expanding compiler with respect to the amplification order of the base gadgets as well as the maximal eigenvalue of the complexity matrix associated to the base compiler.

Remark 7. Equation 8.5 is obtained by considering the leakage probability as a constant. If the leakage probability is not considered as a constant but as a parameter, the formula generalizes as:

$$|\widehat{C}| = \mathcal{O}\left(|C| \cdot \left(\frac{\kappa}{\log p}\right)^e\right) \quad \text{with} \quad e = \frac{\log \lambda_{\max}}{\log d} \ .$$

We see that the exponent e applies to the ratio of the security level we want to reach, *i.e.* $\kappa = \log \varepsilon$, over the security level we start from, *i.e.* $\log p$.

8.3.4. Bounding the amplification order

According to the above analysis, the complexity of the expanding compiler is tightly related to the notion of amplification order of its base gadgets. Given the gate counts of base gadgets (and associated $\lambda_{\max}$), the higher the amplification order d, the lower the complexity exponent e. It is therefore of interest to search for gadgets with maximal amplification orders. In this regards, the following lemma gives a generic upper bound on the amplification order. The proof is available in the full version of [BRT21] (see Appendix G).

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.

Lemma 6. *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 as*

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

We deduce from the above lemma that for a randomized arithmetic circuit composed of 1-input (copy) and 2-input (operation) gadgets, the amplification order is upper bounded as

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

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

8.4. Generic constructions of RPE gadgets

This section provides constructions of RPE gadgets which achieve the maximal amplification order for any number of shares n. We first introduce natural constructions of copy and addition gadgets from a refresh gadget and show that these constructions achieve maximal amplification order for specific refresh gadgets. We then argue that common multiplication gadgets from the literature cannot reach maximal amplification order and introduce a new construction to fill this gap.

8.4.1. Generic copy and addition gadgets

Tight random probing expandability. Let us first introduce a tighter version of the RPE security property that shall be instrumental to obtain generic RPE constructions. 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 $\mathcal{F}_j$ in Definition 20 is defined for every j as

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

We then have the two following relations with the standard RPE notion:

1. G is (t, f)-TRPE of amp. order $d \implies G$ is (t, f')-RPE of amp. order $d' \geqslant d$,
2. G is (t, f)-TRPE of amp. order $d = t + 1 \iff G$ is (t, f)-RPE of amp. order $d = t + 1$.

A formal definition of TRPE and a proof of the above relations can be found in [BRT21] (see Appendix G).

Generic copy gadget. Our generic copy gadget simply consists in applying a refresh gadget G_{R} to the input sharing twice in order to obtain two fresh copies. Formally, from G_{R}, the generic copy gadget G_{λ} is defined as

$$G_{\lambda}(\boldsymbol{x}) = (G_{\mathsf{R}}(\boldsymbol{x}), G_{\mathsf{R}}(\boldsymbol{x})) \ . \tag{8.6}$$

We have the following lemma (whose proof is given in the full version of [BRT21] – see Appendix G).

Lemma 7. *Let G_{R} be an n-share (t, f)-TRPE refresh gadget of amplification order d. Then, the copy gadget G_{λ} displayed in Equation 8.6 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. If the refresh gadget is RPE with amplification order reaching the upper bound for 1-input gadgets, which is $d = t + 1 = 2(n - t) = \frac{2(n+1)}{3}$, then the copy gadget is also RPE with same (maximal) amplification order.

Generic addition gadget. Our generic addition gadget simply consists in applying a refresh gadget G_{R} to each of the two input sharings before adding them. Formally, from G_{R}, the generic addition gadget $G_{\oplus}$ is defined as

$$G_{\oplus}(\boldsymbol{x}, \boldsymbol{y}) = G_{\mathsf{R}}(\boldsymbol{x}) + G_{\mathsf{R}}(\boldsymbol{y}) \ . \tag{8.7}$$

We have the following lemma (whose proof is given in the full version of [BRT21] – see Appendix G).

Lemma 8. *Let G_{R} be an n-share refresh gadget and let $G_{\oplus}$ be the corresponding addition gadget displayed in Equation 8.7. Then if G_{R} is (t, f)-RPE (resp. (t, f)-TRPE) of amplification order d, then $G_{\oplus}$ is (t, f')-RPE (resp. (t, f')-TRPE) for some f' of amplification order $d' \geqslant \lfloor \frac{d}{2} \rfloor$.*

The above lemma shows that an RPE refresh gadget of amplification order d directly yields an 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 $\frac{n+1}{2}$.

Although Lemma 8 is insufficient to ensure an optimal amplification order for the generic addition gadget, it can still be obtained for specific instantiations of the refresh gadget G_{R} as we explicit hereafter.

Instantiation from ISW refresh gadget. Let us recall that the ISW refresh gadget consists in evaluating $G_{\mathsf{R}} : \boldsymbol{x} \mapsto G_{\otimes}(\boldsymbol{x}, (1, 0, \ldots, 0))$ where $G_{\otimes}$ is the ISW multiplication gadget. This gadget is depicted in Algorithm 3 (see Section 3.4.3). We have the following result:

Theorem 10. *Let $t, n \in \mathbb{N}$ with $t \leqslant n - 2$. Let G_{R} be the n-share ISW refresh gadget, G_{λ} be the copy gadget of Equation 8.6 and $G_{\oplus}$ the addition gadget of Equation 8.7. We have*

1. *G_{R} is (t, f)-TRPE of amplification order $d = \min(t + 1, n - t)$;*
2. *G_{λ} is (t, f)-RPE of amplification order $d = \min(t + 1, n - t)$;*
3. *$G_{\oplus}$ is (t, f)-RPE of amplification order $d = \min(t + 1, n - t)$.*

The first item is proven in [BRT21, Corollary 3], the second item holds from the first one and Lemma 7 above, while the third item is a consequence of [BRT21, Lemma 11] (see Appendix G).

Instantiation from BCPZ refresh gadget. In [BCPZ16a], Battistello, Coron, Prouff and Zeitoun introduced a refresh gadget with quasilinear complexity $\mathcal{O}(n \log n)$. This refresh gadget (already overviewed in Section 5.3.1) is defined recursively as:

$$G_{\mathsf{R}}(\boldsymbol{x}) = \big(G_{\mathsf{R}}(\boldsymbol{x}_1 + \boldsymbol{r}) + \boldsymbol{s} \parallel G_{\mathsf{R}}(\boldsymbol{x}_2 - \boldsymbol{r}) - \boldsymbol{s}\big) \tag{8.8}$$

for any $\boldsymbol{x} = (\boldsymbol{x}_1 \parallel \boldsymbol{x}_2) \in \mathbb{K}^n$ with $n > 1$, with randomly sampled $\boldsymbol{r} \leftarrow \mathbb{K}^{n/2}$ and $\boldsymbol{s} \leftarrow \mathbb{K}^{n/2}$, and $G_{\mathsf{R}}(\boldsymbol{x}) = \boldsymbol{x}$ for $n = 1$. While this definition implicitly assumes that n is a power of 2, this refresh gadget is defined more generally for any $n \in \mathbb{N}$ (see description in [BCPZ16a]).

We have the following result:

Theorem 11. *Let $t, n \in \mathbb{N}$ with $t \leqslant n - 1$. Let G_{R} be the n-share BCPZ refresh gadget, G_{λ} be the copy gadget of Equation 8.6 and $G_{\oplus}$ the addition gadget of Equation 8.7. We have*

1. *G_{R} is (t, f)-TRPE of amplification order $d = \min(t + 1, n - t)$;*
2. *G_{λ} is (t, f)-RPE of amplification order $d = \min(t + 1, n - t)$;*

3. $G_\oplus$ *is* (t, f)*-RPE of amplification order* $d = \min(t+1, n-t)$.

The first item is proven in [BRTV21, Lemma 3], the second item holds from the first one and Lemma 7 above, while the third item is a consequence of [BRTV21, Lemmas 2 & 3].

The above theorems provide concrete instantiations of the generic copy and addition gadgets (based on ISW and BCPZ refresh gadgets) reaching the upper bound for the amplification order (for 2-input gadgets) for any number of shares n. We note that the instantiations from the ISW refresh gadget have quadratic complexity $\mathcal{O}(n^2)$ while the instantiations from the BCPZ refresh gadget enjoy quasilinear complexity $\mathcal{O}(n \log n)$; one might therefore prefer the latter.

Remark 8. The generic addition gadget of Equation 8.7 might more generally reach the upper bound on the amplification order from other refresh gadget constructions. We introduce in [BRTV21] the *strong TRPE2* notion which, together with TRPE, is sufficient for G_{R} to ensure a maximal amplification order for $G_\oplus$.

8.4.2. Multiplication gadget with maximal amplification order

Constructing a multiplication gadget with maximal amplification order is tricky. As a matter of fact, we show in [BRT21] (see Appendix G) that the widely spread ISW multiplication gadget (see Algorithm 1) achieves RPE but with multiplication order $\min(t+1, n-t)/2$ which is not optimal. We further show in [BRT21, Lemma 2] that any multiplication gadget which computes the cross products of the input shares have an amplification order upper bounded by $\min((t+1)/2, n-t)$ which is strictly lower than the upper bound of Lemma 6 for 2-input gadgets. In order to reach the maximal amplification order, we must hence build a multiplication gadget avoiding a direct cross-product of 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 asymptotic complexity of the expanding compiler (see Section 8.3). This suggests that we can be greedy in terms of randomness to reach the maximal amplification order.

From those observations, we describe hereafter a multiplication gadget $\boldsymbol{z} = G_\otimes(\boldsymbol{x}, \boldsymbol{y})$ which achieves the maximal amplification order $\min(t+1, n-t)$ by making greedy use of randomness and input refreshing. We first describe our core 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:

$$M = \begin{pmatrix} x_1 \cdot y_1 & x_1 \cdot y_2 & \cdots & x_1 \cdot y_n \\ x_2 \cdot y_1 & x_2 \cdot y_2 & \cdots & x_2 \cdot y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_n \cdot y_1 & x_n \cdot y_2 & \cdots & x_n \cdot y_n \end{pmatrix}$$

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 $(z_1, \ldots, z_n) = V + X$.

In order to further increase the maximal amplification order attainable by the gadget, we need to avoid performing a direct product of shares. To this aim, we add a pre-processing phase to the gadget using a refresh gadget G_{R}. Specifically, we refresh the input $(y_1, \ldots, y_n)$ each time it is used. In other terms, each row of the matrix M uses a fresh copy of $(y_1, \ldots, y_n)$ produced by a new call to G_{R}, that is

$$M = \begin{pmatrix} x_1 \cdot y_1^{(1)} & x_1 \cdot y_2^{(1)} & \cdots & x_1 \cdot y_n^{(1)} \\ x_2 \cdot y_1^{(2)} & x_2 \cdot y_2^{(2)} & \cdots & x_2 \cdot y_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n \cdot y_1^{(n)} & x_n \cdot y_2^{(n)} & \cdots & x_n \cdot y_n^{(n)} \end{pmatrix}$$

where $(y_1^{(j)}, \ldots, y_n^{(j)})$, $j \in [n]$, are the n independent refreshings of the input $(y_1, \ldots, y_n)$.

With this refreshing scheme, we avoid using the same share more than once for one of the two input sharings. In addition, the asymptotic computational overhead of these n additional refreshes is negligible compared to the joint contribution of the copy and addition gadgets.

We have the following result [BRT21, Corollary 6] (see Appendix G):

Theorem 12. *Let $t \leqslant n - 1$. Let G_{R} be a (t, f')-TRPE refresh gadget of amplification order $d' \geqslant \min(t+1, n-t)$. The above multiplication gadget using G_{R} as refresh gadget is (t, f)-RPE of the amplification order $d = \min(t+1, n-t)$.*

We note that the weaker notion of TRPE1 is sufficient from G_{R} in the above statement (see [BRT21] for details). As a corollary of the above theorem and the previous results on ISW and BCPZ refresh gadgets, the instantiation of our construction with any of these two refresh gadgets yields an RPE multiplication gadget with maximal amplification order.

In [BRT21] (see Appendix G), we provide an asymptotic analysis of the expanding compiler based on the generic gadgets introduced in this section. We show that, while increasing the number of shares n, the asymptotic complexity of the expanding compiler converges towards $\mathcal{O}(\kappa^2)$. On the other hand, the tolerated leakage probability decreases while n grows. In the next section, we provide concrete instantiations of the RPE framework for small gadgets for which we can exhibit the tolerated leakage probability.

8.5. Efficient instantiation with small RPE gadgets

This section presents constructions of small RPE gadgets for copy, addition, and multiplication operations with a low number of shares. It can be checked that RPE security with a relevant amplification order (*i.e.* strictly greater than 1) cannot be obtained by a gadget with less than 3 shares. Then, as explained in Section 8.3.4, the highest amplification orders can only be achieved for gadgets with an odd number of shares. We therefore consider the cases of 3-share and 5-share gadgets. Each one of these gadgets is experimentally verified using the VRAPS verification tool [BCP+20] from which we derive the tolerated leakage probability as well as the complexity of the underlying expanding compiler.

Copy and addition gadgets. For the construction of small 3-share and 5-share addition and copy gadgets, we use the generic constructions depicted in Equation 8.6 and Equation 8.7 (Section 8.4.1)

which are based a refresh gadget as building block. We hence start by looking for refresh gadgets that have a good complexity in terms of gate count, and achieve the upper bound on the amplification order for the specific cases 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 Section 8.4.2 which, to the best of our knowledge, is the only multiplication gadget achieving the maximal amplification order for any number of shares, and specifically for 3-share and 5-share constructions. As for the refresh gadget G_{R} 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.

While the multiplication gadget from Section 8.4.2 reaches the desired amplification order, we add another pre-processing phase. In addition to the n refreshes performed on the second input $\boldsymbol{b}$, we add another single refresh of the input $\boldsymbol{a}$ before computing the cross-products, using the same refresh gadget G_{R}. Refreshing the input $\boldsymbol{a}$ before usage experimentally showed a further increase in the maximum tolerated leakage probability, by adding more randomness to the input shares before computing the cross-product matrix M.

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

8.5.1. Three-share gadgets

For the 3-share instances, we use the following refresh gadget as building block:

$$\begin{aligned} G_{\mathsf{R}} : z_1 &\leftarrow r_1 + x_1 \\ z_2 &\leftarrow r_2 + x_2 \\ z_3 &\leftarrow (r_1 + r_2) + x_3. \end{aligned}$$

This refresh is sufficient to reach the upper bounds on the amplification orders for the three gadgets while being more efficient in terms of gate count than *e.g.* the ISW refresh gadget.

Results. Table 8.1 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.

Table 8.1.: 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_{R}	$(4, 2, 0, 2)$	2	-5.14
$G_{\oplus}$	$(11, 4, 0, 4)$	2	-4.75
G_{λ}	$(8, 7, 0, 4)$	2	-7.50
$G_{\otimes}$	$(40, 29, 9, 17)$	2	-7.41
Compiler	$\mathcal{O}(\lvert C\rvert \cdot \kappa^{\mathbf{3.9}})$	**2**	$\mathbf{-7.50}$

8.5.2. Five-share gadgets

For the 5-shares instances, we use the *circular refresh gadget* described in [BBD+20, BDF+17] (a.k.a. *block refresh gadget*):

$$\begin{aligned} G_{\mathsf{R}} : z_1 &\leftarrow (r_1 + r_2) + x_1 \\ z_2 &\leftarrow (r_2 + r_3) + x_2 \\ z_3 &\leftarrow (r_3 + r_4) + x_3 \\ z_4 &\leftarrow (r_4 + r_5) + x_4 \\ z_5 &\leftarrow (r_5 + r_1) + x_5. \end{aligned}$$

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 [BBD+20, BDF+17]), it proves to be enough to achieve the necessary amplification order for our 5-share constructions. We use a variant of the original version (also suggested in [BBD+20]): 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 $\{x_i\}_{1 \leqslant i \leqslant n}$ and the final output variables $\{z_i\}_{1 \leqslant i \leqslant n}$. Intuitively, this trick allows to have less tuples of intermediate variables resulting in simulation failures because there are less variables that could leak information about the input. This is validated experimentally since we obtain better results in terms of amplification order and tolerated leakage probability for small gadgets.

Results. Table 8.2 gives the results for the above gadgets obtained through the VRAPS tool.

Table 8.2.: 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_{R}	$(10, 5, 0, 5)$	3	-4.83
$G_{\oplus}$	$(25, 10, 0, 10)$	3	$[-6.43, -3.79]$
$G_{\wedge}$	$(20, 15, 0, 10)$	3	$[-6.43, -5.78]$
$G_{\otimes}$	$(130, 95, 25, 55)$	3	$[-12.00, -6.03]$
Compiler	$\mathcal{O}(\lvert C\rvert \cdot \kappa^{\mathbf{3.23}})$	**3**	$[\mathbf{-12.00}, \mathbf{-6.03}]$

From Table 8.1 and Table 8.2, 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.

8.6. Conclusion and related works

In this chapter, we have presented the random probing expandability (RPE) framework introduced in [BCP+20] and inspired from the work of Ananth, Ishai and Sahai [AIS18]. We have also introduced

generic gadget constructions achieving the RPE notion as well as concrete instantiations of the RPE framework based on small gadgets. Those instantiations tolerate the highest leakage probability exhibited so far, although we expect the state of the art to evolve soon, possibly with better instantiations of the RPE framework. In a recent follow-up work with Belaïd, Taleb and Vergnaud [BRTV21], we have pushed this approach one step further by exhibiting new RPE gadget constructions with quasilinear complexity (under some conditions on the base field). We have further formalized the dynamic expanding compiler which adapts the set of base gadgets throughout the expansion. This approach can make the most of two different sets of base gadgets, inheriting from the highest tolerated leakage probability on one hand and from the best asymptotic complexity on the other hand. In another follow-up work with Belaïd, Mercadier and Taleb [BMRT22], we have introduced a new formal verification tool, named IronMask, which provides complete RPC/RPE verification for a large class of relevant gadgets and which is several orders of magnitude faster than VRAPS. Other previous works have achieved constant leakage probability in the random probing model. Namely, Ajtai proposed a scheme based on expander graphs [Ajt11], which was further improved by Andrychowicz, Dziembowski and Faust in [ADF16]. The concrete instantiation and exact leakage probability still remain to be investigated for these schemes. More generally, the research of new schemes, or improved RPE gadgets, achieving leakage probability closer to one as well as better asymptotic complexity (while relaxing the constraint on the base field) is an interesting direction for further researches.

Part V.

Conclusion

Conclusion

Provable security has become a *de facto* requirement for cryptographic algorithms and protocols in the scientific literature as well as in the industry. For cryptographic implementations on the other hand, expectations and achievements in terms of formal security guaranties are much less ambitious. And yet, side-channel attacks, and in particular power and electromagnetic attacks, are truly devastating for the security of cryptographic implementations embedded in somehow accessible devices. Motivated by this challenge, we have presented in this thesis several contributions on the provable security of cryptographic implementations against those attacks.

Soon after the publication of differential power analysis [KJJ99], masking, a.k.a. secret sharing at the implementation level, was identified as a sound approach to protect cryptographic implementations in the presence of power and electromagnetic leakages [CJRR99, GP99]. For one decade, masking countermeasures were *ad hoc* and mostly limited to a single mask (a.k.a. first-order masking). In 2010, we have shown that the probing security framework and the probing-secure scheme both introduced in the seminal work of Ishai, Sahai and Wagner [ISW03] could provide a practical answer to the open issue of masking schemes secure against higher-order side-channel attacks. From this observation, our humble contribution was to make the so-called ISW scheme efficiently applicable to AES [RP10] and to any SPN-like block ciphers [CGP+12] (see Chapter 3). Many subsequent works have followed a similar approach and proposed further probing secure masking schemes, masking composition notions, verification methods and optimized implementations. As of today, probing security has become the common approach to reason about and formally prove the security of masked implementations.

While the probing security paradigm provides a first level of provable security against side-channel attacks, it fails to fully capture the ability of an adversary monitoring the power or electromagnetic leakage of a cryptographic computation. Indeed, in the probing model, the adversary is restricted to exploit the leakage on a limited number of intermediate variables of the computation, whereas in practice, a side-channel adversary gets some leakage on all the intermediate variables. To fill this gap, we have introduced the noisy leakage model [PR13] (see Chapter 4). This model captures any leakage distribution by quantifying its amount of noise through a simple parameter called the bias. In a remarkable follow up work [DDF14], Duc, Dziembowski and Faust have unified our noisy model to strong versions of the probing model, namely the region probing and random probing models. Thanks to their result, region or random probing secure implementations inherit noisy leakage security, while designing and proving masking schemes in those models is conceptually much simpler.

Many research works over the last decade have focused on the design, implementation, composition and verification of masking schemes in the probing model and much less has been done in the region and random probing models. Only a few schemes have been proposed which achieve provable security in these models and most of them are theoretical and without concrete instantiation or concrete evaluation of the tolerated leakage rate. Motivated by this challenge, we have introduced new composition approaches and new provably secure masking schemes in some recent works [GJR18, BCP+20, GPRV21, BRT21]. Chapter 5 and Chapter 6 present the masking composition approaches introduced in these works. Chapter 7 describes a region probing secure scheme with quasilinear complexity (but which only tolerates quasiconstant leakage rate). Chapter 8 introduces a framework to bootstrap small masking gadgets into random probing secure implementations. Our instantiations of this framework achieve concrete leakage rates, which are the highest exhibited so far.

In view of these developments, many interesting and challenging research directions remain open. A first concrete topic for further investigations is the research of new schemes tolerating higher leakage rates (as close to 1 as possible) as well as better asymptotic or practical complexity. A promising

approach is to search for better instantiations of the random probing expansion framework. Likewise, a lot remains to be done on the efficient implementation and optimization of existing schemes. As more constructions and improvements will emerge, the community might further investigate formal verification and code generation tools for masked implementations with provable security in the noisy leakage model. Another worthwhile direction is to tighten the gap between the theory and practice of provably secure masked implementations. In particular, much more work would be needed on the implicit assumptions underlying the noisy leakage model, namely data isolation and noise independence, to either enforce them in practice or to relax them with more robust designs. A related practical issue is the conception of efficient and reliable methods to quantify the noise parameters and to characterize a wide range of devices in the noisy model.

Finally, and as a more general perspective, while we now have some proof-of-concept provably secure implementations with respect to passive side-channel attacks, a more challenging question is that of active attacks. In particular, *fault attacks* are known to be equally devastating against unprotected cryptographic implementations [BDL97, BS97] and while many works on fault attacks and countermeasures have been published in the literature, formal models and provably secure designs would deserve further investigations. A particularly challenging issue is to capture a wide range of fault attacks as well as combined side-channel and fault attacks. While pushing the adversary's tampering capabilities further, one might consider the so-called *white-box model* in which an adversary has complete control over the target implementation [CEJv03, DLPR14]. All attempts to provide secure implementations in this model have failed. The design of secure white-box cryptography, possibly in weaken adversarial models (which are still to be defined), is a stimulating open issue.

Bibliography

[ADF16] 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.

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of DES and AES, secure against some attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 309–318. Springer, Heidelberg, May 2001.

[AIS18] 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.

[Ajt11] 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.

[BBB+20] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Trans. Symm. Cryptol.*, 2020(S1):295–349, 2020.

[BBC+19] Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and François-Xavier Standaert. maskVerif: Automated verification of higher-order masking in presence of physical defaults. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part I*, volume 11735 of *LNCS*, pages 300–318. Springer, Heidelberg, September 2019.

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 457–485. Springer, Heidelberg, April 2015.

[BBD+16] 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.

[BBD+20] 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.

[BBP+16] 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.

[BBP+17] 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.

[BCP+20] 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.

[BCPZ16a] 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.

[BCPZ16b] 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.

[BDF+17] 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.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997.

[BDM+20] Sonia Belaïd, Pierre-Évariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff. Tornado: Automatic generation of probing-secure masked bitsliced implementations. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 311–341. Springer, Heidelberg, May 2020.

[BFO+21] Gianluca Brian, Antonio Faonio, Maciej Obremski, João L. Ribeiro, Mark Simkin, Maciej Skórski, and Daniele Venturi. The mother of all leakages: How to simulate noisy leakages via bounded leakage (almost) for free. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 408–437. Springer, Heidelberg, October 2021.

[BGI+18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 321–353. Springer, Heidelberg, April / May 2018.

[BGK04] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably secure masking of AES. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 69–83. Springer, Heidelberg, August 2004.

[BGR18] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. Tight private circuits: Achieving probing security with the least refreshing. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 343–372. Springer, Heidelberg, December 2018.

[BK21] Nicolas Bordes and Pierre Karpman. Fast verification of masking schemes in characteristic two. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 283–312. Springer, Heidelberg, October 2021.

[BMRT22] Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. IronMask: Versatile Verification of Masking Security. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 142–160. IEEE, 2022.

[BRT21] Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 313–343. Springer, Heidelberg, October 2021.

[BRTV21] Sonia Belaïd, Matthieu Rivain, Abdul Rahman Taleb, and Damien Vergnaud. Dynamic random probing expansion with quasi linear asymptotic complexity. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 157–188. Springer, Heidelberg, December 2021.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997.

[Can89] David G. Cantor. On arithmetical algorithms over finite fields. *J. Comb. Theory, Ser. A*, 50(2):285–300, 1989.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *CHES 2000*, volume 1965 of *LNCS*, pages 252–263. Springer, Heidelberg, August 2000.

[CDI+13] Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 185–202. Springer, Heidelberg, August 2013.

[CEJv03] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 250–270. Springer, Heidelberg, August 2003.

[CFOS21] Gaëtan Cassiers, Sebastian Faust, Maximilian Orlt, and François-Xavier Standaert. Towards tight random probing security. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 185–214, Virtual Event, August 2021. Springer, Heidelberg.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for S-boxes. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 366–384. Springer, Heidelberg, March 2012.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on a sum of observations. *Ann. Math. Statis.*, 23(4):493–507, 1952.

[CJRR99] 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.

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and improvement of the random delay countermeasure of CHES 2009. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 95–109. Springer, Heidelberg, August 2010.

[Cor14] 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.

[Cor18] Jean-Sébastien Coron. Formal verification of side-channel countermeasures via elementary circuit transformations. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 65–82. Springer, Heidelberg, July 2018.

[CPR07] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side channel cryptanalysis of a higher order masking scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 28–44. Springer, Heidelberg, September 2007.

[CPR12] Jean-Sébastien Coron, Emmanuel Prouff, and Thomas Roche. On the use of shamir's secret sharing against side-channel analysis. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2012.

[CPRR14] 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.

[CPRR15] Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 742–763. Springer, Heidelberg, August 2015.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 13–28. Springer, Heidelberg, August 2003.

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 170–187. Springer, Heidelberg, September 2014.

[CRZ13] Guilhem Castagnos, Soline Renner, and Gilles Zémor. High-order masking by using coding theory and its application to AES. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 193–212. Springer, Heidelberg, December 2013.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[CS21a] Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR TCHES*, 2021(2):136–158, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8790.

[CS21b] Jean-Sébastien Coron and Lorenzo Spignoli. Secure wire shuffling in the probing model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 215–244, Virtual Event, August 2021. Springer, Heidelberg.

[DDF14] 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.

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 230–247. Springer, Heidelberg, March 2012.

[DFS15] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 159–188. Springer, Heidelberg, April 2015.

[DLPR14] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-box security notions for symmetric encryption schemes. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 247–264. Springer, Heidelberg, August 2014.

[DLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 225–244. Springer, Heidelberg, March 2006.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th FOCS*, pages 293–302. IEEE Computer Society Press, October 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 21–40. Springer, Heidelberg, August 2010.

[Eve64] J. Eve. The evaluation of polynomials. *Comm. ACM*, 6(1):17–21, 1964.

[FGP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR TCHES*, 2018(3):89–120, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7270.

[FIP99a] FIPS PUB 202. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology, August 1999.

[FIP99b] FIPS PUB 46-3. *Data Encryption Standard (DES)*. National Institute of Standards and Technology, October 1999.

[GGNS13] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block ciphers that are easier to mask: How far can we go? In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 383–399. Springer, Heidelberg, August 2013.

[GJK+20] Dahmun Goudarzi, Jeremy Jean, Stefan Kölbl, Thomas Peyrin, Matthieu Rivain, Yu Sasaki, and Siang Meng Sim. Pyjamask: Block cipher and authenticated encryption with highly efficient masked implementation. *IACR Trans. Symm. Cryptol.*, 2020(S1):31–59, 2020.

[GJR18] Dahmun Goudarzi, Antoine Joux, and Matthieu Rivain. How to securely compute with noisy leakage in quasilinear complexity. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 547–574. Springer, Heidelberg, December 2018.

[GJRS18] Dahmun Goudarzi, Anthony Journault, Matthieu Rivain, and François-Xavier Standaert. Secure multiplication for bitslice higher-order masking: Optimisation and comparison. In Junfeng Fan and Benedikt Gierlichs, editors, *COSADE 2018*, volume 10815 of *LNCS*, pages 3–22. Springer, Heidelberg, April 2018.

[GLSV15] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 18–37. Springer, Heidelberg, March 2015.

[GM10] S. Gao and T. Mateer. Additive fast Fourier transforms over finite fields. *IEEE Transactions on Information Theory*, 56(12):6265–6272, Dec 2010.

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir's secret sharing scheme. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 79–94. Springer, Heidelberg, September / October 2011.

[GP99] 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.

[GPRV21] Dahmun Goudarzi, Thomas Prest, Matthieu Rivain, and Damien Vergnaud. Probing security through input-output separation and revisited quasilinear masking. *IACR TCHES*, 2021(3):599–640, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8987.

[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *53rd FOCS*, pages 31–40. IEEE Computer Society Press, October 2012.

[GR16] Dahmun Goudarzi and Matthieu Rivain. On the multiplicative complexity of Boolean functions and bitsliced higher-order masking. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 457–478. Springer, Heidelberg, August 2016.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 567–597. Springer, Heidelberg, April / May 2017.

[GRVV17] Dahmun Goudarzi, Matthieu Rivain, Damien Vergnaud, and Srinivas Vivek. Generalized polynomial decomposition for S-boxes with application to side-channel countermeasures. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 154–171. Springer, Heidelberg, September 2017.

[HM00] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, January 2000.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS 06*, volume 3989 of *LNCS*, pages 239–252. Springer, Heidelberg, June 2006.

[ISW03] 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.

[JS17] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 623–643. Springer, Heidelberg, September 2017.

[KJJ98a] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Improved des and other cryptographic process with leak minimization for smartcards and other cryptosystems. U.S. Patent. WO 99/67919. 3 June 1998, 1998.

[KJJ98b] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Using unpredictable information to minimize leakage from smartcards and other cryptosystems. U.S. Patent. WO 99/63696. 3 June 1998, 1998.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

[Knu62] Donald E. Knuth. Evaluation of polynomials by computers. *Comm. ACM*, 5(12):595–599, 1962.

[Knu88] D.E. Knuth. *The Art of Computer Programming*, volume 2. Addison Wesley, third edition, 1988.

[Koc96] 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.

[KR18] 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.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER - statistical independence and leakage verification. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 787–816. Springer, Heidelberg, December 2020.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Çetin Kaya Koç and Christof Paar, editors, *CHES 2000*, volume 1965 of *LNCS*, pages 238–251. Springer, Heidelberg, August 2000.

[Mes01] Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 150–164. Springer, Heidelberg, April 2001.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, Heidelberg, 2007.

[MOPT12] Andrew Moss, Elisabeth Oswald, Dan Page, and Michael Tunstall. Compiler assisted masking. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 58–75. Springer, Heidelberg, September 2012.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 351–365. Springer, Heidelberg, February 2005.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, Heidelberg, February 2004.

[MSQ07] François Macé, François-Xavier Standaert, and Jean-Jacques Quisquater. Information theoretic evaluation of side-channel resistant logic styles. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 427–442. Springer, Heidelberg, September 2007.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 18–35. Springer, Heidelberg, August 2009.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order DPA attacks for masked smart card implementations of block ciphers. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 192–207. Springer, Heidelberg, February 2006.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A side-channel analysis resistant description of the AES S-box. In Henri Gilbert and Helena Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 413–423. Springer, Heidelberg, February 2005.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a Rényi day. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 683–712. Springer, Heidelberg, August 2019.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, Heidelberg, April 2009.

[PM05] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In Josyula R. Rao and Berk Sunar, editors, *CHES 2005*, volume 3659 of *LNCS*, pages 172–186. Springer, Heidelberg, August / September 2005.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 63–78. Springer, Heidelberg, September / October 2011.

[PR13] 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.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[PRC12] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - a block cipher allowing efficient higher-order side-channel resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 12*, volume 7341 of *LNCS*, pages 311–328. Springer, Heidelberg, June 2012.

[RDP08] Matthieu Rivain, Emmanuelle Dottax, and Emmanuel Prouff. Block ciphers implementations provably secure against second order side channel analysis. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 127–143. Springer, Heidelberg, February 2008.

[RP10] 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.

[RV13] Arnab Roy and Srinivas Vivek. Analysis and improvement of the generic higher-order masking scheme of FSE 2012. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 417–434. Springer, Heidelberg, August 2013.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SÖP04] François-Xavier Standaert, Siddika Berna Örs, and Bart Preneel. Power analysis of an FPGA: Implementation of Rijndael: Is pipelining a DPA countermeasure? In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 30–44. Springer, Heidelberg, August 2004.

[SP06] 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.

[WVGX15] Junwei Wang, Praveen Kumar Vadnala, Johann Großschädl, and Qiuliang Xu. Higher-order masking in practice: A vector implementation of masked AES for ARM NEON. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 181–198. Springer, Heidelberg, April 2015.

[WZ88] Y. Wang and X. Zhu. A fast algorithm for the Fourier transform over finite fields and its VLSI implementation. *IEEE Journal on Selected Areas in Communications*, 6(3):572–577, April 1988.

Part VI.

Appended publications

Appendix A

Provably Secure Higher-Order Masking of AES

Hereafter is appended the full version of our paper [RP10], joint work with Emmanuel Prouff, published at **CHES 2010**.

Provably Secure Higher-Order Masking of AES*

Matthieu Rivain[1] and Emmanuel Prouff[2]

[1] CryptoExperts
matthieu.rivain@cryptoexperts.com
[2] Oberthur Technologies
e.prouff@oberthur.com

Abstract. Implementations of cryptographic algorithms are vulnerable to Side Channel Analysis (SCA). To counteract it, masking schemes are usually involved which randomize key-dependent data by the addition of one or several random value(s) (the *masks*). When dth-order masking is involved (*i.e.* when d masks are used per key-dependent variable), the complexity of performing an SCA grows exponentially with the order d. The design of generic dth-order masking schemes taking the order d as security parameter is therefore of great interest for the physical security of cryptographic implementations. This paper presents the first generic dth-order masking scheme for AES with a provable security and a reasonable software implementation overhead. Our scheme is based on the hardware-oriented masking scheme published by Ishai *et al.* at Crypto 2003. Compared to this scheme, our solution can be efficiently implemented in software on any general-purpose processor. This result is of importance considering the lack of solution for $d \geqslant 3$.

1 Introduction

Side Channel Analysis exploits information that leaks from physical implementations of cryptographic algorithms. This leakage (*e.g.* the power consumption or the electro-magnetic emanations) may indeed reveal information on the data manipulated by the implementation. Some of these data are *sensitive* in the sense that they are related to the secret key, and the leaking information about them enables efficient key-recovery attacks [7, 19].

Due to the very large variety of side channel attacks reported against cryptosystems and devices, important efforts have been done to design countermeasures with provable security. They all start from the assumption that a cryptographic device can keep at least some secrets and that only computation leaks [25]. Based on these assumptions, two main approaches have been followed. The first one consists in designing new cryptographic primitives inherently resistant to side channel attacks. In [25], a very powerful side channel adversary is considered who has access to the whole internal state of the ongoing computation. In such a model, the authors show that if a *physical* one-way permutation exists which does not leak any information, then it can be used in the pseudo-random number generator (PRNG) construction proposed in [4] to give a PRNG provably secure against the aforementioned side channel adversary. Unfortunately, no such leakage-resilient one-way permutation is known at this day. Besides, the obtained construction is quite inefficient since each computation of the one-way permutation produces one single random bit. To get more practical constructions, further works focused on designing primitives secure against a *limited* side channel adversary [13]. The definition of such a limited adversary is inspired by the *bounded retrieval model* [10, 22] which assumes that the device leaks a limited amount of information about its internal state for each elementary computation. In such a setting, the block cipher based PRNG construction proposed in [30] is provably secure assuming that the underlying cipher is *ideal*. Other

* Full version of the paper published in the proceedings of CHES 2010.

constructions were proposed in [13,31] which do not require such a strong assumption but are less efficient [40]. The main limitations of these constructions is that they do not enable the choice of an initialization vector (otherwise the security proofs do not hold anymore) which prevents their use for encryption with synchronization constraints or for challenge-response protocols [40]. Moreover, as they consist in new constructions, these solutions do not allow for the protection of the implementation of standard algorithms such as DES or AES [14,15].

The second approach to design countermeasures provably secure against side channel attacks consists in applying *secret sharing schemes* [2,39]. In such schemes, the sensitive data is randomly split into several shares in such a way that a chosen number (called the *threshold*) of these shares is required to retrieve any information about the data. When the SCA threat appeared, secret sharing was quickly identified as a pertinent protection strategy [6, 17] and numerous schemes (often called *masking schemes*) were published that were based on this principle (see for instance [1,3,18,23,26,29,34,38]). Actually, this approach is very close to the problem of defining Multi Party Communication (MPC) schemes (see for instance [9,12]) but the resources and constraints differ in the two contexts (*e.g.* MPC schemes are often based on a *trusted dealer* who does not exist in the SCA context). A first advantage of this approach is that it can be used to secure standard algorithms such as DES and AES. A second advantage is that *dth-order masking schemes*, for which sensitive data are split into $d + 1$ shares (the threshold being $d + 1$), are sound countermeasures to SCA in *realistic leakage model*. This fact has been formally demonstrated by Chari *et al.* [6] who showed that the complexity of recovering information by SCA on a bit shared into several pieces grows exponentially with the number of shares. As a direct consequence of this work, the number of shares (or equivalently of masks) in which sensitive data are split is a sound security parameter of the resistance of a countermeasures against SCA.

The present paper deals with the problem of defining an efficient masking scheme to protect the implementation of the AES block cipher [11]. Until now, most of works published on this subject have focused on first-order masking schemes where sensitive variables are masked with a single random value (see for instance [1, 3, 23, 26, 29]). However, this kind of masking have been shown to be efficiently breakable in practice by *second-order SCA* [24,27,42]. To counteract those attacks, *higher-order masking schemes* must be used but a very few have been proposed. A first method has been introduced by Ishai *et al.* [18] which enables to protect an implementation at any chosen order. Unfortunately, it is not suited for software implementations and it induces a prohibitive overhead for hardware implementations. A scheme devoted to secure the software implementation of AES at any chosen order has been proposed by Schramm and Paar [38] but it was subsequently shown to be secure only in the second-order case [8]. Alternative second-order masking schemes with provable security were further proposed in [34], but no straightforward extension of them exist to get efficient and secure masking scheme at any order. Actually, at this day, no method exists in the literature that enables to mask an AES implementation at any chosen order $d \geqslant 3$ with a practical overhead; the present paper fills this gap.

2 Preliminaries on Higher-Order Masking

2.1 Basic Principle

When higher-order masking is involved to secure the physical implementation of a cryptographic algorithm, every sensitive variable x occurring during the computation is randomly split into $d+1$ shares $x_0, \ldots, x_d$ in such a way that the following relation is satisfied for a group operation $\perp$:

$$x_0 \perp x_1 \perp \cdots \perp x_d = x \ . \tag{1}$$

In the rest of the paper, we shall consider that $\perp$ is the exclusive-or (XOR) operation denoted by $\oplus$. Usually, the d shares $x_1, \ldots, x_d$ (called *the masks*) are randomly picked up and the last one x_0 (called *the masked variable*) is processed such that it satisfies (1). When d random masks are involved per sensitive variable the masking is said to be *of order d*.

Assuming that the masks are uniformly distributed, masking renders every intermediate variable of the computation statistically independent of any sensitive variable. As a result, classical side channel attacks exploiting the leakage related to a single intermediate variable are not possible anymore. However, a dth-order masking is always theoretically vulnerable to $(d+1)$*th-order SCA* which exploits the leakages related to $d+1$ intermediate variables at the same time [24, 37, 38]. Indeed, the leakages resulting from the $d+1$ shares (i.e. the masked variable and the d masks) are jointly dependent on the sensitive variable. Nevertheless, such attacks become impractical as d increases, which makes higher-order masking a sound countermeasure.

2.2 Soundness of Higher-Order Masking

The soundness of higher-order masking was formally demonstrated by Chari *et al.* in [6]. They assume a simplified but still realistic leakage model where a bit b is masked using d random bits $x_1, \ldots, x_d$ such that the masked bit is defined as $x_0 = b \oplus x_1 \oplus \cdots \oplus x_d$. The adversary is assumed to be provided with observations of $d+1$ leakage variables L_i, each one corresponding to a share x_i. For every i, the leakage is modelled as $L_i = x_i + N_i$ where the noises N_i's are assumed to have Gaussian distributions $\mathcal{N}(\mu, \sigma^2)$ and to be mutually independent. Under this leakage model, they show that the number of samples q required by the adversary to distinguish the distribution $(L_0, \ldots, L_d | b = 0)$ from the distribution $(L_0, \ldots, L_d | b = 1)$ with a probability at least α satisfies:

$$q \geqslant \sigma^{d+\delta} \tag{2}$$

where $\delta = 4 \log \alpha / \log \sigma$. This result encompasses all the possible side-channel distinguishers and hence formally states the resistance against every kind of side channel attack. Although the model is simplified, it could probably be extended to more common leakage models such as the Hamming weight/distance model. The point is that if an attacker observes noisy side channel information about $d+1$ shares corresponding to a variable masked with d random masks, the number of samples required to retrieve information about the unmasked variable is lower bounded by an exponential function of the masking order whose base is related to the noise standard deviation. This formally demonstrates that higher-order masking is a sound countermeasure especially when combined with noise. Many works also made this observation in practice for particular side channel distinguishers (see for instance [37, 38, 41]).

2.3 Higher-Order Masking Schemes

When dth-order masking is involved in protecting a block cipher implementation, a so-called *dth-order masking scheme* (or simply a *masking scheme* if there is no ambiguity on d) must be designed to enable computation on masked data. In order to be complete and secure, the scheme must satisfy the two following properties:

- *completeness*: at the end of the computation, the sum of the d shares must yield the expected ciphertext (and more generally each masked transformation must result in a set of shares whose sum equal the correct intermediate result),
- *dth-order SCA security*: every tuple of d or less intermediate variables must be independent of any sensitive variable.

If the dth-order security property is satisfied, then no attack of order lower than $d + 1$ is possible and we benefit from the security bound (2).

Most block cipher structures (*e.g.* AES or DES) alternate several rounds composed of a key addition, one or several linear transformation(s), and a non-linear transformation. The main difficulty in designing masking schemes for such block ciphers lies in masking the nonlinear transformations. Many solutions have been proposed to deal with this issue but the design of a dth-order secure scheme for $d > 1$ has quickly been recognized as a difficult problem by the community. As mentioned above, only three methods exist in the literature that have been respectively proposed by Ishai, Sahai and Wagner [18], by Schramm and Paar [38] (secure only for $d \leqslant 2$) and by Rivain, Dottax and Prouff [34] (dedicated to $d = 2$). Among them, only [18] can be applied to secure a non-linear transformation at any order d. This scheme is recalled in the next section.

2.4 The Ishai-Sahai-Wagner Scheme

In [18], Ishai *et al.* propose a higher-order masking scheme (referred to as ISW in this paper) enabling to secure the hardware implementation of any *circuit* at any chosen order d. They describe a way to transform the circuit to protect into a new circuit (dealing with masked values) such that no subset of d of its *wires* reveals information about the unmasked values[3]. For such a purpose, they assume without loss of generality that the circuit to protect is exclusively composed of NOT and AND gates. Securing a NOT for any order d is straightforward since $x = \bigoplus_i x_i$ implies $\mathrm{NOT}(x) = \mathrm{NOT}(x_0) \oplus x_1 \cdots \oplus x_d$. The main difficulty is therefore to secure the AND gates. To answer this issue, Ishai *et al.* suggest the following elegant solution.

Secure logical AND. Let a an b be two bits and let c denote $\mathrm{AND}(a, b) = ab$. Let us assume that a and b have been respectively split into $d+1$ shares $(a_i)_{0 \leqslant i \leqslant d}$ and $(b_i)_{0 \leqslant i \leqslant d}$ such that $\bigoplus_i a_i = a$ and $\bigoplus_i b_i = b$. To securely compute a $(d + 1)$-tuple $(c_i)_{0 \leqslant i \leqslant d}$ s.t. $\bigoplus_i c_i = c$, Ishai *et al.* perform the following steps:

1. For every $0 \leqslant i < j \leqslant d$, pick up a random bit $r_{i,j}$.
2. For every $0 \leqslant i < j \leqslant d$, compute $r_{j,i} = (r_{i,j} \oplus a_i b_j) \oplus a_j b_i$.
3. For every $0 \leqslant i \leqslant d$, compute $c_i = a_i b_i \oplus \bigoplus_{j \neq i} r_{i,j}$.

[3] Considering wires as intermediate variables, this is equivalent to the security property given in Section 2.3.

Remark 1. The use of brackets indicates the order in which the operations are performed, which is mandatory for security of the scheme.

The completeness of the solution follows from:

$$\bigoplus_i c_i = \bigoplus_i \left(a_i b_i \oplus \bigoplus_{j \neq i} r_{i,j}\right) = \bigoplus_i \left(a_i b_i \oplus \bigoplus_{j>i} r_{i,j} \oplus \bigoplus_{j<i} (r_{j,i} \oplus a_i b_j \oplus a_j b_i)\right)$$
$$= \bigoplus_i \left(a_i b_i \oplus \bigoplus_{j<i} (a_i b_j \oplus a_j b_i)\right) = \left(\bigoplus_i a_i\right)\left(\bigoplus_i b_i\right) \ .$$

In [18] it is shown that the AND computation above is secure against any attack of order lower than or equal to $d/2$. In Section 4, we give a tighter security proof: we show that the scheme is actually dth-order secure.

Practical issues. Although the ISW scheme is an important theoretical result, its practical application suffers few issues. Firstly, it induces an important overhead in silicon area for the masked circuit. Indeed, every single AND gate is encoded using $(d+1)^2$ AND gates plus $2d(d+1)$ XOR gates, and it requires the generation of $d(d+1)/2$ random bits at every clock cycle. As an illustration, masking the compact circuit for the AES S-box described in [5] would multiply its size (in terms of number of gates) by 7 for $d = 2$, by 14 for $d = 3$ and by 22 for $d = 4$ (without taking the random bits generation into account). Secondly, masking at the hardware level is sensitive to glitches, which induces first-order flaws although in theory every internal wire carries values that are independent of the sensitive variables [20,21]. Preventing glitches in masked circuits imply the addition of synchronizing elements (*e.g.* registers or latches) which still significantly increases the circuit size (see for instance [32]).

Since software implementations of masking schemes do not suffer area overhead and are not impacted by the presence of glitches at the hardware level, a straightforward approach to deal with the practical issues discussed above could be to implement the ISW scheme in software. Namely, we could represent each non-linear transformation S to protect by a tuple of Boolean functions $(f_i)_i$ usually called *coordinate functions* of S, and evaluate the f_i's with the ISW scheme by processing the AND and XOR operations with CPU instructions. However, this approach is not practical since the timing overhead would clearly be prohibitive. The present paper follows a different approach: we generalize the ISW scheme to secure any finite field multiplication rather than a simple multiplication over $\mathbb{F}_2$ (*i.e.* a logical AND). We apply this idea to design a secure higher-order masking scheme for the AES and we show that its software implementation induces a reasonable overhead.

3 Higher-Order Masking of AES

The AES block cipher iterates a round transformation composed of a key addition, a linear layer and a nonlinear layer which applies the same substitution-box (S-box) to every byte of the internal state. As previously explained, the main difficulty while designing a masking scheme for such a cipher is the masking of the nonlinear transformation, which in that case lies in the masking of the S-box. Our method for masking the AES S-box is presented in the next section, afterward the masking of the whole cipher is described.

In what follows, we shall consider that a random generator is available which on an invocation $\mathsf{rand}(n)$ returns n unbiased random bits.

3.1 Higher-Order Masking of the AES S-box

The AES S-box S is defined as the right-composition of an affine transformation Af over $\mathbb{F}_2^8$ with the power function $x \mapsto x^{254}$ over the field $\mathbb{F}_{2^8} \equiv \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. Since the affine transformation is straightforward to mask, our scheme mainly consists in a method for masking the power function at any order d. Our solution consists in a secure computation of the exponentiation to the power 254 over $\mathbb{F}_{2^8}$. Such an approach has already been described by Blömer *et al.* for $d = 1$ [3]. The core idea is to apply an exponentiation algorithm (*e.g.* the square-and-multiply algorithm) on the first-order masked input while ensuring the mask correction step by step. Compared to Blömer *et al.* 's solution, our exponentiation algorithm is able to operate on dth-order masked inputs and it achieves dth-order SCA security for any value of d. To perform such a secure exponentiation, we define hereafter some methods to securely compute a squaring and a multiplication over $\mathbb{F}_{2^8}$ at the dth order.

Masking the field squaring. Since we operate on a field of characteristic 2, the squaring is a linear operation and we have $x_0^2 \oplus x_1^2 \oplus \cdots \oplus x_d^2 = x^2$. Securely computing a squaring can hence be carried out by squaring every share separately. More generally, for every natural integer j, raising x to the power 2^j can be done securely by raising each x_i to the 2^j separately.

Masking the field multiplication. For the usual field multiplication we use the ISW scheme recalled in Section 2.4. Even if it has been described to securely compute a logical AND (that is a multiplication over $\mathbb{F}_2$), it can actually be transposed to secure a multiplication over any field of characteristic 2: variables over $\mathbb{F}_2$ are replaced by variables over $\mathbb{F}_{2^n}$, binary multiplications (*i.e.* ANDs) are replaced by multiplications over $\mathbb{F}_{2^n}$ and binary additions (*i.e.* XORs) are replaced by addition over $\mathbb{F}_{2^n}$ (that are n-bit XORs). This keep unchanged the completeness of the scheme recalled in Section 2.4. The whole secure multiplication over $\mathbb{F}_{2^n}$ is depicted in the following algorithm.

Algorithm 1 SecMult - dth-order secure multiplication over $\mathbb{F}_{2^n}$

INPUT: shares a_i satisfying $\bigoplus_i a_i = a$, shares b_i satisfying $\bigoplus_i b_i = b$
OUTPUT: shares c_i satisfying $\bigoplus_i c_i = ab$

1. **for** $i = 0$ **to** d **do**
2. **for** $j = i + 1$ **to** d **do**
3. $r_{i,j} \leftarrow \mathsf{rand}(n)$
4. $r_{j,i} \leftarrow (r_{i,j} \oplus a_i b_j) \oplus a_j b_i$
5. **for** $i = 0$ **to** d **do**
6. $c_i \leftarrow a_i b_i$
7. **for** $j = 0$ **to** d, $j \neq i$ **do** $c_i \leftarrow c_i \oplus r_{i,j}$

Masking the power function. Now we have a secure squaring and a secure multiplication over $\mathbb{F}_{2^8}$ it remains to specify an exponentiation algorithm. It is clear from Algorithm 1 that the running time of a secure multiplication is huge compared to that of a secure squaring. A secure squaring indeed requires $d + 1$ squarings while a secure multiplication requires $(d+1)^2$ field multiplications, $2d(d+1)$ XORs and the generation of $d(d+1)/2$ random 8-bit values. Our goal is therefore to design an exponentiation algorithm using the least possible multiplications which are not squares. It can be checked that an exponentiation to the power

254 requires at least 4 such multiplications. The exponentiation algorithm presented hereafter achieves this lower bound and requires few additional squares. It involves three intermediate variables denoted y, z and w (note that x and y may be associated to the same memory address).

Algorithm 2 Exponentiation to the 254

INPUT: x
OUTPUT: $y = x^{254}$

1. $z \leftarrow x^2$	$[z = x^2]$
2. $y \leftarrow zx$	$[y = x^2 x = x^3]$
3. $w \leftarrow y^4$	$[w = (x^3)^4 = x^{12}]$
4. $y \leftarrow yw$	$[y = x^3 x^{12} = x^{15}]$
5. $y \leftarrow y^{16}$	$[y = (x^{15})^{16} = x^{240}]$
6. $y \leftarrow yw$	$[y = x^{240} x^{12} = x^{252}]$
7. $y \leftarrow yz$	$[y = x^{252} x^2 = x^{254}]$

As we will argue in Section 4, , for the dth-order security to hold, it is important that the masks $(a_i)_{i \geqslant 1}$ and $(b_i)_{i \geqslant 1}$ in input of the SecMult algorithm are mutually independent. That is why we shall refresh the masks at some points during the secure exponentiation by calling a procedure RefreshMasks[4]. The whole exponentiation to the power 254 over $\mathbb{F}_{2^8}$ secure against dth-order SCA is depicted in the following algorithm.

Algorithm 3 SecExp254 - dth-order secure exponentiation to the 254 over $\mathbb{F}_{2^8}$

INPUT: shares x_i satisfying $\bigoplus_i x_i = x$
OUTPUT: shares y_i satisfying $\bigoplus_i y_i = x^{254}$

1. **for** $i = 0$ **to** d **do** $z_i \leftarrow x_i^2$	$[\bigoplus_i z_i = x^2]$
2. RefreshMasks$(z_0, z_1, \ldots, z_d)$	
3. $(y_0, y_1, \ldots, y_d) \leftarrow$ SecMult$\big((z_0, z_1, \ldots, z_d), (x_0, x_1, \ldots, x_d)\big)$	$[\bigoplus_i y_i = x^3]$
4. **for** $i = 0$ **to** d **do** $w_i \leftarrow y_i^4$	$[\bigoplus_i w_i = x^{12}]$
5. RefreshMasks$(w_0, w_1, \ldots, w_d)$	
6. $(y_0, y_1, \ldots, y_d) \leftarrow$ SecMult$\big((y_0, y_1, \ldots, y_d), (w_0, w_1, \ldots, w_d)\big)$	$[\bigoplus_i y_i = x^{15}]$
7. **for** $i = 0$ **to** d **do** $y_i \leftarrow y_i^{16}$	$[\bigoplus_i y_i = x^{240}]$
8. $(y_0, y_1, \ldots, y_d) \leftarrow$ SecMult$\big((y_0, y_1, \ldots, y_d), (w_0, w_1, \ldots, w_d)\big)$	$[\bigoplus_i y_i = x^{252}]$
9. $(y_0, y_1, \ldots, y_d) \leftarrow$ SecMult$\big((y_0, y_1, \ldots, y_d), (z_0, z_1, \ldots, z_d)\big)$	$[\bigoplus_i y_i = x^{254}]$

For completeness, we describe the RefreshMasks algorithm hereafter.

Algorithm 4 RefreshMasks

INPUT: shares x_i satisfying $\bigoplus_i x_i = x$
OUTPUT: shares x_i satisfying $\bigoplus_i x_i = x$

```
1. for i = 1 to d do
2.     tmp ← rand(8)
3.     x_0 ← x_0 ⊕ tmp
4.     x_i ← x_i ⊕ tmp
```

Algorithm 3 involves of $8d(d+1) + 4d$ XORs, $4(d+1)^2$ multiplications (over $\mathbb{F}_{2^8}$), $d+1$ squares, $d+1$ raising to the 4 and $d+1$ raising to the 16. It uses $3(d+1) + d(d+1)/2$ bytes

[4] Note that the masks resulting from the SecMult algorithm are independent of the input masks.

Table 1. Complexity of SecExp254.

order	nb. XORs	nb. mult.	nb. $\hat{}2^j$	nb. rand. bytes	memory (bytes)
1	20	16	6	6	7
2	56	36	9	16	12
3	108	64	12	20	18
4	176	100	15	48	25
5	260	144	18	70	33
d	$8d^2+12d$	$4d^2+8d+4$	$3d+3$	$2d^2+4d$	$\frac{1}{2}d^2+\frac{7}{2}d+3$

of memory[5] and it requires the generation of $2d(d+1)+2d$ random bytes (see illustrative values in Table 1). In comparison, the 2nd-order countermeasures previously published [34,38] require at least 512 look-ups and 512 XORs and have a memory consumption of at least 256 bytes (see [33, 35] for a detailed comparison).

Masking the full S-box. The affine transformation is straightforward to mask. After recalling that the additive part of Af equals `0x63`, it can be checked that we have:

$$Af(x_0) \oplus Af(x_1) \oplus \cdots \oplus Af(x_d) = \begin{cases} Af(x) & \text{if } d \text{ is even,} \\ Af(x) \oplus \texttt{0x63} & \text{if } d \text{ is odd.} \end{cases}$$

Masking the affine transformation hence simply consists in applying it to every input share separately and, in case of an even d, in adding `0x63` to one of the share afterward. The full S-box computation secure against dth-order SCA is summarized in the following algorithm.

Algorithm 5 SecSbox

INPUT: shares x_i satisfying $\bigoplus_i x_i = x$
OUTPUT: shares y_i satisfying $\bigoplus_i y_i = \mathrm{S}(x)$

1. $(y_0, \ldots, y_d) \leftarrow \mathsf{SecExp254}(x_0, \ldots, x_d)$
2. **for** $i = 0$ **to** d **do** $y_i \leftarrow Af(y_i)$
3. **if** ($d \bmod 2 = 1$) **then** $y_0 \leftarrow y_0 \oplus \texttt{0x63}$

Implementation aspects. Multiplications over $\mathbb{F}_{2^8}$ are typically implemented in software using log/alog tables (see for instance [11]). Note that for security reasons, such an implementation must avoid conditional branches in order to ensure a constant operation flow. The squaring and raisings to the 4 and 16 may be looked-up. Different time-memory tradeoffs are possible. If not much ROM is available, the squaring can be implemented using logical shifts and XORs (see for instance [11]), and the raising to the 2^j, $j \in \{2, 4\}$, can then be simply processed by j sequential squarings. Otherwise, depending on the amount of ROM available, one can either use one, two or three look-up table(s) to implement the raisings to 2^j, $j \in \{1, 2, 4\}$.

Remark 2. For the implementations presented in Section 5, we chose to implement the squaring by a look-up table, getting the raising to the 4 (resp. 16) by accessing this table sequentially 2 (resp. 4) times.

[5] $3(d+1)$ bytes for the shares y_i's, z_i's and w_i's (Algorithm 3), and $d(d+1)/2$ for the intermediate variables $r_{i,j}$'s (Algorithm 1).

Our scheme may also be implemented in hardware. The sensitive part is the implementation of the SecMult algorithm (see Algorithm 1) which may be subject to glitches and which should incorporate synchronizing elements. In particular, the evaluation of the c_i shares should not start before the evaluation of all the $r_{i,j}$'s has been fully completed. Another approach would be to enhance the software implementation of the scheme with special purpose hardware instructions. For instance, the multiplication, squaring and raisings to powers 4 and 16 over $\mathbb{F}_{2^8}$ could be added to the instructions set of the processor.

3.2 Higher-Order Masking of the Whole Cipher

In the previous section, we have shown how the AES S-box can be masked at any chosen order d. We now detail the dth-order masking scheme for the whole AES block cipher.

The AES block cipher [11] operates on a 4×4 array of bytes called the state and denoted $\mathbf{s} = (s_{l,j})_{1 \leqslant l,j \leqslant 4}$. The state is initialized by the plaintext value and holds the ciphertext value at the end of the encryption. Each round of AES is composed of four stages: AddRoundKey, SubBytes, ShiftRows, and MixColumns (except the last round that omits the MixColumns). AES is composed of either 10, 12 or 14 rounds, depending on the key length (the longer the key, the higher the number of rounds) plus a final AddRoundKey stage. The round keys involved in the different rounds are derived from the secret key through a key expansion process.

In what follows, we describe how to mask an AES computation at the dth order. We will assume that the secret key has been previously masked and that its $d+1$ shares are provided as input to the algorithm (otherwise a straightforward first-order attack would be possible). At the beginning of the computation, the state (holding the plaintext) is split into $d+1$ states $\mathbf{s}_0$, $\mathbf{s}_1$, ..., $\mathbf{s}_d$ satisfying:

$$\mathbf{s} = \mathbf{s}_0 \oplus \mathbf{s}_1 \oplus \cdots \oplus \mathbf{s}_d \ .$$

This is done by generating d random states $\mathbf{s}_i \leftarrow \mathsf{rand}(16 \times 8)$ and by computing $\mathbf{s}_0 \leftarrow \mathbf{s} \oplus \bigoplus_{i \geqslant 1} \mathbf{s}_i$. At the end of the AES computation, the state (holding the ciphertext) is recovered by $\mathbf{s} \leftarrow \bigoplus_i \mathbf{s}_i$.

In the next sections, we describe how to perform the different AES transformations on the state shares in order to guarantee the completeness as well as the dth-order security.

Masking AddRoundKey. The AddRoundKey stage at round r consists in adding (by XOR) the rth round key $\mathbf{k}^r$ to the state. The masked key expansion (see description hereafter) provides $d+1$ shares $(\mathbf{k}_i^r)_i$ for every round key $\mathbf{k}^r$. To securely process the addition of $\mathbf{k}^r$, one simply adds each of its share to one share of the state and the completeness holds from:

$$\mathbf{s} \oplus \mathbf{k}^r = (\mathbf{s}_0 \oplus \mathbf{k}_0^r) \oplus (\mathbf{s}_1 \oplus \mathbf{k}_1^r) \oplus \cdots \oplus (\mathbf{s}_d \oplus \mathbf{k}_d^r) \ .$$

Masking SubBytes. The SubBytes transformation consists in applying the AES S-box S to each byte of the state:

$$\mathsf{SubBytes}(\mathbf{s}) = (\mathrm{S}(s_{l,j}))_{1 \leqslant l,j \leqslant 4} \ .$$

In order to mask this transformation, we apply the secure S-box computation described in Section 3.1 to the $(d+1)$-tuple of byte shares $((\mathbf{s}_0)_{l,j}, (\mathbf{s}_1)_{l,j}, \ldots, (\mathbf{s}_d)_{l,j})$ for every row-coordinate $l \in [1,4]$ and for every column-coordinate $j \in [1,4]$.

Masking ShiftRows and MixColumns. The ShiftRows and MixColumns transformations compose the linear layer of AES. In the ShiftRows transformation, the bytes in the last three rows of the state are cyclically shifted over different numbers of bytes (1 for the second row, 2 for the third row and 3 for the fourth row). The MixColumns transformation operates on the state column-by-column. Each column is treated as a four-term polynomial over $\mathbb{F}_2[x]/(x^8 + x^4+x^3+x+1)$ and is multiplied modulo x^4+1 with a fixed polynomial $a(x) = 3x^3+x^2+x+2$. Since they are both linear with respect to the XOR operation, masking these transformations is straightforward. One just apply them to every state share separately and the completeness holds from:

$$\mathsf{ShiftRows}(\mathbf{s}) = \bigoplus_{i=0}^{d} \mathsf{ShiftRows}(\mathbf{s}_i) \ ,$$

and:

$$\mathsf{MixColumns}(\mathbf{s}) = \bigoplus_{i=0}^{d} \mathsf{MixColumns}(\mathbf{s}_i).$$

Masking the key expansion. The AES key expansion generates a $4 \times 4(\mathrm{Nr} + 1)$ array of bytes $\mathbf{w}$, called the key schedule, where Nr is the number of rounds (which depends on the key-length). Let $\mathbf{w}_{*,j}$ denotes the jth column of $\mathbf{w}$. Each group of 4 columns $(\mathbf{w}_{*,4r-3}, \mathbf{w}_{*,4r-2}, \mathbf{w}_{*,4r-1}, \mathbf{w}_{*,4r})$ forms a round key $\mathbf{k}^r$ that is XORed to the state during the rth AddRoundKey stage. The first Nk columns of the key schedule are filled with the key bytes (where the key byte-length is 4Nk) and the next ones are derived according to the process described hereafter.

Let SubWord be the transformation that takes a four-byte input column and applies the AES S-box to each byte. Let RotWord be the transformation that takes a 4-byte column as input and performs a cyclic shift of one byte from bottom to top. Finally, let $\mathtt{Rcon}_j$ denotes the constant 4-bytes column $(\{02\}^{j-1}, 0, 0, 0)^T$, where $\{02\}^{j-1}$ is the $(j-1)$th power of x in the field $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. The jth column of the key schedule $\mathbf{w}_{*,j}$ is defined as:

$$\mathbf{w}_{*,j} = \mathbf{w}_{*,j-\mathrm{Nk}} \oplus \mathbf{t}$$

with:

$$\mathbf{t} = \begin{cases} \mathsf{RotWord}(\mathsf{SubWord}(\mathbf{w}_{*,j-1})) \oplus \mathtt{Rcon}_{j/\mathrm{Nk}} & \text{if } (j \bmod \mathrm{Nk} = 0), \\ \mathsf{SubWord}(\mathbf{w}_{*,j-1}) & \text{if } (\mathrm{Nk} = 8) \text{ and } (j \bmod \mathrm{Nk} = 4), \\ \mathbf{w}_{*,j-1} & \text{otherwise.} \end{cases}$$

In order to securely process the key expansion at the dth-order, the key schedule $\mathbf{w}$ is split into $d+1$ schedules $\mathbf{w}_0$, $\mathbf{w}_1$, ..., $\mathbf{w}_d$. The first columns of each schedule shares are filled with the key shares at the beginning of the ciphering. Each time a new schedule column $\mathbf{w}_{*,j}$ must be computed, its $d+1$ shares $(\mathbf{w}_0)_{*,j}$, $(\mathbf{w}_1)_{*,j}$, ..., $(\mathbf{w}_d)_{*,j}$ are computed as:

$$(\mathbf{w}_i)_{*,j} = (\mathbf{w}_i)_{*,j-\mathrm{Nk}} \oplus \mathbf{t}_i$$

where the $\mathbf{t}_i$'s denote the 4-bytes shares of $\mathbf{t}$ that are securely computed from the 4-bytes shares of $\mathbf{w}_{*,j-1}$. Such a secure computation can be easily deduced from the methods described above. The SubWord transformation is processed by applying the secure S-box computation described in Section 3.1 to the byte shares $(\mathbf{w}_0)_{l,j}$, $(\mathbf{w}_1)_{l,j}$, ..., $(\mathbf{w}_d)_{l,j}$ for each row-coordinate

$l \in [1,4]$. Since RotWord is linear with respect to the XOR, it is applied (when involved) to every share separately. Finally, when $\texttt{Rcon}_{j/\mathrm{Nk}}$ must be added to $\mathbf{t}$, it is added to one of its share (*e.g.* $\mathbf{t}_0$).

The whole dth-order secure key expansion process is summarized in the following algorithm.

Algorithm 6 dth-order secure AES key expansion
INPUT: key shares $\mathbf{k}_i$ satisfying $\bigoplus_i \mathbf{k}_i = \mathbf{k}$
OUTPUT: shares $\mathbf{w}_i$ satisfying $\bigoplus_i \mathbf{w}_i = \mathbf{w}$

1. **for** $j = 1$ **to** Nk **do**
2. **for** $i = 0$ **to** d **do** $(\mathbf{w}_i)_{*,j} \leftarrow (\mathbf{k}_i)_{*,j}$
3. **for** $j = \mathrm{Nk}+1$ **to** $4(\mathrm{Nr}+1)$ **do**
4. **for** $i = 0$ **to** d **do** $\mathbf{t}_i \leftarrow (\mathbf{w}_i)_{*,j-1}$
5. **if** $\big((j \bmod \mathrm{Nk} = 0)$ or $(\mathrm{Nk} = 8)$ and $(j \bmod \mathrm{Nk} = 4)\big)$ **then**
6. **for** $l = 1$ **to** 4 **do** $\big((\mathbf{t}_0)_l, (\mathbf{t}_1)_l, \ldots, (\mathbf{t}_d)_l\big) \leftarrow \mathsf{SecSbox}\big((\mathbf{t}_0)_l, (\mathbf{t}_1)_l, \ldots, (\mathbf{t}_d)_l\big)$
7. **if** $(j \bmod \mathrm{Nk} = 0)$ **then**
8. **for** $i = 0$ **to** d **do** $\mathbf{t}_i \leftarrow \mathsf{RotWord}(\mathbf{t}_i)$
9. $\mathbf{t}_0 \leftarrow \mathbf{t}_0 \oplus \texttt{Rcon}_{j/\mathrm{Nk}}$
10. **for** $i = 0$ **to** d **do** $(\mathbf{w}_i)_{*,j-1} \leftarrow (\mathbf{w}_i)_{*,j-\mathrm{Nk}} \oplus \mathbf{t}_i$

Remark 3. Note that the key expansion can be executed on-the-fly during the AES computation in order to avoid the storage of all the round keys.

Masking the whole AES: algorithmic description. Algorithm 7 summarizes the whole AES computation secure against dth-order SCA.

Algorithm 7 dth-order secure AES computation
INPUT: plaintext $\mathbf{p}$, key shares $\mathbf{k}_i$ satisfying $\bigoplus_i \mathbf{k}_i = \mathbf{k}$
OUTPUT: ciphertext $\mathbf{c}$

1. $\mathbf{s}_0 \leftarrow \mathbf{p}$

*** State masking ***
2. **for** $i = 0$ **to** d **do**
3. $\mathbf{s}_i \leftarrow \mathsf{rand}(16 \times 8)$
4. $\mathbf{s}_0 \leftarrow \mathbf{s}_0 \oplus \mathbf{s}_i$

*** All but last rounds ***
5. **for** $r = 1$ **to** $\mathrm{Nr} - 1$ **do**
6. **for** $i = 0$ **to** d **do** $\mathbf{s}_i \leftarrow \mathbf{s}_i \oplus \mathbf{k}_i^r$
7. **for** $l = 1$ **to** 4, $j = 1$ **to** 4 **do**
8. $\big((\mathbf{s}_0)_{l,j}, (\mathbf{s}_1)_{l,j}, \ldots, (\mathbf{s}_d)_{l,j}\big) \leftarrow \mathsf{SecSbox}\big((\mathbf{s}_0)_{l,j}, (\mathbf{s}_1)_{l,j}, \ldots, (\mathbf{s}_d)_{l,j}\big)$
9. **for** $i = 0$ **to** d **do** $\mathbf{s}_i \leftarrow \mathsf{MixColumns}\big(\mathsf{ShiftRows}(\mathbf{s}_i)\big)$

*** Last round ***
10. **for** $i = 0$ **to** d **do** $\mathbf{s}_i \leftarrow \mathbf{s}_i \oplus \mathbf{k}_i^{\mathrm{Nr}}$
11. **for** $l = 1$ **to** 4, $j = 1$ **to** 4 **do**
12. $\big((\mathbf{s}_0)_{l,j}, (\mathbf{s}_1)_{l,j}, \ldots, (\mathbf{s}_d)_{l,j}\big) \leftarrow \mathsf{SecSbox}\big((\mathbf{s}_0)_{l,j}, (\mathbf{s}_1)_{l,j}, \ldots, (\mathbf{s}_d)_{l,j}\big)$
13. **for** $i = 0$ **to** d **do** $\mathbf{s}_i \leftarrow \mathsf{ShiftRows}(\mathbf{s}_i)$
14. **for** $i = 0$ **to** d **do** $\mathbf{s}_i \leftarrow \mathbf{s}_i \oplus \mathbf{k}_i^{\mathrm{Nr}+1}$

*** State unmasking ***
15. $\mathbf{c} \leftarrow \mathbf{s}_0$
16. **for** $i = 1$ **to** d **do** $\mathbf{c} \leftarrow \mathbf{c} \oplus \mathbf{s}_i$

---

4 Security Analysis

In this section, we give a formal security proof for our scheme. After describing the security model, we pay particular attention to the secure field multiplication algorithm SecMult (*i.e.* the generalized ISW scheme) which is the sensitive part of our scheme. We improve the security proof given in [18] for the ISW scheme and we show that it achieves dth-order security rather than $(d/2)$th-order security. Afterward, we prove the security of the whole AES computation (Algorithm 7).

4.1 Security Model

We consider a *randomized encryption algorithm* $\mathcal{E}$ taking a plaintext p and a (randomly shared) secret key k as inputs[6] and performing a deterministic encryption of p under the secret key k while randomizing its internal computations by means of an external random number generator (RNG). The RNG outputs are assumed to be perfectly random (uniformly distributed, mutually independent and independent of the plaintext and of the secret key). Any variable that can be expressed as a deterministic function of the plaintext and the secret key, which is not constant with respect to the secret key, is called a *sensitive variable* with the exception of the ciphertext $\mathcal{E}_k(p)$ or any deterministic function of it. Note that every intermediate variable computed during an execution of $\mathcal{E}$ (except the plaintext and the ciphertext) can be expressed as a deterministic function of a sensitive variable and of the RNG outputs.

We shall consider the plaintext, the secret key and the intermediate variables of $\mathcal{E}$ as random variables. The distributions of the intermediate variables are induced by the algorithm inputs (p and k) distributions and by the uniformity of the RNG outputs. The joint distribution of all the intermediate variables of $\mathcal{E}$ thus depends on (p, k). On the other hand, some subsets of intermediate variables may be jointly independent of (p, k). This leads us to the following formal definition of *dth-order SCA security*.

Definition 1. *A randomized encryption algorithm is said to achieve* dth-order SCA security *if every d-tuple of its intermediate variables is independent of any sensitive variable.*

Equivalently, an encryption algorithm achieves dth-order SCA security if any d-tuple of its intermediate variables, except the plaintext and the ciphertext (or any function of one of them), is independent of the algorithm inputs (p, k).

Before proving the security of our scheme, we need to introduce a few additional notions. A $(d+1)$*-family of shares* is a family of $d+1$ intermediate variables $(x_i)_{0 \leqslant i \leqslant d}$ such that every d-tuple of x_i's is uniformly distributed and independent of any sensitive variable and $\bigoplus_{0 \leqslant i \leqslant d} x_i$ is a sensitive variable. Two $(d+1)$-families of shares $(x_i)_i$ and $(y_i)_i$ are said to be

[6] The secret key k is assumed to be split into $d+1$ shares k_0, k_1, ..., k_d such that $\bigoplus_i k_i = k$ and every d-tuple of k_i's is uniformly distributed and independent of k.

d-independent one of each other if every $(2d)$-tuple composed of d elements from $(x_i)_i$ and of d elements from $(y_i)_i$ is uniformly distributed and independent of any sensitive variable. Two $(d+1)$-families of shares are said to be *d-dependent* one on each other if they are not d-independent. A randomized encryption algorithm aiming at dth-order SCA security typically operates on $(d+1)$-families of shares. Such an algorithm can hence be split into several *randomized elementary transformations* defined as algorithms taking one or two d-independent $(d+1)$-families of shares as input and returning a $(d+1)$-family of shares.

To prove the dth-order SCA security of our scheme, we will first show that it can be split into several randomized elementary transformations each achieving dth-order SCA security. Afterward, the security of the whole algorithm will be demonstrated.

As in [18], our proofs shall apply similar techniques as zero-knowledge proofs [16]. We shall show that the distribution of every d-tuple of intermediate variables $(v_1, v_2, \ldots, v_d)$ of our randomized AES algorithm can be *perfectly simulated* without knowing p and k. Namely, we show that it is possible to construct a d-tuple of random variables which is identically distributed as $(v_1, v_2, \ldots, v_d)$, independently of any statement about p and k. In some cases, the simulated distribution shall involve some intermediate variables $(w_i)_i$ (different from the v_i's). We shall then say that $(v_1, v_2, \ldots, v_d)$ can be *perfectly simulated from the w_i's*. It follows that if $(v_1, v_2, \ldots, v_d)$ can be perfectly simulated from some intermediate variables w_i's which are jointly independent of p and k, then $(v_1, v_2, \ldots, v_d)$ is also independent of p and k. We are now able to introduce the first lemma of our security proof.

Lemma 1. *A randomized elementary transformation $\mathcal{T}$ achieves dth-order SCA security if and only if the distribution of every d-tuple of its intermediate variables can be perfectly simulated from at most d shares of each of its input $(d+1)$-families.*

Proof. Let us assume that every d-tuple $\mathbf{v} = (v_1, v_2, \ldots, v_d)$ of intermediate variables of $\mathcal{T}$ can be perfectly simulated from at most d shares of each of its input $(d+1)$-families of shares. By definition of a $(d+1)$-family of shares, this amounts to assume that every such $\mathbf{v}$ can be simulated from (at most) $2d$ uniform random variables that are independent of any sensitive variable. It follows that every d-tuple of intermediate variables $\mathbf{v}$ is independent of any sensitive variable, which implies that $\mathcal{T}$ is dth-order SCA secure. Let us now assume that there exits a d-tuple $\mathbf{v}$ of intermediate variables which requires all the $d+1$ shares of one of its input $(d+1)$-families – let say $(x_i)_i$ – to be perfectly simulated. Then, denoting $\bigoplus_i x_i = x$ where x is a sensitive variable, we get that $\mathbf{v}$ depends on x (otherwise d shares x_i would suffice to the simulation) which contradicts the dth-order SCA security of $\mathcal{T}$. □

Lemma 1 shows that proving the security of a randomized elementary transformation $\mathcal{T}$ can be done by exhibiting a method for perfectly simulating the distribution of any d-tuple of intermediate variables of $\mathcal{T}$ from the values of at most d shares of each input $(d+1)$-family of $\mathcal{T}$. We follow this approach in the next section to prove the security of the secure field multiplication algorithm SecMult (Algorithm 1).

4.2 Improved Security Proof for the ISW Scheme

The theorem hereafter states that the generalized ISW scheme (Algorithm 1) achieves dth-order SCA security.

Theorem 1. *Let $(a_i)_{0\leqslant i\leqslant d}$ and $(b_i)_{0\leqslant i\leqslant d}$ be two d-independent $(d+1)$-families of shares in input of Algorithm 1. Then, the distribution of every tuple of d or less intermediate variables in Algorithm 1 is independent of the distribution of values taken by $a = \bigoplus_{0\leqslant i\leqslant d} a_i$ and $b = \bigoplus_{0\leqslant i\leqslant d} b_i$.*

The proof given hereafter follows the outlines of that given by Ishai *et al.* in their paper but it is tighter: we prove that the scheme achieves dth-order SCA security rather than $(d/2)$th-order SCA security as proved in [18]. The core idea of our improvement is to simulate the distribution of any d-tuple of intermediate variables of Algorithm 1 from d shares in $(a_i)_i$ and d shares in $(b_i)_i$ instead of simulating any $(d/2)$-tuple of intermediate variables from d pairs of shares in $(a_i, b_i)_i$.

Proof. Our proof consists in constructing two sets I and J of indices in $[0; d]$ with cardinalities lower than or equal to d and such that the distribution of any d-tuple $(v_1, v_2, \ldots, v_d)$ of intermediate variables of Algorithm 1 can be perfectly simulated from $a_{|I} := (a_i)_{i\in I}$ and $b_{|J} = (b_j)_{j\in J}$. This will prove the theorem statement since, by definition, $a_{|I}$ and $b_{|J}$ are jointly independent of (a, b) as long as the cardinalities of I and J are strictly smaller than d. We describe the constructions of I and J hereafter.

1. Initially, I and J are empty and all the v_h's are unassigned.
2. For every intermediate variable v_h of the form a_i, b_i, a_ib_i, $r_{i,j}$ (for any $i \neq j$) or a sum of values of the above form (including c_i as a special case) add i to I and J. This covers all the intermediate variables of Algorithm 1 except those appearing in the computation of $r_{j,i}$ (Step 4) which are of the form a_ib_j or $r_{i,j} \oplus a_ib_j$. For those intermediate variables add i to I and j to J.
3. Now that the sets I and J have been determined – and note that since there are at most d intermediate variables v_h, the cardinalities of I and J can be at most d – we show how to complete a perfect simulation of the d-tuple $(v_0, v_1, \ldots, v_d)$ using only the values of $a_{|I}$ and $b_{|J}$. First, we assign values to every $r_{i,j}$ entering in the computation of any v_h as follows:
   - If $i \notin I$ (regardless of j), then $r_{i,j}$ does not enter into the computation for any v_h. Thus, its value can be left unassigned.
   - If $i \in I$, but $j \notin I$, then $r_{i,j}$ is assigned a random independent value. Indeed, if $i < j$ this is what would have happened in Algorithm 1. If $i > j$, however, we are making use of the fact that $r_{j,i}$ will never be used in the computation of any v_h (otherwise we would have $j \in I$ by construction). Hence we can treat $r_{i,j}$ as a uniformly random and independent value.
   - If $\{i, j\} \subseteq I$ and $\{i, j\} \subseteq J$, then we have access to a_i, a_j, b_i and b_j and we thus compute $r_{i,j}$ and $r_{j,i}$ exactly as they would have been computed in Algorithm 1; i.e., one of them (say $r_{i,j}$) is assigned a random value and the other $r_{j,i}$ is assigned $r_{i,j} \oplus a_ib_j \oplus a_jb_i$.
   - If $\{i, j\} \subseteq I$ and $\{i, j\} \nsubseteq J$, then at least $r_{i,j}$ or $r_{j,i}$ (or both) does not enter into the computation for any v_h (otherwise we would have $\{i, j\} \subseteq J$ by construction). Following the same reasoning as previously (case $i \in I$, $j \notin I$), we can then assign a random independent value to the one (if any) that enters in the computation of the v_h's.
4. For every intermediate variable v_h of the form a_i, b_i, a_ib_i, $r_{i,j}$ (for any $i \neq j$), or a sum of values of the above form (including c_i as a special case), we know that $i \in I$ and $i \in J$,

and all the needed values of $r_{i,j}$ have already been assigned in a perfect simulation. Thus, v_h can be computed in a perfect simulation.

5. The only types of intermediate variables remaining are $v_h = a_i b_j$ or $v_h = r_{i,j} \oplus a_i b_j$. By construction, we have $i \in I$ and $j \in J$ which allows us to compute $a_i b_j$, and since all the $r_{i,j}$ (entering into the computation of the v_h's) has been assigned, the value of v_h can be simulated perfectly.

□

4.3 Security Proof of Our Scheme

The following theorem states the security of our whole randomized AES (Algorithm 7).

Theorem 2. *The randomized AES computation depicted in Algorithm 7 achieves dth-order SCA security.*

In order to demonstrate the theorem statement, we will use the following lemma.

Lemma 2. *Let $\mathcal{T}$ be a randomized elementary transformation. If $\mathcal{T}$ achieves dth-order SCA security then the distribution of every intermediate variable of $\mathcal{T}$ can be perfectly simulated from at most one share of every input $(d+1)$-families of $\mathcal{T}$.*

Proof. Suppose that the simulation of the distribution of an intermediate variable v from $\mathcal{T}$ requires at least two shares x_{i_1} and x_{i_2} from the same family $(x_i)_i$. The d-tuple composed of v and of the $d-2$ shares $(x_i)_{i \neq i_1, i_2}$ requires the whole $(d+1)$-family of shares $(x_i)_i$ to be perfectly simulated which by Lemma 1 is in contradiction with the dth-order security of $\mathcal{T}$. □

Proof (Theorem 2). An execution of our randomized AES algorithm can be expressed as a sequence of executions of the following randomized elementary transformations[7]:

- the secure key addition (Steps 6, 10 and 14 of Algorithm 7),
- the secure affine transformation (Steps 1 and 2 of Algorithm 5),
- the secure square (Step 1 of Algorithm 3), the secure raising to the 4 (Step 4 of Algorithm 3) and the secure raising to the 16 (Step 7 of Algorithm 3),
- the RefreshMasks procedure (Algorithm 4),
- the SecMult algorithm (Algorithm 1),
- the secure ShiftRows and the secure MixColumns transformations (Steps 9 and 13 of Algorithm 7).

[7] For simplicity we omit the randomized elementary transformations used in the secure key expansion (Algorithm 6). Note that they could be listed without affecting the rest of the proof.

All these transformations take as input either a single $(d+1)$-family of shares (all transformations but the secure key addition and SecMult) or two d-independent $(d+1)$-families of shares (secure key addition and SecMult). Moreover they all achieve dth-order SCA security (it has been proven for SecMult in the previous section and it is straightforward for the remaining randomized elementary transformations since they operate on each input share independently). Let us consider a d-tuple $(v_1, v_2, \ldots, v_d)$ of intermediate variables each from a randomized elementary transformation $\mathcal{T}_i$. By Lemma 2, the distribution of every v_i can be perfectly simulated given the value of at most one share of every $(d+1)$-families in input of $\mathcal{T}_i$. Since by definition the $(d+1)$-families in input of the same $\mathcal{T}_i$ are independent, the set of shares which are necessary to simulate $(v_1, v_2, \ldots, v_d)$ does not contain more than d shares from the same $(d+1)$-family or from d-dependent $(d+1)$-families. It follows that the distribution of $(v_1, v_2, \ldots, v_d)$ can be perfectly simulated from uniform random values that are jointly independent of any sensitive variable. In other words, $(v_1, v_2, \ldots, v_d)$ is independent of any sensitive variable. □

5 Implementation Results

To compare the efficiency of our proposal with that of other methods proposed in the literature, we applied them to protect an implementation of the AES-128 algorithm in encryption mode. We have implemented our new countermeasure for $d \in \{1, 2, 3\}$, namely to counteract either first-order SCA ($d = 1$) or second-order SCA ($d = 2$) or third-order SCA ($d = 3$). Among the numerous methods proposed in the literature to thwart first-order SCA we chose to implement only that having the best timing performance (the *table re-computation method* [23]) and that offering the best memory performance (the *tower field method* [28]). In the second-order case, we implemented the only two existing methods: the one proposed in [38][8] and the one proposed [34]. Eventually, since no countermeasure against 3rd-order SCA was existing before that introduced in this paper, it is the single one in its category.

We wrote the codes in assembly language for an 8051-based 8-bit architecture. The implementations only differ in their approaches to protect the S-box computations. The linear steps of the AES have been implemented in the same way, by following the outlines of the method presented in Sect. 3.2 (and also used in [38] and [34]). In Table 2, we list the timing/memory performances of the different implementations.

As expected, in the first-order case the countermeasures introduced in [23] and [28,29] are much more efficient than ours. This is a consequence of the generic character of our method which is not optimized for one choice of d but aims to work for any d. For instance, the representation of the AES S-box used in [28, 29] involves less field multiplications than our representation. Moreover, those field multiplications can be defined in the subfield $\mathbb{F}_{16}$ of $\mathbb{F}_{256}$, where the field operations can be entirely looked-up thanks to a table of 256 bytes in code memory.

In the second-order case, our proposal becomes much more efficient than the existing solutions. It is 2.2 times faster than the countermeasure proposed in [38] with a RAM memory requirement divided by around 10. It is also 2.5 times faster than the countermeasure in [34] and requires 5.3 times less RAM. Memory allocation differences are merely due to the fact

[8] Initially, the method of [38] was devoted to thwart dth-order SCA for any chosen order d but it has been shown insecure for $d \geqslant 3$ [8].

Table 2. Comparison of secure AES implementations

Method	Reference	cycles	RAM (bytes)	ROM (bytes)
Unprotected Implementation				
No Masking	Na.	3×10^3	32	1150
First Order Masking				
Re-computation	[23]	10×10^3	$256 + 35$	1553
Tower Field in $\mathbb{F}_4$	[28, 29]	77×10^3	42	3195
Our scheme for $d = 1$	This paper	129×10^3	73	3153
Second Order Masking				
Double Re-computations	[38]	594×10^3	$512 + 90$	2336
Single Re-computation	[34]	672×10^3	$256 + 86$	2215
Our scheme for $d = 2$	This paper	271×10^3	79	3845
Third Order Masking				
Our scheme for $d = 3$	This paper	470×10^3	103	4648

that the methods [38] and [34] generalize the table re-computation method and thus require the storage of one (for [34]) or two (for [38]) randomized representation(s) of the AES S-box. The differences in timing performances come from the fact that the methods in [38] and [35] process one loop over all the 256 elements of the S-box look-up table (each loop iteration processing itself a few elementary operations), which is more costly than the 36 field multiplications and 56 bitwise additions involved in our method (see Table 1).

Remark 4. In [34], an improvement of the method implemented for Table 2 is proposed that enables to decrease the number of iterations required for the secure S-box computation when implemented on a 16-bit or 32-bit architecture. In such a context ($d = 2$, 16-bit or 32-bit architecture), the method would still requires much more RAM allocation than ours but it could be slightly more efficient in timing.

Eventually, in the third-order case our method has acceptable timing/memory performances. For comparison, it stays faster than the second-order countermeasures proposed in [38] and [34] and it still requires much less RAM memory. For chips running at 5MHz and 31MHz, an AES encryption of one block requiring 470×10^3 cycles, takes 94ms and 15ms respectively. For some use cases where the size of the message to encrypt/decrypt is not too long such a timing performance is acceptable (*e.g.* challenge-response protocols, Message Authentication Codes for one-block messages as in banking transactions).

6 Conclusion

In this paper, we have presented the first masking scheme dedicated to AES which is provably secure at any chosen order and which can be implemented in software at the cost of a reasonable overhead. We gave a formal security proof of our scheme including an improved security proof for the scheme published by Ishai *et al.* at Crypto 2003. We also provided implementation results showing the practical interest of our scheme as well as its efficiency compared to the existing second-order masking schemes.

References

1. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In **c**C. Ko**c**c, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
2. G. Blakely. Safeguarding cryptographic keys. In *National Comp. Conf.*, volume 48, pages 313–317, New York, June 1979. AFIPS Press.
3. J. Blömer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
4. M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
5. D. Canright. A Very Compact S-Box for AES. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.
6. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
7. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In B. Kaliski Jr., **c**C. Ko**c**c, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–29. Springer, 2002.
8. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.
9. R. Cramer, I. Damgård, and Y. Ishai. Share Conversion, Pseudorandom Secret-Sharing and Applications to Secure Computation. In J. Kilian, editor, *Theory of Cryptography Conference – TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 342–362. Springer, 2005.
10. G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly Secure Password Protocols in the Bounded Retrieval Model. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 225–244. Springer, 2006.
11. J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer, 2002.
12. I. Damgård and M. Keller. Secure Multiparty AES (full paper). Cryptology ePrint Archive, Report 20079/614, 2009. `http://eprint.iacr.org/`.
13. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.
14. FIPS PUB 197. *Advanced Encryption Standard*. National Institute of Standards and Technology, Nov. 2001.
15. FIPS PUB 46-3. *Data Encryption Standard (DES)*. National Institute of Standards and Technology, Oct. 1999.
16. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
17. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In **c**C. Ko**c**c and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
18. Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
19. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
20. S. Mangard, T. Popp, and B. M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
21. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
22. U. Maurer. A provably-secure strongly-randomized cipher. In I. Damgård, editor, *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 361–388. Springer, 1990.
23. T. Messerges. Securing the AES Finalists against Power Analysis Attacks. In B. Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
24. T. Messerges. Using Second-order Power Analysis to Attack DPA Resistant Software. In **c**C. Ko**c**c and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

25. S. Micali and L. Reyzin. Physically Observable Cryptography (Extended Abstract). In M. Naor, editor, *Theory of Cryptography Conference – TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
26. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches. In P. J. Lee and J. H. Cheon, editors, *Information Security and Cryptology – ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2008.
27. E. Oswald, S. Mangard, C. Herbst, and S. Tillich. Practical Second-order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
28. E. Oswald, S. Mangard, and N. Pramstaller. Secure and Efficient Masking of AES – A Mission Impossible ? Cryptology ePrint Archive, Report 2004/134, 2004.
29. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In H. Handschuh and H. Gilbert, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.
30. C. Petit, F.-X. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In M. Abe and V. D. Gligor, editors, *Symposium on Information, Computer and Communications Security – ASIACCS 2008*, pages 56–65. ACM, 2008.
31. K. Pietrzak. A Leakage-Resilient Mode of Operation. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.
32. T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2007.
33. M. Rivain. *On the Physical Security of Cryptographic Implementations*. PhD thesis, University of Luxembourg, September 2009.
34. M. Rivain, E. Dottax, and E. Prouff. Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In K. Nyberg, editor, *Fast Software Encryption – FSE 2008*, Lecture Notes in Computer Science, pages 127–143. Springer, 2008.
35. M. Rivain, E. Dottax, and E. Prouff. Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. Cryptology ePrint Archive, Report 2008/021, 2008. `http://eprint.iacr.org/`.
36. M. Rivain and E. Prouff. Provably Secure Higher-Order Masking of AES. Cryptology ePrint Archive, 2010. `http://eprint.iacr.org/`.
37. M. Rivain, E. Prouff, and J. Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.
38. K. Schramm and C. Paar. Higher Order Masking of the AES. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.
39. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
40. F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. Cryptology ePrint Archive, Report 2009/341, 2009. `http://eprint.iacr.org/`.
41. F.-X. Standaert, N. Veyrat-Charvillon, E. Oswald, B. Gierlichs, M. Medwed, M. Kasper, and S. Mangard. The World is Not Enough: Another Look on Second-Order DPA. Cryptology ePrint Archive, Report 2010/180, 2010. `http://eprint.iacr.org/`.
42. S. Tillich and C. Herbst. Attacking State-of-the-Art Software Countermeasures-A Case Study for AES. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2008.

Appendix B

Higher-Order Masking Schemes for S-boxes

Hereafter is appended the full version of our paper [CGP+12], joint work with Claude Carlet, Louis Goubin, Emmanuel Prouff and Michael Quisquater, published at **FSE 2012**.

Higher-Order Masking Schemes for S-Boxes

Claude Carlet[1], Louis Goubin[2], Emmanuel Prouff[3], Michael Quisquater[2], and Matthieu Rivain[4]

[1] LAGA, Université de Paris 8
claude.carlet@univ-paris8.fr
[2] Université de Versailles St-Quentin-en-Yvelines
louis.goubin@prism.uvsq.fr
michael.quisquater@prism.uvsq.fr
[3] Oberthur Technologies
e.prouff@gmail.com
[4] CryptoExperts
matthieu.rivain@cryptoexperts.com

Abstract. Masking is a common countermeasure against side-channel attacks. The principle is to randomly split every sensitive intermediate variable occurring in the computation into $d+1$ shares, where d is called the *masking order* and plays the role of a security parameter. The main issue while applying masking to protect a block cipher implementation is to design an efficient scheme for the s-box computations. Actually, masking schemes with arbitrary order only exist for Boolean circuits and for the AES s-box. Although any s-box can be represented as a Boolean circuit, applying such a strategy leads to inefficient implementation in software. The design of an efficient and generic higher-order masking scheme was hence until now an open problem. In this paper, we introduce the first masking schemes which can be applied in software to efficiently protect any s-box at any order. We first describe a general masking method and we introduce a new criterion for an s-box that relates to the best efficiency achievable with this method. Then we propose concrete schemes that aim to approach the criterion. Specifically, we give optimal methods for the set of *power functions*, and we give efficient heuristics for the general case. As an illustration we apply the new schemes to the DES and PRESENT s-boxes and we provide implementation results.

1 Introduction

Side-channel analysis is a class of cryptanalytic attacks that exploit the physical environment of a cryptosystem to recover some *leakage* about its secrets. It is often more efficient than a cryptanalysis mounted in the so-called *black-box model* where no leakage occurs. In particular, *continuous side-channel attacks* in which the adversary gets information at each invocation of the cryptosystem are especially threatening. Common attacks as those exploiting the running-time, the power consumption or the electromagnetic radiations of a cryptographic computation fall into this class.

Many implementations of block ciphers have been practically broken by continuous side-channel analysis — see for instance [6, 18, 20, 22] — and securing them has been a longstanding issue for the embedded systems industry. A sound approach is to use *secret sharing* [3, 30], often called *masking* in the context of side-channel attacks. This approach consists in splitting each sensitive variable of the implementation (*i.e.* variables depending on the secret key) into $d+1$ shares, where d is called the *masking order*. It has been shown that the complexity of mounting a successful side-channel attack against a masked implementation increases exponentially with the masking order [7]. Starting from this observation, the design of efficient masking schemes for different ciphers has become a foreground issue.

The DES cipher has been the focus of first designs, with the notable work of Goubin and Patarin in [13]. Further schemes have been subsequently published, in particular for the AES cipher, applying masking in hardware or software with different area-time-memory trade-offs [2, 4, 21, 23, 26, 29]. All these schemes deal with *first-order masking*, namely the intermediate variables are split in two shares (a mask and a masked variable). As a result, they only thwart *first order* side-channel attacks in which the adversary exploits the leakage of a single intermediate computation. During the last years, several works have demonstrated that this defense strategy was not sufficient for long term security purpose and that *higher-order attacks* could be successfully performed against cryptographic implementations (see *e.g.* [22]). This has raised the need for secure and efficient higher-order masking schemes.

Higher-Order Masking. The principle of higher-order masking is to split every sensitive variable x occurring during the computation into $d+1$ shares x_0, $\ldots$, x_d in such a way that the following relation is satisfied for a group operation $\perp$:

$$x_0 \perp x_1 \perp \cdots \perp x_d = x \ . \tag{1}$$

In the rest of the paper, we shall consider that $\perp$ is the addition over some field of characteristic 2. Usually, the d shares $x_1, \ldots, x_d$ (called *the masks*) are randomly picked up and the last one x_0 (called *the masked variable*) is processed such that it satisfies (1). When d random masks are involved per sensitive variable the masking is said to be *of order* d. The tuple $(x_i)_i$ is further called a *dth-order encoding of x*.

When higher-order masking is involved to protect a block cipher implementation, a so-called *masking scheme* must be designed to enable the computation on masked data. Such a scheme must ensure that the final shares correspond to the expected ciphertext on the one hand, and it must ensure the dth-order security property for the chosen order d on the other hand. The latter property states that every tuple of d or less intermediate variables is independent of any sensitive variable. When satisfied, it guarantees that no attack of order lower than or equal to d is possible.

Most block cipher structures (*e.g.* AES or DES) are iterative, meaning that they apply several times a same transformation, called *round*, to an internal state

initially filled with the plaintext. The round itself is composed of a key addition, one or several linear transformation(s) and one or several non-linear s-box(es). Key addition and linear transformations are easily handled as linearity enables to process each share independently. The main difficulty in designing masking schemes for block ciphers hence lies in masking the s-box(es).

Masking and S-Boxes. Whereas many solutions have been proposed to deal with the case of first-order masking (see *e.g.* [2, 4, 21, 25]), only a few solutions exist for the higher-order case. A scheme has been proposed by Schramm and Paar in [29] which generalizes the (first-order) table recomputation method described in [2,21]. Although the authors apply their method in the particular case of an AES implementation, it is generic and can be applied to protect any s-box. Unfortunately, this scheme has been shown to be vulnerable to a 3rd-order attack whatever the chosen masking order [8]. In other words, it only provides 2nd-order security. Further schemes were proposed by Rivain, Dottax and Prouff in [26] with formal security proofs but still limited to 2nd-order security.

The first scheme achieving dth-order security for an arbitrary chosen d has been designed by Ishai, Sahai and Wagner in [14]. The here-called *ISW scheme* consists in masking the Boolean representation of an algorithm which is composed of logical operations NOT and AND. Securing a NOT for any order d is straightforward since $x = \bigoplus_i x_i$ implies $\mathrm{NOT}(x) = \mathrm{NOT}(x_0) \oplus x_1 \cdots \oplus x_d$. The main contribution of [14] is a method to secure the AND operation for any arbitrary order d (the description of this scheme is recalled in Section 2.1). Although the ISW scheme is an important theoretical result, its practical application faces some issues. At the hardware level, the obtained circuits may have prohibitive area requirements, especially for being used in embedded systems (privileged targets of side-channel attacks). Moreover, Mangard *et al.* have shown in [19,20] that masking at the hardware level is sensitive to *glitches* which induce unpredicted flaws in masked circuits. Preventing glitches can be done thanks to synchronization elements (*e.g.* registers or latches) [24] or by performing additional sharing [23] but in both cases, the circuit size is still significantly increased. On the other hand, a direct application of the ISW scheme to secure an s-box computation in software would consist in taking the Boolean representation of the s-box and in processing every logical operation successively in a masked way. Since the Boolean representation of common s-boxes involves a huge number of logical operations, the resulting implementation would likely be inefficient.

In the particular case of AES, a solution has been proposed by Rivain and Prouff in [27] to efficiently mask the s-box processing at any order. Specifically, the authors use the algebraic structure of the AES s-box, which is the composition of an affine function over $\mathbb{F}_2^8$ with the power function $x \mapsto x^{254}$ over $\mathbb{F}_{256}$, and they show that it can be expressed as a sequence of operations involving a few linear functions over $\mathbb{F}_2^8$ (easy to mask) and four multiplications over $\mathbb{F}_{256}$. The latter are secured by applying the ISW scheme (generalized to $\mathbb{F}_{256}$). Subsequently, Kim, Hong and Lim have presented in [15] an extension of Rivain and

Prouff's scheme, which is based on the tower-field approach from [28]. On the other hand, Genelle, Prouff and Quisquater have proposed in [12] a higher-order scheme based on the alternate use of Boolean masking and multiplicative masking. Although schemes in [15] and [12] achieve better performances than [27], they are still restricted to the AES s-box and their generalization to any s-box (or subclasses) is an open issue.

Our Contribution. The present paper introduces the first higher-order masking scheme which can be applied to efficiently protect any s-box processing in software. We first give a general method that extends the Rivain and Prouff approach to mask any s-box and we introduce a new criterion for an s-box that relates to the best efficiency achievable with our method. Then we give concrete schemes that aim to approach the so-called *masking complexity*. Specifically, we give optimal methods for the set of *power functions*, and we give efficient heuristics for the general case. As an illustration we apply our scheme to the DES and PRESENT s-boxes and we provide implementation results.

2 Higher-Order Masking of any S-Box

In this section, we describe a general method to mask any s-box and we introduce a related *masking complexity* criterion.

2.1 General Method

An s-box is a function from $\{0,1\}^n$ to $\{0,1\}^m$ with $m \leq n$ and n small (typically $n \in \{4,6,8\}$). We shall use the terminology of (n,m) s-box when the dimensions need to be specified. To design a higher-order masking scheme for such a function, our approach is to express it as a sequence of affine functions over $\mathbb{F}_2^n$, and multiplications over $\mathbb{F}_{2^n}$. Such a strategy is always possible since any (n,m) s-box can be represented by a polynomial function $x \mapsto \sum_{i=0}^{2^n-1} a_i x^i$ over $\mathbb{F}_{2^n}$ where the a_i are constant coefficients in $\mathbb{F}_{2^n}$. The a_i can be obtained from the s-box look-up table by applying Lagrange's Interpolation Theorem. When m is strictly lower than n, the m-bit outputs can be embedded into $\mathbb{F}_{2^n}$ by padding them to n-bit outputs (*e.g.* by setting most significant bits to 0). The padding is then removed after the polynomial evaluation. We recall hereafter the Lagrange Interpolation Theorem applied to our context.

Theorem 1 (Lagrange Interpolation). *Let* S *be a function* $\mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$. *Then, for every* $x \in \mathbb{F}_{2^n}$, *we have:*

$$\mathrm{S}(x) = \sum_{\alpha \in \mathbb{F}_{2^n}} \mathrm{S}(\alpha)\ell_\alpha(x) \ , \tag{2}$$

where, for every $\alpha \in \mathbb{F}_{2^n}$, ℓ_α *is defined as:*

$$\ell_\alpha(x) = \prod_{\substack{\beta \in \mathbb{F}_{2^n} \\ \beta \neq \alpha}} \frac{x-\beta}{\alpha-\beta} \ . \tag{3}$$

Remark 1. The ℓ_α are called the *Lagrange basis polynomials* and satisfy $\ell_\alpha(x) = 1$ if $x = \alpha$ and $\ell_\alpha(x) = 0$ otherwise. In particular, every ℓ_α is a monic polynomial of degree $2^n - 1$, and we have $\ell_\alpha(x) = (x+\alpha)^{2^n-1} + 1$. Moreover, the coefficients of $\mathrm{S}(x)$ can be directly computed from the Mattson-Solomon polynomial by:

$$a_i = \begin{cases} \mathrm{S}(0) & \text{if } i = 0 \\ \sum_{k=0}^{2^n-2} \mathrm{S}(\alpha^k)\alpha^{-ki} & \text{if } 1 \leq i \leq 2^n - 2 \\ \mathrm{S}(1) + \sum_{i=0}^{2^n-2} a_i & \text{if } i = 2^n - 1 \end{cases}$$

for every primitive element α of $\mathbb{F}_{2^n}$.

The polynomial representation of an s-box is based on four kinds of operations over $\mathbb{F}_{2^n}$: additions, scalar multiplications (*i.e.* multiplications by constants), squares, and regular multiplications (*i.e.* of two different variables). Except for the latter, all these operations are $\mathbb{F}_2^n$*-linear* (or $\mathbb{F}_2^n$*-affine*), that is the corresponding function over $\mathbb{F}_2^n$ are linear (resp. affine). The processing of any s-box can then be performed as a sequence of $\mathbb{F}_2^n$-affine functions (themselves composed of additions, squares and scalar multiplications over $\mathbb{F}_{2^n}$) and of regular multiplications over $\mathbb{F}_{2^n}$, called *nonlinear multiplications* in the following. Masking an s-box processing can hence be done by masking every affine function and every nonlinear multiplication independently. We recall hereafter how this can be done for each category.

Masking of $\mathbb{F}_2^n$-affine functions. Let $x = \sum_i x_i$ be a shared variable. Every affine function g with additive part c_g satisfies:

$$g(x) = \begin{cases} \sum_{i=0}^{d} g(x_i) & \text{if } d \text{ is even,} \\ c_g + \sum_{i=0}^{d} g(x_i) & \text{if } d \text{ is odd.} \end{cases}$$

The masked processing of g then simply consists in evaluating g for every share x_i, and possibly correcting one of them by addition of c_g. Such a processing clearly achieves dth-order security as the shares are all processed independently.

Masking of nonlinear multiplications. Every nonlinear multiplication can be processed by using the ISW scheme. Let $a, b \in \mathbb{F}_{2^n}$ and let $(a_i)_{0 \leq i \leq d}$ and $(b_i)_{0 \leq i \leq d}$ be dth-order encoding of a and b. To securely compute a dth-order encoding $(c_i)_{0 \leq i \leq d}$ of $c = ab$, the ISW method over $\mathbb{F}_{2^n}$ performs as follows:[5]

1. For every $0 \leq i < j \leq d$, pick up a random value $r_{i,j}$ in $\mathbb{F}_{2^n}$.
2. For every $0 \leq i < j \leq d$, compute $r_{j,i} = (r_{i,j} + a_i b_j) + a_j b_i$.
3. For every $0 \leq i \leq d$, compute $c_i = a_i b_i + \sum_{j \neq i} r_{i,j}$.

It can be checked that the obtained shares are a sound encoding of c. Namely, we have:

$$\sum_{i=0}^{d} c_i = \big(\sum_{i=0}^{d} a_i\big)\big(\sum_{i=0}^{d} b_i\big) = ab = c.$$

[5] The use of brackets indicates the order in which the operations are performed, which is mandatory for the security of the scheme.

In [14] it is shown that the above computation achieves $(d/2)$th-order security. A tighter security proof is given in [27] which shows that dth-order security is actually achieved as long as the masks of the two inputs are independent.

Remark 2. Another method to process a masked multiplication at an arbitrary order is used in [10] to achieve provable security under specific leakage assumptions. However this method requires more operations and more random bits than the ISW scheme does. For this reason, the ISW scheme should be preferred in a usual dth-order security model.

2.2 Masking Complexity

The scheme described in the previous section secures the computation of any (n, m) s-box S by masking its polynomial representation over $\mathbb{F}_{2^n}$. The evaluation of such a polynomial is composed of $\mathbb{F}_2^n$-affine functions g and of nonlinear multiplications. The masked processing of each $\mathbb{F}_2^n$-affine function g merely involves $d+1$ evaluations of g itself, while it involves $(d+1)^2$ field multiplications, $2d(d+1)$ field additions and the generation of $nd(d+1)/2$ random bits for each nonlinear multiplication. The masked processing of $\mathbb{F}_2^n$-affine functions hence quickly becomes negligible compared to the masked processing of nonlinear multiplications as d grows. This observation motivates the following definition of the *masking complexity* for an s-box.

Definition 1 (Masking Complexity). *Let m and n be two integers such that $m \leq n$. The* masking complexity *of a (n, m) s-box is the minimal number of nonlinear multiplications required to evaluate its polynomial representation over $\mathbb{F}_{2^n}$.*

The following proposition directly results from this definition.

Proposition 1. *The masking complexity of an s-box is invariant when composed with $\mathbb{F}_2^n$-affine bijections in input and/or in output.*

Remark 3. Since field isomorphisms are $\mathbb{F}_2$-linear bijections, the choice of the irreducible polynomial to represent field elements does not impact the masking complexity of an s-box.

In the next sections, we address the issue of finding polynomial evaluations of an s-box that aim at minimizing the number of nonlinear multiplications. Those constructions will enable us to deduce upper bounds on the masking complexity of an s-box. We first study the case of power functions whose polynomial representation has a single monomial (*e.g.* the AES s-box). For these functions, we exhibit the exact masking complexity by deriving addition chains with minimal number of nonlinear multiplications. We then address the general case and provide efficient heuristics to evaluate any s-box with a low number of nonlinear multiplications.

3 Optimal Masking of Power Functions

In this section, we consider s-boxes for which the polynomial representation over $\mathbb{F}_{2^n}$ is a single monomial. These s-boxes are usually called *power functions* in the literature. We describe a generic method to compute the masking complexity of such s-boxes. Our method involves the notion of *cyclotomic class*.

Definition 2. *Let $\alpha \in [0; 2^n - 2]$. The* cyclotomic class *of α is the set C_α defined by:*

$$C_\alpha = \{\alpha \cdot 2^i \bmod 2^n - 1;\ i \in [0; n-1]\}.$$

We have the following proposition.

Proposition 2. *Let $\mu(m)$ denote the multiplicative order of 2 modulo m and let φ denote the Euler's totient function. For every divisor δ of $2^n - 1$, the number of distinct cyclotomic classes $C_\alpha \subseteq [0; 2^n - 2]$ with $\gcd(\alpha, 2^n - 1) = \delta$ is $\varphi\left(\frac{2^n-1}{\delta}\right)/\mu\left(\frac{2^n-1}{\delta}\right)$. It follows that the total number of distinct cyclotomic classes of $[0; 2^n - 2]$ equals:*

$$\sum_{\delta | (2^n-1)} \frac{\varphi(\delta)}{\mu(\delta)}\ .$$

Proof. Proposition 2 can be deduced from the following facts:

- An integer $\alpha \in [0; 2^n - 2]$ satisfies $\gcd(\alpha, 2^n - 1) = \delta$ if and only if $\alpha = \delta\beta$, with $\gcd(\beta, \frac{2^n-1}{\delta}) = 1$. There are thus $\varphi\left(\frac{2^n-1}{\delta}\right)$ integers $\alpha \in [0; 2^n - 2]$ such that $\gcd(\alpha, 2^n - 1) = \delta$.
- For any α such that $\gcd(\alpha, 2^n - 1) = \delta$ (hence of the form $\alpha = \delta\beta$ with $\gcd(\beta, \frac{2^n-1}{\delta}) = 1$), we have $\alpha \cdot 2^i \equiv \alpha \cdot 2^j \bmod 2^n - 1$ if and only if $\beta \cdot 2^i \equiv \beta \cdot 2^j \bmod \frac{2^n-1}{\delta}$, that is, if and only if $2^i \equiv 2^j \bmod \frac{2^n-1}{\delta}$. Hence C_α has cardinality $\#C_\alpha = \mu\left(\frac{2^n-1}{\delta}\right)$.

The set of integers $\alpha \in [0; 2^n - 2]$ such that $\gcd(\alpha, 2^n - 1) = \delta$ is partitioned into cyclotomic classes, each of them having cardinality $\mu\left(\frac{2^n-1}{\delta}\right)$. Hence the number of such cyclotomic classes is $\varphi\left(\frac{2^n-1}{\delta}\right)/\mu\left(\frac{2^n-1}{\delta}\right)$. It follows that the total number of distinct cyclotomic classes of $[0; 2^n - 2]$ equals $\sum_{\delta|(2^n-1)} \varphi\left(\frac{2^n-1}{\delta}\right)/\mu\left(\frac{2^n-1}{\delta}\right) = \sum_{\delta|(2^n-1)} \varphi(\delta)/\mu(\delta)$.

□

The study of cyclotomic classes is interesting in our context since a power x^α can be computed from a power x^β without any nonlinear multiplication if and only if α and β lie in the same cyclotomic class. Hence, all the power functions with exponents within a given cyclotomic class have the same masking complexity and computing the masking complexity for all the power functions over $\mathbb{F}_{2^n}$ thus amounts to compute this complexity for each cyclotomic class over $\mathbb{F}_{2^n}$. In what follows, we perform such a computation for fields $\mathbb{F}_{2^n}$ of small dimensions n.

To compute the masking complexity for an element in a cyclotomic class, we use the following observation: determining the masking complexity of a power function $x \mapsto x^\alpha$ amounts to find the addition chain for α with the least number of additions which are not doublings (see [16] for an introduction to addition chains). This kind of addition chain is usually called a 2-*addition chain*.[6] Let $(\alpha_i)_i$ denote some addition chain. At step i, it is possible to obtain any element within the cyclotomic classes $(C_{\alpha_j})_{j \leq i}$ using doublings only. As we are interested in finding the addition chain with the least number of additions which are not doublings, the problem we need to solve is the following: given some $\alpha \in C_\alpha$, find the shortest chain $C_{\alpha_0} \to C_{\alpha_1} \to \cdots \to C_{\alpha_k}$ where $C_{\alpha_0} = C_1$, $C_{\alpha_k} = C_\alpha$ and for every $i \in [1; k]$, there exists $j, \ell < i$ such that $\alpha_i = \alpha'_j + \alpha'_\ell$ where $\alpha'_j \in C_{\alpha_j}$ and $\alpha'_\ell \in C_{\alpha_\ell}$.

We shall denote by $\mathcal{M}_k^n$ the class of exponents α such that $x \mapsto x^\alpha$ has a masking complexity equal to k. The family of classes $(\mathcal{M}_k^n)_k$ is a partition of $[0; 2^n - 2]$ and each $\mathcal{M}_k^n$ is the union of one or several cyclotomic classes. For a small dimension n, we can proceed by exhaustive search to determine the shortest 2-addition chain(s) for each cyclotomic class. We implemented such an exhaustive search from which we obtained the masking complexity classes $\mathcal{M}_k^n$ for $n \leq 11$ (note that in practice most s-boxes have dimension $n \leq 8$). Table 1 summarizes the obtained results for $n \in \{4, 6, 8\}$ (usual dimensions). Results for other dimensions are summarized in appendix. Additionally, Table 2 gives the optimal 2-addition chains (in exponential notation) corresponding to every cyclotomic class for $n = 8$.

It is interesting to note that for every n, the *inverse function* $x \mapsto x^{2^n-2}$ related to the cyclotomic class $C_{2^{n-1}-1}$ always has the highest masking complexity. In particular, the inverse function $x \mapsto x^{254}$ (for $n = 8$) used in the AES has a masking complexity of 4 as it was conjectured in [27].

4 Efficient Heuristics for General S-Boxes

We now address the general case of an s-box having a polynomial representation $\sum_{j=0}^{2^n-1} a_j x^j$ over $\mathbb{F}_{2^n}$. A straightforward solution is to successively compute every power x^j using $x^j = (x^{j/2})^2$ if j is even and $x^j = x^{j-1}x$ if j is odd, while updating the polynomial value by adding the monomial $a_j x^j$ at every step. Such a method requires $2^{n-1} - 1$ nonlinear multiplications. As we show hereafter, less naive methods exist that substantially lower the number of nonlinear multiplications. We propose two different methods and then compare their efficiency.

[6] This problem has been studied in the general setting where the multiplication by q (and not specifically by 2) is considered *free* and the obtained addition chains are called *q-addition chains* [31]. The purpose is to find efficient exponentiation methods in $\mathbb{F}_q$ (as in such field the Frobenius map $x \mapsto x^q$ is efficient). To the best of our knowledge, apart from a specific application to the SFLASH signature algorithm in [1], the case of 2-addition chains has not been particularly investigated.

Table 1. Cyclotomic classes for $n \in \{4, 6, 8\}$ w.r.t. the masking complexity k.

k	Cyclotomic classes in $\mathcal{M}_k^n$
	$n = 4$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8\}$
1	$C_3 = \{3, 6, 12, 9\}$, $C_5 = \{5, 10\}$
2	$C_7 = \{7, 14, 13, 11\}$
	$n = 6$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8, 16, 32\}$
1	$C_3 = \{3, 6, 12, 24, 48, 33\}$, $C_5 = \{5, 10, 20, 40, 17, 34\}$, $C_9 = \{9, 18, 36\}$
2	$C_7 = \{7, 14, 28, 56, 49, 35\}$, $C_{11} = \{11, 22, 44, 25, 50, 37\}$, $C_{13} = \{13, 26, 52, 41, 19, 38\}$, $C_{15} = \{15, 30, 29, 27, 23\}$, $C_{21} = \{21, 42\}$, $C_{27} = \{27, 54, 45\}$
3	$C_{23} = \{23, 46, 29, 58, 53, 43\}$, $C_{31} = \{31, 62, 61, 59, 55, 47\}$
	$n = 8$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8, 16, 32, 64, 128\}$
1	$C_3 = \{3, 6, 12, 24, 48, 96, 192, 129\}$, $C_5 = \{5, 10, 20, 40, 80, 160, 65, 130\}$, $C_9 = \{9, 18, 36, 72, 144, 33, 66, 132\}$, $C_{17} = \{17, 34, 68, 136\}$
2	$C_7 = \{7, 14, 28, 56, 112, 224, 193, 131\}$, $C_{11} = \{11, 22, 44, 88, 176, 97, 194, 133\}$, $C_{13} = \{13, 26, 52, 104, 208, 161, 67, 134\}$, $C_{15} = \{15, 30, 60, 120, 240, 225, 195, 135\}$, $C_{19} = \{19, 38, 76, 152, 49, 98, 196, 137\}$, $C_{21} = \{21, 42, 84, 168, 81, 162, 69, 138\}$, $C_{25} = \{25, 50, 100, 200, 145, 35, 70, 140\}$, $C_{27} = \{27, 54, 108, 216, 177, 99, 198, 141\}$, $C_{37} = \{37, 74, 148, 41, 82, 164, 73, 146\}$, $C_{45} = \{45, 90, 180, 105, 210, 165, 75, 150\}$, $C_{51} = \{51, 102, 204, 153\}$, $C_{85} = \{85, 170\}$
3	$C_{23} = \{23, 46, 92, 184, 113, 226, 197, 139\}$, $C_{29} = \{29, 58, 116, 232, 209, 163, 71, 142\}$, $C_{31} = \{31, 62, 124, 248, 241, 227, 199, 143\}$, $C_{39} = \{39, 78, 156, 57, 114, 228, 201, 147\}$, $C_{43} = \{43, 86, 172, 89, 178, 101, 202, 149\}$, $C_{47} = \{47, 94, 188, 121, 242, 229, 203, 151\}$, $C_{53} = \{53, 106, 212, 169, 83, 166, 77, 154\}$, $C_{55} = \{55, 110, 220, 185, 115, 230, 205, 155\}$, $C_{59} = \{59, 118, 236, 217, 179, 103, 206, 157\}$, $C_{61} = \{61, 122, 244, 233, 211, 167, 79, 158\}$, $C_{63} = \{63, 126, 252, 249, 243, 231, 207, 159\}$, $C_{87} = \{87, 174, 93, 186, 117, 234, 213, 171\}$, $C_{91} = \{91, 182, 109, 218, 181, 107, 214, 173\}$, $C_{95} = \{95, 190, 125, 250, 245, 235, 215, 175\}$, $C_{111} = \{111, 222, 189, 123, 246, 237, 219, 183\}$, $C_{119} = \{119, 238, 221, 187\}$
4	$C_{127} = \{127, 254, 253, 251, 247, 239, 223, 191\}$

Table 2. Optimal 2-addition chains (in exponential notation) for cyclotomic classes for $n = 8$.

k	2-addition chains with k nonlinear multiplications
1	$x^3 \leftarrow x \times x^2$ – $x^5 \leftarrow x \times x^4$ $x^9 \leftarrow x \times x^8$ – $x^{17} \leftarrow x \times x^{16}$
2	$x^7 \leftarrow x \times x^2 \times x^4$ – $x^{11} \leftarrow x \times x^2 \times x^8$ $x^{13} \leftarrow x \times x^4 \times x^8$ – $x^{15} \leftarrow x^3 \times (x^3)^4$ $x^{19} \leftarrow x \times x^2 \times x^{16}$ – $x^{21} \leftarrow x \times x^4 \times x^{16}$ $x^{27} \leftarrow x^3 \times (x^3)^8$ – $x^{37} \leftarrow x \times x^4 \times x^{32}$ $x^{45} \leftarrow x^5 \times (x^5)^8$ – $x^{51} \leftarrow x^3 \times (x^3)^{16}$ $x^{85} \leftarrow x^5 \times (x^5)^{16}$
3	$x^{23} \leftarrow x \times x^2 \times x^4 \times x^{16}$ – $x^{29} \leftarrow x \times x^4 \times x^8 \times x^{16}$ $x^{31} \leftarrow x^3 \times (x^3)^4 \times x^{16}$ – $x^{29} \leftarrow x \times x^2 \times x^4 \times x^{32}$ $x^{43} \leftarrow x \times x^2 \times x^8 \times x^{32}$ – $x^{47} \leftarrow x^3 \times (x^3)^4 \times x^{32}$ $x^{53} \leftarrow x \times x^2 \times x^{16} \times x^{32}$ – $x^{55} \leftarrow x^3 \times x^4 \times (x^3)^{16}$ $x^{59} \leftarrow x^3 \times (x^3)^8 \times x^{32}$ – $x^{59} \leftarrow x^5 \times x^{16} \times (x^5)^8$ $x^{63} \leftarrow x^7 \times (x^7)^8$ – $x^{87} \leftarrow x^2 \times x^5 \times (x^5)^{16}$ $x^{91} \leftarrow x^3 \times (x^3)^8 \times x^{64}$ – $x^{95} \leftarrow x^5 \times (x^5)^2 \times (x^5)^{16}$ $x^{111} \leftarrow x^3 \times (x^3)^4 \times (x^3)^{32}$ – $x^{63} \leftarrow x^7 \times (x^7)^{16}$
4	$x^{127} \leftarrow x^3 \times (x^3)^4 \times (x^3)^{16} \times x^{64}$

4.1 Cyclotomic Method

Let q denote the number of distinct cyclotomic classes of $[0; 2^n - 2]$. The polynomial representation of S can be written as:

$$\mathrm{S}(x) = a_0 + \Big(\sum_{i=1}^{q} Q_i(x) \Big) + a_{2^n-1}\, x^{2^n-1} \ ,$$

where the Q_i are polynomials such that every Q_i has powers from a single cyclotomic class C_{α_i}, namely we can write $Q_i(x) = \sum_j a_{i,j} x^{\alpha_i 2^j}$ for some coefficients $a_{i,j}$ in $\mathbb{F}_{2^n}$. Let us then denote L_i the linearized polynomial $L_i(x) = \sum_j a_{i,j} x^{2^j}$ which is a $\mathbb{F}_2^n$-linear function of x. We have $Q_i(x) = L_i(x^{\alpha_i})$ by definition. The *cyclotomic method* simply consists in deriving the powers x^{α_i} for each cyclotomic class C_{α_i} as well as x^{2^n-1} if $a_{2^n-1} \neq 0$, and in evaluating $\mathrm{S}(x) = a_0 + \Big(\sum_{i=1}^{q} L_i(x^{\alpha_i}) \Big) + a_{2^n-1}\, x^{2^n-1}$. The powers x^{α_i} can each be derived with a single nonlinear multiplication. This is obvious for the α_i lying in $\mathcal{M}_1^n$. Then it is clear that every power x^{α_i} with $\alpha_i \in \mathcal{M}_{k+1}^n$ can be derived with a single multiplication from the powers $(x^{\alpha_i})_{\alpha_i \in \mathcal{M}_k^n}$. The power x^{2^n-1} can then be derived with a single nonlinear multiplication from the power x^{2^n-2}. The cyclotomic method hence involves a number of nonlinear multiplications equal to the number of cyclotomic classes, minus 2 (as x^0 and x^1 are obtained without nonlinear multiplication), plus 1 (to derive x^{2^n-1}). By Proposition 2, we then have the following result.

Proposition 3 (Cyclotomic Method). *Let m and n be two positive integers such that $m \leq n$. The masking complexity of every (n, m) s-box is upper-bounded by:*

$$\sum_{\delta \mid (2^n-1)} \frac{\varphi(\delta)}{\mu(\delta)} - 1 \ .$$

An (n, m) s-box S is said to be *balanced* if for every $y \in \{0,1\}^m$, the number of preimages of y for S is constant to 2^{n-m}. The following lemma gives a well-known folklore result.

Lemma 1. *Let m and n be two positive integers such that $m \leq n$. The polynomial representation of every balanced (n, m) s-box has degree strictly lower than $2^n - 1$.*

Proof. Since Lagrange basis polynomials are all monic of degree $2^n - 1$, the coefficient a of the power to the $2^n - 1$ in the polynomial representation of S satisfies $a = \sum_{\alpha \in \mathbb{F}_{2^n}} \mathrm{S}(\alpha)$, which equals 0 if S is balanced. □

When the polynomial representation of the s-box has degree strictly lower than $2^n - 1$, the cyclotomic method saves one nonlinear multiplication since the power x^{2^n-1} is not required. Namely, we have the following corollary of Proposition 3.

Corollary 1 (Cyclotomic Method). *Let m and n be two positive integers such that $m \leq n$ and let* S *be a (n, m) s-box. If* S *is balanced, then the masking complexity of* S *is upper-bounded by:*

$$\sum_{\delta|(2^n-1)} \frac{\varphi(\delta)}{\mu(\delta)} - 2 \ .$$

4.2 Parity-Split Method

The *parity-split method* is composed of two stages. The first stage derives a set of powers $(x^j)_{j \leq q}$ for some q using the straightforward method described in the introduction of this section. The second stage essentially consists in an application of the Knuth-Eve polynomial evaluation algorithm [9, 17] which is based on a recursive use of the following lemma.

Lemma 2. *Let n and t be two positive integers and let Q be a polynomial of degree t over $\mathbb{F}_{2^n}[x]$. There exist two polynomials Q_1 and Q_2 of degree upper-bounded by $\lfloor t/2 \rfloor$ over $\mathbb{F}_{2^n}[x]$ such that:*

$$Q(x) = Q_1(x^2) + Q_2(x^2)x \ . \tag{4}$$

By applying Lemma 2 to the polynomial representation of S, we get $\mathrm{S}(x) = Q_1(x^2) + Q_2(x^2)x$, where Q_1 and Q_2 are two polynomials of degrees upper-bounded by $2^{n-1} - 1$. We deduce that S can be computed based on the set of powers $(x^{2j})_{j \leq 2^{n-1}-1}$ plus a single multiplication by x. Then, applying Lemma 2 again to the polynomials Q_1 and Q_2 both of degrees upper bounded by $2^{n-1} - 1$, we get two new pairs of polynomials (Q_{11}, Q_{12}) and (Q_{21}, Q_{22}) such that $Q_1(x^2) = Q_{11}(x^4) + Q_{12}(x^4)x^2$ and $Q_2(x^2) = Q_{21}(x^4) + Q_{22}(x^4)x^2$. The degrees of the new polynomials are upper bounded by $2^{n-2} - 1$. We then deduce that S can be computed based on the set of powers $(x^{4j})_{j \leq 2^{n-2}-1}$ plus 1 multiplication by x and 2 multiplications by x^2. Eventually, by applying Lemma 2 recursively r times, we get an evaluation of S involving evaluations in x^{2^r} of polynomials of degrees upper-bounded by $2^{n-r} - 1$, plus $\sum_{i=0}^{r-1} 2^i = 2^r - 1$ multiplications by powers of x of the form x^{2^i} with $i \leq r - 1$. The overall evaluation of S hence requires $2^r - 1$ nonlinear multiplications (the x^{2^i} being obtained with squares only) plus the evaluation in x^{2^r} of polynomials of degrees upper-bounded by $2^{n-r} - 1$. The latter evaluation can be performed by first deriving all the powers $(x^{2^r j})_{j \leq 2^{n-r}-1}$ and then evaluating the polynomials (which only involves scalar multiplications and additions once the powers have been derived). For every $j \leq 2^{n-r} - 1$, the powers $(x^{2^r j})_{j \leq 2^{n-r}-1}$ can be computed successively from $y = x^{2^r}$ by $y^j = (y^{j/2})^2$ if j is even and $y^j = y^{j-1}x$ if j is odd. This takes some squares plus $2^{n-r-1} - 1$ nonlinear multiplications (*i.e.* one per odd integer in $[3, 2^{n-r} - 1]$).

We then deduce the following proposition.

Proposition 4. *Let m and n be two positive integers such that $m \leq n$. The masking complexity of every (n, m) s-box is upper-bounded by:*

$$\min_{0 \leq r \leq n} (2^{n-r-1} + 2^r) - 2 = \begin{cases} 3 \cdot 2^{(n/2)-1} - 2 & \text{if } n \text{ is even,} \\ 2^{(n+1)/2} - 2 & \text{if } n \text{ is odd.} \end{cases} \quad (5)$$

Note that the value of r for which the minimum is reached in (5) is $r = \lfloor \frac{n}{2} \rfloor$.

4.3 Comparison

Table 3 summarizes the number of nonlinear multiplications obtained by the cyclotomic method (for balanced s-boxes) and by the parity-split method. We see that the cyclotomic method works better for small dimensions ($n \leq 5$) and the parity-split method for higher dimensions ($n \geq 6$). Furthermore, the superiority of the parity-split method becomes significant as n grows.

Table 3. Number of nonlinear multiplications w.r.t. the evaluation method.

Method \ n	3	4	5	6	7	8	9	10	11
Cyclotomic	1	3	5	11	17	33	53	105	192
Parity-Split	2	4	6	10	14	22	30	46	62

We emphasize that these bounds may not be optimal, namely they may be higher than the maximum masking complexity of (n, m) s-boxes. We let open the issue of finding more efficient (or provably optimal) methods in the general case for further research.

5 Application to DES and PRESENT

In this section we apply the proposed methods to the s-boxes of two different block ciphers: the well-known and still widely used Data Encryption Standard (DES) [11], and the lightweight block cipher PRESENT [5]. The former uses eight different $(6, 4)$ s-boxes and the latter uses a single $(4, 4)$ s-box. According to Table 3, we shall prefer the parity-split method for the DES s-boxes (10 nonlinear multiplications), and the cyclotomic method for the PRESENT s-box (3 nonlinear multiplications).

5.1 Parity-Split Method on DES S-boxes

The parity-split method on a DES s-box uses a polynomial representation of the s-box over $\mathbb{F}_{64}$ which satisfies:

$$\begin{aligned} S \,:\, x \longmapsto\; & Q_0(x^8) + Q_1(x^8) \cdot x^4 + \left(Q_2(x^8) + Q_3(x^8) \cdot x^4\right) \cdot x^2 \\ & + \left(Q_4(x^8) + Q_5(x^8) \cdot x^4 + \left(Q_6(x^8) + Q_7(x^8) \cdot x^4\right) \cdot x^2\right) \cdot x \end{aligned} \quad (6)$$

where the Q_i are degree-7 polynomials, namely, there exist coefficients $a_{i,j}$ for $0 \leq i, j \leq 7$ such that:

$$Q_i(x^8) = a_{i,0} + a_{i,1}x^8 + a_{i,2}x^{16} + a_{i,3}x^{24} + a_{i,4}x^{32} + a_{i,5}x^{40} + a_{i,6}x^{48} + a_{i,7}x^{56} \ .$$

We first derive the powers x^{8j} for $j = 1, 2, \ldots, 7$, which is done at the cost of 3 nonlinear multiplications by:

$$x^8 \leftarrow ((x^2)^2)^2;\ x^{16} \leftarrow (x^8)^2;\ x^{24} \leftarrow x^8 \cdot x^{16};\ x^{32} \leftarrow (x^{16})^2;$$
$$x^{40} \leftarrow x^8 \cdot x^{32};\ x^{48} \leftarrow (x^{24})^2;\ x^{56} \leftarrow x^8 \cdot x^{48};$$

Then we evaluate each polynomial $Q_i(x^8)$ as a linear combination of the above powers. Finally, we evaluate (6) at the cost of 7 nonlinear multiplications and a few additions. The nonlinear multiplications are computed using the ISW scheme over $\mathbb{F}_{64}$ such as recalled in Section 2.1. A detailed implementation for the overall masked s-box evaluation is given in the extended version of this paper.

5.2 Cyclotomic Method on PRESENT S-box

The cyclotomic method on the PRESENT s-box starts from the straightforward polynomial representation of the s-box over $\mathbb{F}_{16}$:

$$\mathrm{S} \ : \ x \ \longmapsto \ a_0 + a_1 x + a_2 x^2 + \cdots + a_{14} x^{14} \ ,$$

(where the degree is indeed strictly lower than 15 by Lemma 1). We then have:

$$\mathrm{S}(x) = a_0 + L_1(x) + L_3(x^3) + L_5(x^5) + L_7(x^7) \ . \tag{7}$$

where:

$$\begin{aligned} L_1 : x \ &\mapsto \ a_1 x + a_2 x^2 + a_4 x^4 + a_8 x^8 \\ L_3 : x \ &\mapsto \ a_3 x + a_6 x^2 + a_{12} x^4 + a_9 x^8 \\ L_5 : x \ &\mapsto \ a_5 x + a_{10} x^2 \\ L_7 : x \ &\mapsto \ a_7 x + a_{14} x^2 + a_{13} x^4 + a_{11} x^8 \end{aligned}$$

and the L_i are $\mathbb{F}_2^4$-linear.

We first derive the powers x^3, x^5, and x^7, which is done at the cost of 3 nonlinear multiplications by: $x^3 \leftarrow x \cdot x^2;\ x^5 \leftarrow x^3 \cdot x^2;\ x^7 \leftarrow x^5 \cdot x^2$. Then we evaluate (7) which costs a few linear transformations and additions. A detailed implementation for the overall masked s-box evaluation is given in the extended version of this paper.

5.3 Implementation Results

In this section, we give implementation results for our scheme applied to DES and PRESENT s-boxes. For comparison, we also give performances of some

higher-order masking schemes for the AES s-box, as well as performances of existing schemes for DES and PRESENT s-boxes at orders 1 and 2. For the AES s-box processing, we implemented Rivain and Prouff's method [27] and its improvement by Kim *et al.* [15]. We did not implement Genelle *et al.* 's scheme [12] since it addresses the masking of an overall AES and is not interesting while focusing on a single s-box processing. Regarding existing schemes for DES and PRESENT s-boxes, we implemented the generic methods proposed in [25] (for $d = 1$) and in [26] (for $d = 2$). We also implemented the improvement of these schemes described in [26, §3.3] that consists in treating two 4-bit outputs at the same time.[7] Note that we did not implement the table re-computation method (for $d = 1$) since it only makes sense for an overall cipher and not for a single s-box processing.

Table 4 lists the timing/memory performances of the different implementations. We wrote the codes in assembly language for an `8051` based 8-bit architecture with bit-addressable memory. `ROM` consumptions (*i.e.* code sizes) are not listed since they are not prohibitive.

As expected, the cyclotomic method is very efficient when applied to protect the PRESENT s-box. The small input dimension of the s-box indeed implies a low masking complexity (equal to 3). Moreover, it enables to tabulate the multiplication over $\mathbb{F}_{16}$. At first order, it is even slightly better than the method in [25] (or its improvement). At second order, the cost of the secure multiplications involved in the cyclotomic method is approximatively doubled, which explains that the overall cost is multiplied by 1.8. This makes it less efficient than [25] and [26], which are less impacted by the increase of the masking order from 1 to 2. At third order, our method is the only one. The number of cycles staying small (630), Table 4 shows that achieving resistance against 3rd-order side-channel analysis is realistic for an implementation of PRESENT on a 8051 architecture. For DES s-boxes, the parity-split method is less efficient than the state-of-the art methods for $d = 1, 2$. This is an expected consequence of the high number of nonlinear multiplications (here 10) achieved with the parity-split method in dimension 6 and of the fact that the field multiplications can no longer be tabulated (and must therefore be computed thanks to log/alog look-up tables). At third order, the timing efficiency of the method becomes very low. The masked s-box processing is 5 (resp. 10) times slower than the efficiency of the AES s-box protected thanks to [15] (resp. [27]), though its input dimension is smaller.

The ranking of the timing efficiencies for AES, DES and PRESENT s-boxes is correlated to the number of nonlinear multiplications in the used scheme (3, 4-5, and 10, for PRESENT, AES and DES respectively) which underline the soundness of the masking complexity criterion. Therefore, while selecting an s-box for a block cipher design, one should favor an s-box with small masking complexity if side-channel attacks are taken into account.

[7] This improvement is only described in [26] for $d = 2$ but it can be applied likewise to the 1st-order scheme of [25].

Table 4. Comparison of secure s-box implementations

	Method	Reference	cycles	RAM (bytes)
First Order Masking				
1.	AES s-box	[27]	533	10
2.	AES s-box	[15]	320	14
3.	DES s-box	Simple version [25]	1096	2
4.	DES s-box	Improved version [25] & [26]	439	14
5.	DES s-box	this paper	4100	50
6.	PRESENT s-box	Simple Version [25]	281	2
7.	PRESENT s-box	Improved Version [25] & [26]	231	14
4.	PRESENT s-box	this paper	220	18
Second Order Masking				
1.	AES s-box	[27]	832	18
2.	AES s-box	[15]	594	24
3.	DES s-box	Simple version [26]	1045	69
4.	DES s-box	Improved version [26]	652	39
5.	DES s-box	this paper	7000	78
6.	PRESENT s-box	Simple Version [26]	277	21
7.	PRESENT s-box	Improved Version [26]	284	15
8.	PRESENT s-box	this paper	400	31
Third Order Masking				
1.	AES s-box	[27]	1905	28
2.	AES s-box	[15]	965	38
3.	DES s-box	this paper	10500	108
4.	PRESENT s-box	this paper	630	44

6 Discussion

In previous sections we have introduced the first schemes that can be used to mask any s-box at any order with fair performances in software. In particular, these schemes enable to apply higher-order masking on random s-boxes (*e.g.* the DES s-boxes) which have no specific mathematical structure. Prior to our work, the only existing methods were the circuit-oriented proposals of Ishai *et al.* [14] and of Faust *et al.* [10]. The main purpose of these works was a proof of concept for applying higher-order masking to circuits with formal security proofs, but they did not address efficient implementation. A direct application of [14] or [10] to a block cipher consists in taking its Boolean representation and in replacing every XOR and AND with $O(d)$ and $O(d^2)$ logical operations respectively (where d is the masking order). Applying such a strategy in software leads to inefficient implementation as the Boolean representation of an s-box includes a huge number of nonlinear gates (with a $O(d^2)$ factor to be protected). Compared to these techniques, our schemes achieve significant improvements. These are obtained by starting from the field representation of the s-box and applying methods to significantly reduce the number of nonlinear multiplications compared to the

Boolean representation of the s-box. For instance, we have shown that a DES s-box can be computed with 10 nonlinear multiplications whereas its Boolean representation involves several dozens of logical AND operations.

We believe that our work opens up new avenues for research in block cipher implementations and side-channel security. In particular, the issue of designing s-boxes with low masking complexity and good cryptographic criteria is still to be investigated. On the other hand, our work could be extended to take into account more general definitions of the masking complexity. Indeed Definition 1 is software oriented and hence does not encompass the hardware case. As discussed above, the complexity of masking in hardware merely depends on the number of nonlinear gates [10, 14], that is on the number of nonlinear multiplications in the (n-variate) s-box representation over $\mathbb{F}_2$, the so-called *algebraic normal form*. One may also want to minimize the number of nonlinear multiplications in the (ℓ-variate) s-box representation over $\mathbb{F}_{2^k}$ for some k (and $\ell = \lceil n/k \rceil$). This approach has actually already been followed in [15], where Kim *et al.* speeds up the scheme in [27] by using the fact that the AES s-box can be processed with 5 nonlinear multiplications over $\mathbb{F}_{16}$ rather than 4 nonlinear multiplications over $\mathbb{F}_{256}$. Although requiring an additional nonlinear multiplication, the resulting implementation is faster since multiplications over $\mathbb{F}_{16}$ can be tabulated while multiplications over $\mathbb{F}_{256}$ are computed based on the slower log/alog approach. These observations motivate the following — more general — definition of the masking complexity.

Definition 3 (Masking Complexity). *Let m, n and k be three integers such that $m, k \leq n$. The* masking complexity *of a (n, m) s-box over $\mathbb{F}_{2^k}$ is the minimal number of nonlinear multiplications required to evaluate its polynomial representation over $\mathbb{F}_{2^k}$.*

Here again, the masking complexity is independent of the representation of $\mathbb{F}_{2^k}$ since one can go from one representation to another without any nonlinear multiplication. The issue of finding efficient methods with respect to the masking complexity over a smaller field $\mathbb{F}_{2^k}$ is left open for further researches.

7 Conclusion

In this paper we have introduced new generic higher-order masking schemes for s-boxes with efficient software implementation. Specifically, we have extended the Rivain and Prouff's approach for the AES s-box to any s-box. The method consists in masking the polynomial representation of the s-box over $\mathbb{F}_{2^n}$ where n is the input dimension. As argued, the complexity of this method mainly depends on the number of nonlinear multiplications involved in the polynomial representation (*i.e.* multiplications which are not squares nor scalar multiplications). We have then introduced the masking complexity parameter for an s-box as the minimal number of nonlinear multiplications required for its evaluation. We have provided the exact values of this parameter for the set of power functions and upper bounds for all s-boxes. Namely, we have presented optimal methods to

mask power functions and efficient heuristics for the general case. Eventually we have applied our schemes to the DES s-boxes and to the PRESENT s-box and we have provided implementation results. Our work stresses interesting open issues for further research. Among them the design of s-boxes taking into account the masking complexity criterion and the extension of our approach to masking over $\mathbb{F}_{2^k}$ with $k < n$ (*e.g.* for efficient hardware implementations) are of particular interest.

References

1. M.-L. Akkar, N. Courtois, R. Duteuil, and L. Goubin. A Fast and Secure Implementation of Sflash. In Y. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2003.
2. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
3. G. Blakley. Safeguarding cryptographic keys. In *National Comp. Conf.*, volume 48, pages 313–317, New York, June 1979. AFIPS Press.
4. J. Blömer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
5. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
6. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
7. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
8. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.
9. J. Eve. The evaluation of polynomials. *Comm. ACM*, 6(1):17–21, 1964.
10. S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, 2010.
11. FIPS PUB 46. *The Data Encryption Standard*. National Bureau of Standards, Jan. 1977.

12. L. Genelle, E. Prouff, and M. Quisquater. Thwarting Higher-Order Side Channel Analysis with Additive and Multiplicative Maskings. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 240–255. Springer, 2011.
13. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
14. Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
15. H. Kim, S. Hong, and J. Lim. A Fast and Provably Secure Higher-Order Masking of AES S-Box. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 2011.
16. D. Knuth. *The Art of Computer Programming*, volume 2. Addison Wesley, third edition, 1988.
17. D. E. Knuth. Evaluation of polynomials by computers. *Comm. ACM*, 5(12):595–599, 1962.
18. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
19. S. Mangard, T. Popp, and B. M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
20. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
21. T. Messerges. Securing the AES Finalists against Power Analysis Attacks. In B. Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
22. T. Messerges. Using Second-order Power Analysis to Attack DPA Resistant Software. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
23. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. In P. J. Lee and J. H. Cheon, editors, *Information Security and Cryptology – ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2008.
24. T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2007.
25. E. Prouff and M. Rivain. A Generic Method for Secure SBox Implementation. In S. Kim, M. Yung, and H.-W. Lee, editors, *Information Security Applications – WISA 2007*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2008.

26. M. Rivain, E. Dottax, and E. Prouff. Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In T. Baignères and S. Vaudenay, editors, *Fast Software Encryption – FSE 2008*, Lecture Notes in Computer Science, pages 127–143. Springer, 2008.
27. M. Rivain and E. Prouff. Provably Secure Higher-Order Masking of AES. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
28. A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In E. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 239–254. Springer, 2001.
29. K. Schramm and C. Paar. Higher Order Masking of the AES. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.
30. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
31. J. von zur Gathen. Efficient and Optimal Exponentiation in Finite Fields. *Computational Complexity*, 1:360–394, 1991.

A Masking Complexity of Power Functions

Table 5 summarizes the masking complexity classes $(\mathcal{M}_k^n)_k$ for dimensions n in the set $\{3, 5, 7, 9, 10, 11\}$.

Table 5. Cyclotomic classes for $n \in \{3, 5, 7, 9, 10, 11\}$ w.r.t. the masking complexity k.

k	Cyclotomic classes in $\mathcal{M}_k^n$
	$n = 3$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4\}$
1	$C_3 = \{3, 6, 5\}$
	$n = 5$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8, 16\}$
1	$C_3 = \{3, 6, 12, 24, 17\}$, $C_5 = \{5, 10, 20, 9, 18\}$
2	$C_7 = \{7, 14, 28, 25, 19\}$, $C_{11} = \{11, 22, 13, 26, 21\}$, $C_{15} = \{15, 30, 29, 27, 23\}$
	$n = 7$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8, 16, 32, 64\}$
1	$C_3 = \{3, 6, 12, 24, 48, 96, 65\}$, $C_5 = \{5, 10, 20, 40, 80, 33, 66\}$, $C_9 = \{9, 18, 36, 72, 17, 34, 68\}$
2	$C_7 = \{7, 14, 28, 56, 112, 97, 67\}$, $C_{11} = \{11, 22, 44, 88, 49, 98, 69\}$, $C_{13} = \{13, 26, 52, 104, 81, 35, 70\}$, $C_{15} = \{15, 30, 60, 120, 113, 99, 71\}$, $C_{19} = \{19, 38, 76, 25, 50, 100, 73\}$, $C_{21} = \{21, 42, 84, 41, 82, 37, 74\}$, $C_{27} = \{27, 54, 108, 89, 51, 102, 77\}$, $C_{43} = \{43, 86, 45, 90, 53, 106, 85\}$
3	$C_{23} = \{23, 46, 92, 57, 114, 101, 75\}$, $C_{29} = \{29, 58, 116, 105, 83, 39, 78\}$, $C_{31} = \{31, 62, 124, 121, 115, 103, 79\}$, $C_{47} = \{47, 94, 61, 122, 117, 107, 87\}$, $C_{55} = \{55, 110, 93, 59, 118, 109, 91\}$, $C_{63} = \{63, 126, 125, 123, 119, 111, 95\}$
	$n = 9$
0	C_0, C_1
1	C_3, C_5, C_9, C_{17}
2	C_7, C_{11}, C_{13}, C_{15}, C_{19}, C_{21}, C_{25}, C_{27}, C_{35}, C_{37}, C_{41}, C_{45}, C_{51}, C_{73}, C_{75}, C_{83}, C_{85}
3	C_{23}, C_{29}, C_{31}, C_{39}, C_{43}, C_{47}, C_{53}, C_{55}, C_{57}, C_{59}, C_{61}, C_{63}, C_{75}, C_{77}, C_{79}, C_{87}, C_{91}, C_{93}, C_{95}, C_{103}, C_{107}, C_{109}, C_{111}, C_{117}, C_{119}, C_{123}, C_{125}, C_{127}, C_{171}, C_{175}, C_{183}, C_{187}, C_{219}
4	C_{191}, C_{223}, C_{239}
	$n = 10$
0	C_0, C_1
1	C_3, C_5, C_9, C_{17}, C_{33}
2	C_7, C_{11}, C_{13}, C_{15}, C_{19}, C_{21}, C_{25}, C_{27}, C_{35}, C_{37}, C_{41}, C_{45}, C_{49}, C_{51}, C_{69}, C_{73}, C_{85}, C_{99}, C_{147}, C_{165}
3	C_{23}, C_{29}, C_{31}, C_{39}, C_{43}, C_{47}, C_{53}, C_{55}, C_{57}, C_{59}, C_{61}, C_{63}, C_{71}, C_{75}, C_{77}, C_{79}, C_{83}, C_{87}, C_{89}, C_{91}, C_{93}, C_{95}, C_{101}, C_{103}, C_{105}, C_{107}, C_{109}, C_{111}, C_{115}, C_{117}, C_{119}, C_{121}, C_{123}, C_{125}, C_{149}, C_{151}, C_{155}, C_{157}, C_{167}, C_{171}, C_{173}, C_{175}, C_{179}, C_{181}, C_{183}, C_{187}, C_{189}, C_{205}, C_{207}, C_{213}, C_{215}, C_{219}, C_{221}, C_{231}, C_{235}, C_{237}, C_{245}, C_{255}, C_{341}, C_{347}, C_{363}, C_{447}, C_{495}
4	C_{127}, C_{159}, C_{191}, C_{223}, C_{239}, C_{247}, C_{251}, C_{253}, C_{343}, C_{351}, C_{367}, C_{375}, C_{379}, C_{383}, C_{439}, C_{479}, C_{511}
	$n = 11$
0	C_0, C_1
1	C_3, C_5, C_9, C_{17}, C_{33}
2	C_7, C_{11}, C_{13}, C_{15}, C_{19}, C_{21}, C_{25}, C_{27}, C_{35}, C_{37}, C_{41}, C_{45}, C_{49}, C_{51}, C_{67}, C_{69}, C_{73}, C_{81}, C_{85}, C_{99}, C_{137}, C_{153}, C_{163}, C_{165}, C_{293}
3	C_{23}, C_{29}, C_{31}, C_{39}, C_{43}, C_{47}, C_{53}, C_{55}, C_{57}, C_{59}, C_{61}, C_{63}, C_{71}, C_{75}, C_{77}, C_{79}, C_{83}, C_{87}, C_{89}, C_{91}, C_{93}, C_{95}, C_{101}, C_{103}, C_{105}, C_{107}, C_{109}, C_{111}, C_{113}, C_{115}, C_{117}, C_{119}, C_{121}, C_{123}, C_{125}, C_{139}, C_{141}, C_{143}, C_{147}, C_{149}, C_{151}, C_{155}, C_{157}, C_{167}, C_{169}, C_{171}, C_{173}, C_{175}, C_{179}, C_{181}, C_{185}, C_{187}, C_{189}, C_{199}, C_{201}, C_{203}, C_{205}, C_{207}, C_{211}, C_{213}, C_{217}, C_{219}, C_{221}, C_{229}, C_{231}, C_{243}, C_{245}, C_{255}, C_{295}, C_{299}, C_{301}, C_{307}, C_{309}, C_{311},C_{315}, C_{317}, C_{331}, C_{333}, C_{335}, C_{343}, C_{347}, C_{359}, C_{363}, C_{365}, C_{379}, C_{411}, C_{423}, C_{427}, C_{429}, C_{339}, C_{341}, C_{437}, C_{439}, C_{469}, C_{495}, C_{683}, C_{703}, C_{879}, C_{887}
4	C_{127}, C_{159}, C_{183}, C_{191}, C_{215}, C_{223}, C_{233}, C_{235}, C_{237}, C_{239}, C_{247}, C_{249}, C_{251}, C_{253}, C_{303}, C_{319}, C_{349}, C_{351}, C_{367}, C_{371}, C_{373}, C_{375}, C_{381}, C_{383}, C_{413}, C_{415}, C_{431}, C_{443}, C_{445}, C_{447}, C_{463}, C_{471}, C_{475}, C_{477}, C_{479}, C_{491}, C_{493}, C_{501}, C_{503}, C_{507}, C_{509}, C_{511}, C_{687}, C_{695}, C_{699}, C_{727}, C_{731}, C_{735}, C_{751}, C_{759}, C_{763}, C_{767}, C_{895}, C_{959}, C_{991}, C_{1023}

Appendix C

Masking against Side Channel Attacks: a Formal Security Proof

Hereafter is appended the full version of our paper [PR13], joint work with Emmanuel Prouff published at **EUROCRYPT 2013**.

Masking against Side-Channel Attacks: a Formal Security Proof*

Emmanuel Prouff[1] and Matthieu Rivain[2]

[1] ANSSI
emmanuel.prouff@ssi.gouv.fr

[2] CryptoExperts
matthieu.rivain@cryptoexperts.com

Abstract. Masking is a well-known countermeasure to protect block cipher implementations against side-channel attacks. The principle is to randomly split every sensitive intermediate variable occurring in the computation into $d + 1$ shares, where d is called the masking order and plays the role of a security parameter. Although widely used in practice, masking is often considered as an empirical solution and its effectiveness is rarely proved. In this paper, we provide a formal security proof for masked implementations of block ciphers. Specifically, we prove that the information gained by observing the leakage from one execution can be made negligible (in the masking order). To obtain this bound, we assume that every elementary calculation in the implementation leaks a noisy function of its input, where the amount of noise can be chosen by the designer (yet polynomially bounded). We further assume the existence of a leak-free component that can refresh the masks of shared variables. Our work can be viewed as an extension of the seminal work of Chari *et al.* published at CRYPTO in 1999 on the soundness of combining masking with noise to thwart side-channel attacks.

1 Introduction

Side-channel analysis is a class of cryptanalytic attacks that exploit the physical environment of a cryptosystem to recover some *leakage* about its secrets. It is often more efficient than a cryptanalysis in the so-called *black-box model* in which no leakage occurs. Two attack categories are usually considered: the *bounded side-channel attacks* and the *continuous side-channel attacks*. In a bounded side-channel attack [9], the total amount of leakage accessible to the adversary is bounded. In a continuous side-channel attack, the adversary gets some information at each invocation of the cryptosystem, and the amount of leakage can thus be arbitrarily large. Attacks where the adversary measures the running-time [24], the power consumption [25] or the electromagnetic radiations [15] of a cryptographic implementation fall into this category.

Continuous side-channel attacks have proved to be especially effective to break unprotected cryptographic implementations. And although many ingenious countermeasures have been developed during past years, very few of them gave rise to concrete security guaranties. This has raised the need for models and methods to formally prove the security of cryptographic implementations against continuous side-channel attacks. A pioneering work in this direction is the *physically observable cryptography* framework introduced by Micali and Reyzin in [29] which puts forward a general theory of side-channel attacks. In particular, they formalize the assumptions that a cryptographic device can at least keep some secrets and that *only*

* Full version of the paper published in the proceedings of EUROCRYPT 2013.

computation leaks information [29]. A few years later, Dziembowski and Pietrzak introduced the *leakage resilient cryptography* model [12], which is a generalization of the *bounded retrieval model* [9] where every step of the computation leaks information on the processed part of the device state trough a function whose range is bounded (*i.e.* taking values in $\{0,1\}^\lambda$ for some parameter λ). Under this assumption, the authors were able to design secure pseudo-random number generators [12,32]. Further leakage resilient cryptographic primitives were then constructed under the same – or sometimes stronger – assumptions (see for instance [10,13,23,42]). The issue of designing generic compilers that can transform any cryptographic algorithm into a leakage resilient implementation was also recently addressed [11,17,18,22]. These works are nice proofs of concept but the proposed constructions are not suited for practical implementation, especially in constrained environments such as embedded systems. Moreover the practical meaning of the underlying bounded range leakage model with respect to power or electromagnetic side channels is questionable [40].

A more practical and traditionally used approach to secure implementations against side-channel attacks is *secret sharing* [1,37] also called *masking* in this context. The idea is to randomly split a secret into several shares such that the adversary needs all of them to reconstruct the secret. Masking was soon identified as a sound countermeasure when side-channel attacks appeared in the literature [4,19]. Since then, many works have been published to address the practical implementation and/or the security of masking for various ciphers. However a formal security proof is still missing at this day. Our goal is to fill this gap.

1.1 Related Works

Soundness of masking. In [4], Chari *et al.* conduct a formal study of masking in the presence of noisy leakage. More precisely, the authors investigate the soundness of randomly sharing a secret bit into d shares when the adversary has only access to a noisy version of those shares, the noise having a normal distribution with variance σ^2. They prove that the number of observations required to distinguish, with success probability α the leakage distribution when the secret equals 0 from the leakage distribution when the secret equals 1 is lower bounded by $\sigma^{d+4\log\alpha/\log\sigma}$. This bound is frequently recalled to argue for the soundness of masking when combined with noise. Despite its great interest and impact, Chari *et al.*'s analysis has an important limitation: no solution is provided to apply masking to the whole implementation of a cryptographic algorithm and to analyze the global security of such an implementation.

Private circuits and extensions. In [21], Ishai *et al.* show that any circuit with n logical gates can be transformed into a circuit of size $O(nd^2)$ which is secure against *probing attacks* spying up to d wires. The main contribution of [21] is a method to compute an AND gate between shared inputs while ensuring the security against a d-probing adversary. However, a security proof against probing attacks does not give full satisfaction since it does not encompass an adversary exploiting the entire leakage produced during the processing.

In [14], Faust *et al.* propose an extension of Ishai *et al.*'s scheme. Their scheme requires a leak-free hardware component but it is provably secure under two different and more general leakage models. In the first model, the leakage at each cycle is any function of the circuit internal state (*i.e.* the logical values carried by all the wires) which is computationally bounded: it is assumed to be in the complexity class AC^0 (*i.e.* it must be computable by a circuit of constant depth). In the second model, the leakage reveals the values of each internal state bit flipped with a probability $p < 1/2$ (*i.e.* xor-ed with a p-Bernoulli noise). In a recent paper [35], Rothblum further showed how to remove the requirement of leak-free component in the AC^0 leakage model. These works achieve an important progress towards provable security against side-channel attacks since it shows that masking can bring security even in the presence of a global leakage on the entire state. However, the practicability of the considered models is questionable. In particular it is unclear whether the AC^0 leakage or the full leakage with Benouilli noise really fit the physical reality of power and/or electromagnetic leakages.

Masking schemes. On the other hand, several works have shown how to apply masking to various algorithms in practice. They however often omit to prove the security of the resulting implementations. The first masking scheme was proposed by Goubin and Patarin in [19] for the DES cipher. Further schemes were subsequently published in which masking is applied at hardware or software level at the cost of different area-time-memory trade-offs (see for instance [2, 28, 30]). Originally, most of these schemes deal with *first-order masking* which splits each sensitive variable in two shares (a mask and a masked variable). Then *higher-order masking schemes* were defined to get security against side-channel attacks exploiting the leakage of several, say d, intermediate computations [3, 16, 33, 34]. The purpose of these schemes is analogous to the d-probing secure circuit of Ishai *et al.* : the computation is performed such that any d intermediate variables occurring in the algorithm reveal no sensitive information. Most of these schemes are actually based on the method of Ishai *et al.* to securely process a multiplication between two shared variables. Consequently, they suffer the same limitation as Ishai *et al.* 's scheme: they only thwart a limited adversary that does not exploit the overall leakage.

1.2 Our Contribution

In this paper we formally prove the security of masked implementations of block ciphers in the *only computation leaks information* model [29]. In this model, every *step* of the processing reveals a leakage function of the touched part of the device state. This function is chosen adaptively by the adversary in some pre-determined class. For our security proof, we split the computation into several *elementary calculations* (in practice, a sequence of few CPU instructions) that each accesses a subpart x of the device state and leaks a function of x. Starting from the observation that masking is sound when combined with noise [4] and that many practical solutions exist to amplify leakage noise (see for instance [6–8, 20, 27, 39, 41]), we assume the leakage functions to be *noisy*. The noisy feature of a leakage function f is captured by assuming that an observation of $f(x)$ only implies a *bounded bias* in the probability distribution of x. Namely the statistical distance between the distributions $\mathrm{P}[x]$ and

$\mathrm{P}[x|f(x)]$ is bounded. We further assume that this bound depends on a noise parameter ω that can be chosen by the designer according to the required security level. Our security proof has a natural limitation which is the requirement of a leak-free component, an elementary calculation refreshing the masks of a shared variable. Under these assumptions, we achieve an information theoretic security proof: we show that the mutual information between the cipher input (plaintext and secret key) and the overall leakage on the block cipher processing is upper bounded by $\omega^{-(d+1)}$, where d is the masking order.

This bound can be seen as an extension of the seminal work of Chari *et al.* [4] as it is derived from the combination of masking with noise. We extend their analysis in two ways. First we consider a more general leakage model which no longer requires particular assumptions (single-bit target variable, Gaussian noise). More importantly, we provide a security bound for a full masked block cipher implementation whereas Chari *et al.* analysis focus on leaking shares independently of any computation. Our work can also be viewed as an alternative to previous works on program or circuit compilers with formal security proofs against side-channel attacks [11,14,17,18,22,35]. Whereas the practical meaning of the leakage models considered in these works is questionable, our leakage model aims to be compliant with practical investigations about side-channel leakage (see for instance [27,31,36,38]).

2 Preliminaries

Calligraphic letters, like $\mathcal{X}$, are used to denote finite sets. The corresponding large letter X is used to denote a random variable over $\mathcal{X}$, while the lower-case letter x denotes a particular element from $\mathcal{X}$. To every discrete random variable X, one associates a probability mass function P_X defined by $\mathrm{P}_X(x) = \mathrm{P}[X = x]$. Let Y be a random variable defined over some set $\mathcal{Y}$ and let $y \in \mathcal{Y}$. Then $(X|Y = y)$ denotes the random variable with probability mass function $x \mapsto \mathrm{P}[X = x|Y = y]$. The *entropy* (or *Shannon entropy*) $\mathrm{H}[X]$ of a discrete random variable X is defined by $\mathrm{H}[X] = -\sum_{x\in\mathcal{X}} \mathrm{P}_X(x) \log_2(\mathrm{P}_X(x))$. The *mutual information* between two random variables X and Y is then defined by $\mathrm{I}(X;Y) = \mathrm{H}[X]-\mathrm{H}[X|Y]$, where $\mathrm{H}[X|Y]$ is called the *conditional entropy of X given Y* and is defined by $\mathrm{H}[X|Y] = \sum_{y\in\mathcal{Y}} \mathrm{P}_Y(y)\, \mathrm{H}[(X|Y = y)]$. The *statistical distance* between two random variables X_0 and X_1 is denoted by $\mathrm{d}(X_0;X_1)$ and is defined by $\mathrm{d}(X_0;X_1) = \|\mathrm{P}_{X_0} - \mathrm{P}_{X_1}\|$ where $\|\cdot\|$ denotes the Euclidean norm and P_{X_i} denotes the vector $(\mathrm{P}_{X_i}(x))_{x\in\mathcal{X}}$. We recall that for any N-dimensional vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$ we have

$$\|\boldsymbol{y}\| \leq L_1(\boldsymbol{y}) \leq \sqrt{N}\|\boldsymbol{y}\| \ , \tag{1}$$

where $L_1(\boldsymbol{y})$ denotes the *Manhattan norm* $\sum_i |y_i|$.

We now introduce the notion of *bias* that is extensively used in our security proof.

Definition 1. *Let X and Y be two random variables. The* bias of X given $Y = y$*, denoted $\beta(X|Y = y)$, is defined as*

$$\beta(X|Y = y) = \mathrm{d}\big(X;(X|Y = y)\big) \ .$$

The bias of X given Y, $\beta(X|Y)$, *is then defined as the expected bias of* X *given* $Y = y$ *over* $\mathcal{Y}$, *that is*

$$\beta(X|Y) = \sum_{y \in \mathcal{Y}} \mathrm{P}_Y(y)\beta(X|Y = y) \; .$$

The bias of X given Y is an information metric between X and Y. If X and Y are independent then $\beta(X|Y)$ equals zero. Moreover, as shown in the next proposition, it is related to the mutual information between X and Y (the proof is given in appendix).

Proposition 1. *Let* X *and* Y *be two random variables, with* X *uniformly distributed over a set* $\mathcal{X}$ *of cardinality* N. *The mutual information between* X *and* Y *satisfies* $\mathrm{I}(X;Y) \leq \frac{N}{\ln 2}\beta(X|Y)$.

3 Model of Leaking Computation

We describe hereafter our model of leaking computation.

An algorithm is modelled by a sequence of *elementary calculations* $(\mathsf{C}_i)_i$ that are Turing machines augmented with a common random access memory called *the state*. Each elementary calculation reads its input and writes its output on the state. When an elementary calculation C_i is invoked, its input is written from the state to its input tape, then C_i is executed, afterwards its output is written back to the state.

A *physical implementation* of an algorithm is modelled by a sequence of *physical elementary calculations*. A physical elementary calculation $(\mathsf{C}_i, f_i)_i$ is composed of an elementary calculation C_i and a *leakage function* f_i. A leakage function is defined as a function that takes two parameters: the value held by the accessed part of the state and a random string long enough to model the leakage noise. Let $\mathcal{I} = (\mathsf{C}_i, f_i)_i$ be a physical implementation of an algorithm $\mathcal{A}$ as defined above. Each execution of $\mathcal{I}$ leaks the values $\big(f_i(x_i, r_i)\big)_i$ where x_i denotes the input of C_i and r_i is a fresh random string. In particular all the r_i involved in successive executions of $\mathcal{I}$ are uniformly and independently drawn.

For the sake of simplicity, we shall omit the random string parameter, which leads to the notation $f_i(x)$ where x is the accessed value. Note that $f_i(x)$ is not the result of a function but it can be seen as the output of a probabilistic Turing machine. In particular, $f_i(x)$ can take several values with a given probability distribution, and is therefore considered as a random variable in the following. Let $\mathcal{X}$ be the definition set of the accessed part of the state. We shall then say that f_i is defined over $\mathcal{X}$ (or equivalently that $\mathcal{X}$ is the domain of f_i).

For our security proof, we will consider special classes of leakage functions that we shall call *noisy leakage functions*. Let f be a leakage function defined over some set $\mathcal{X}$ and let X denote a uniform random variable over $\mathcal{X}$. The noisy property of f is captured by assuming that the bias introduced in the distribution of X given the leakage $f(X)$ is bounded.[3] For any positive real number ε, we define the class

[3] For the above definition of noisy leakage functions to be sound, we need to precise the distribution of X while bounding $\beta(X|f(X))$, and a natural choice is the uniform distribution. Of course, this does not constrain the leakage function to only apply on uniformly distributed inputs.

of *noisy leakage functions w.r.t. bias* ε, and we denote by $\mathcal{N}(\varepsilon)$, the set of noisy functions such that $\beta(X|f(X)) \leq \varepsilon$. In this paper, we shall assume that the designer can constrain as willing the set of noisy leakage functions related to any elementary calculation by linearly increasing the amount of noise in the leakage. More precisely, we assume that the designer controls a noise parameter ω such that an elementary calculation C can yield a physical elementary calculation (C, f) with $f \in \mathcal{N}(1/\omega)$, where ω is linear in the security parameter.

3.1 Discussion

Our model can be seen as a specialization of the physically observable cryptography framework [29] with leakage functions belonging to the class of noisy functions as defined above (the similarities between our model and this framework are discussed in Appendix). Our model is also comparable to the leakage resilient cryptography model [12] with two important differences relating to the computation granularity and the class of leakage functions.

Computation granularity. As nicely explained in [17], a computation in the *only computation leaks* model is divided into several *sub-computations* and a leakage function applies to the input of each sub-computation. In [12, 32] the authors construct stream ciphers that output an unbounded number of key-stream blocks from a secret key block. A sub-computation is then naturally identified as the computation of one block. In contrast, we consider a finer granularity: the computation of one block (*i.e.* a single block cipher computation) is divided into several elementary calculations that each leaks a function of its input. In other words, [12,32] address the issue of constructing secure protocols from a cryptographic primitive with limited leakage whereas we address the issue of constructing secure cryptographic primitives from elementary calculations with noisy leakages.

Class of leakage functions. The most noticeable difference between our work and the previous leakage-resilient constructions resides in the considered class of leakage functions. Most previous works consider the class of (polynomial-time) bounded-range functions where a function takes values in $\{0,1\}^\lambda$ for some parameter λ [10–13, 17, 18, 22, 23, 32]. This hypothesis is conservative in terms of security since it encompasses leakage functions with complex structures. However its practical meaning is unclear with regard to power and electromagnetic side channels for which the leakage is usually substantially larger than the secret state (but hopefully does not contain its overall entropy).

In contrast, several techniques are known to add some noise in the side-channel leakage in practice [6–8, 20, 26, 27, 39, 41]. By noise addition we mean that the relevant signal in the leakage is lowered compared to random variations, although this may not literally result from noise addition (the terminology of *hiding* is sometimes used). This motivates our definition of noisy leakage functions. Note that in practice, power and electromagnetic leakages can realistically be modeled by a multivariate deterministic function g of the processed data with an additional multivariate Gaussian noise N [5, 27, 36, 38]. The class $\mathcal{N}(\varepsilon)$ corresponding to such a leakage function can then be determined from the description of g and N.

4 Masked Implementation of Block Ciphers

Several schemes have been proposed to securely process a block cipher composed of linear layers and non-linear s-boxes. In this paper, we prove the security of the scheme described in [3] with a secure multiplication processing close to those of [14, 21], and with additional mask-refreshing computations. Before presenting the masking scheme, we start by formalizing the considered block cipher processing.

4.1 Block Cipher Processing

A block cipher is a cryptographic algorithm which, from a secret key, transforms a plaintext block into a ciphertext block. In this paper, we focus on block ciphers designed as a succession of linear functions and substitution boxes (s-boxes). S-boxes are nonlinear functions from $\{0,1\}^n$ to $\{0,1\}^m$ with $m \leq n$ and n small (typically $n \in \{4,6,8\}$). We assume that the addition law is the bitwise addition, denoted $\oplus$, and that the s-boxes are *balanced*; namely every $y \in \{0,1\}^m$ has the same number of preimages in $\{0,1\}^n$ under the s-box. In the computation model introduced in Section 3, the processing of such a block cipher is represented as a sequence of elementary calculations, each of them implementing either a linear function or an s-box. The input of this processing is a pair composed of the plaintext and the secret key and the output is the corresponding ciphertext.

Uniformity property. We shall assume that the block cipher satisfies the following uniformity property: a uniform distribution of the cipher input (plaintext-key pair) induces a uniform distribution of the input of every of its elementary calculation. The uniformity property is satisfied by classical block cipher designs such as DES and AES.

4.2 Securing the Block Cipher Processing

We start with the following definition that formalizes the notion of *d^{th}-order encoding*.

Definition 2 (d^{th}-order encoding). *Let d be a positive integer and let I denote the integer interval $\{0,1,\ldots,d\}$. The* d^{th}-order encoding of $x \in \mathcal{X}$ *is a $(d+1)$-tuple $(x_i)_{i\in I}$ satisfying $\bigoplus_i x_i = x$ and such that $(x_i)_{i\in I\setminus\{i_0\}}$ is uniformly distributed over $\mathcal{X}^d$ for every $i_0 \in I$.*

Masking a block cipher implementation consists in choosing a security parameter d (called *masking order*) and in performing the computation on a d^{th}-order encoding of the state. Namely, the plaintext and the secret key are split into $d+1$ shares. Then, a scheme is defined that specifies how each elementary calculation is replaced by a sequence of new elementary calculations operating only on the shares. At the end, the new sequence must return an encoding of the ciphertext (from which the actual ciphertext can be straightforwardly recovered).

According to the block cipher model of Section 4.1, we describe hereafter how to process a linear function and an s-box computation on shared inputs as proposed in [3]. We first introduce the mask-refreshing component used at several steps in the masking scheme and which is assumed to be *leak-free* in our security proof.

Mask refreshing. Our scheme requires a special kind of elementary calculation to refresh the masks of an encoding *without leaking information*. This mask-refreshing oracle is denoted by $\mathcal{O}^{\$}$. From an encoding of any value x, it computes a new encoding of x with fresh random values. For the sake of simplicity, we assume that when invoked the input and output of our leak-free component do not leak. However this assumption could be relaxed since the input comes from a previous elementary calculation and the output is used in the subsequent elementary calculations (otherwise its masks would not need to be refreshed), therefore they both leak at some point in the computation.

Secure linear function processing. To secure the processing of any linear function g, the following process is applied:

1. For every $i \in \{0, 1, \ldots, d\}$, process $z_i \leftarrow g(x_i)$.
2. Output $(z_i)_i \leftarrow \mathcal{O}^{\$}\big((z_i)_i\big)$.
3. If the encoding $(x_i)_i$ is used subsequently in the block cipher processing, process $(x_i)_i \leftarrow \mathcal{O}^{\$}\big((x_i)_i\big)$.

Secure s-box processing. Let S be an s-box with input dimension n and output dimension $m \leq n$. Then S can be represented by a polynomial function $x \in \mathbb{F}_{2^n} \mapsto \bigoplus_{i=0}^{2^n-1} \alpha_i x^i$ where the α_i are constant coefficients in $\mathbb{F}_{2^n}$. The α_i can be computed from the s-box look-up table by applying Lagrange's Theorem.[4] Thanks to this representation, the s-box calculation can be done by processing four kinds of elementary calculations over $\mathbb{F}_{2^n}$: addition, multiplication by a constant, square, and regular multiplication (*i.e.* of two different elements). The three former kinds of calculations are linear (or affine including the addition by a non-zero constant) and their processing can hence be done exactly as for linear transformations. When the calculation is a regular multiplication, the following scheme is applied.

Secure regular multiplication processing. To secure the processing of a regular multiplication, we use a method similar to that of [14, 21]. The input is a pair of encodings of the multiplication operands a and b. The definition of the sequence of elementary calculations computing the encoding of $a \times b$ is deduced from the following relation: $a \times b = \big(\bigoplus_i a_i\big) \times \big(\bigoplus_i b_i\big) = \bigoplus_{i,j} a_i \times b_j$. It is described hereafter:

1. For every $(i, j) \in \{0, 1, \ldots, d\}^2$, process $v_{i,j} \leftarrow a_i \times b_j$.
2. For every $j \in \{0, 1, \ldots, d\}$, process $(v_{0j}, v_{1j}, \ldots, v_{dj}) \leftarrow \mathcal{O}^{\$}(v_{0j}, v_{1j}, \ldots, v_{dj})$.
3. Process $(v_{00}, v_{01}, \ldots, v_{0d}) \leftarrow \mathcal{O}^{\$}(v_{00}, v_{01}, \ldots, v_{0d})$.
4. For every $i \in \{0, 1, \ldots, d\}$, process $z_i \leftarrow \bigoplus_{j=0}^{d} v_{i,j}$
5. Output $(z_i)_i \leftarrow \mathcal{O}^{\$}\big((z_i)_i\big)$.
6. If one of the input encodings $(a)_i$ and $(b_i)_i$ is involved in a subsequent elementary calculation, then process $(a_i)_i \leftarrow \mathcal{O}^{\$}\big((a_i)_i\big)$ and/or $(b_i)_i \leftarrow \mathcal{O}^{\$}\big((b_i)_i\big)$.

[4] When m is strictly lower than n, the m-bit outputs can be embedded in $\mathbb{F}_{2^n}$ by padding them to n-bit outputs (*e.g.* by setting most significant bits to 0). The padding is then removed after the polynomial evaluation.

Note that Steps 2 and 3 intend to refresh the masks between the subsequences of elementary calculations in Steps 1 and 4. Namely, these steps render the probability distributions $\big((a_i)_i, (b_j)_j | (a,b)\big)$ (in Step 1) and $\big((v_{i,j})_{i,j} | (a,b)\big)$ (in Step 4) mutually independent. Note that Step 3 is mandatory to this aim as Step 2 only makes $\big((v_{i,j})_i | (a,b)\big)$ independent of $\big((a_i)_i, (b_j)_j | (a,b)\big)$ for every column j separately. From this point, Step 3 ensures the global independence.

For our security proof, we shall consider each sum $z_i \leftarrow \bigoplus_{j=0}^{d} v_{i,j}$ in Step 4 as d successive elementary calculations $t_{i,j} \leftarrow t_{i,j-1} \oplus v_{i,j}$ for $1 \leq j \leq d$ with $t_{i,0} = v_{i,0}$ and giving $z_i = t_{i,d}$.

It is clear from the above description that securing a regular multiplication is expensive compared to securing a linear function. The complexity of a secure multiplication is quadratic in d whereas the complexity of a secure linear function is linear in d. Moreover the secure multiplication requires several calls to the mask refreshing oracle. For efficiency purpose, one should hence try to minimize the number of multiplications in the polynomial representation of the s-box. We refer to [3] where efficient heuristics of polynomial evaluation are proposed with respect to this criterion.

5 Main Theorems

It is well-known that any subset of at most d shares X_i gives no information on a secret X encoded at order d, while the whole $d+1$ shares enable to fully reconstruct the secret. In [4], Chari *et al.* consider an adversary which has access to the noisy version of all the shares, i.e. $X_i + N_i$ where N_i is a Gaussian noise with standard deviation σ. They restrict themselves to the case of a single secret bit and show that distinguishing the distribution of the noisy shares associated to a secret bit at 0 and that associated to a secret bit at 1 with a probability $\alpha \in [0,1)$ requires at least $\sigma^{d+4\log\alpha/\log\sigma}$ samples. In other words, the required number of leakage observations increases exponentially with the masking order, the underlying base being the noise standard deviation.

Chari *et al.* 's result demonstrates the soundness of using masking under a practically relevant leakage model. However, they only focus on a static leakage of the shares and not on the leakage occurring while computing on the shares. In this paper, we fill this gap by providing security bounds for masked implementations that process shared variables. As explained in Section 4, a block cipher may be decomposed into linear operations and multiplications in a finite field. We derive hereafter upper bounds on the amount of sensitive information leakage for both operations when protected by masking. Then in Section 6, we derive an upper bound on the information leakage for the full masked implementation of the cipher.

5.1 Sequential Processing of the Shares

Our first context deals with the case where the shares are processed sequentially *e.g.* by applying the same linear function to them. For such a processing, we provide hereafter an upper bound on the bias of the secret value distribution given noisy leakages on its shares.

Theorem 1. *Let X be a uniform random variable over some set $\mathcal{X}$ of cardinality N, let d be a positive integer and let $(X_i)_i$ be a d^{th}-order encoding of X. Let $\varepsilon \in [0,1)$ and let f_0, f_1, ..., f_d be noisy functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \big| f_0(X_0), f_1(X_1), \ldots, f_d(X_d)\big) \leq N^{\frac{d}{2}} \varepsilon^{d+1} .$$

Theorem 1 shows that the bias of X given the leakages on its shares decreases exponentially with the order d, provided that the initial bias is sufficiently low, namely lower than $\frac{1}{\sqrt{N}}$. Seeing the *amount of noise* as the inverse of the bias, we hence obtained an upper-bound that decreases exponentially with the encoding order, the base of the exponent being the amount of noise. This result is in accordance with [4] while being more general since it encompasses any (univariate or multivariate) leakage distribution and any data dimension.

Remark 1. The tuple $\big(f_0(X_0), f_1(X_1), \ldots, f_d(X_d)\big)$ can be seen as the output of a noisy function

$$F : X \mapsto \big(f_0(X \oplus X_1 \oplus \cdots \oplus X_d), f_1(X_1), \ldots, f_d(X_d)\big)$$

where X_1, X_2, ..., X_d are part of the internal randomness of F. Theorem 1 is then equivalent to the following statement: if f_0, f_1, ..., f_d belong to $\mathcal{N}(\varepsilon)$ then F belong to $\mathcal{N}\big(N^{\frac{d}{2}}\varepsilon^{d+1}\big)$.

In some contexts, the shared variable is not uniformly distributed, but it is a deterministic function of a uniform secret variable. This case is addressed in the following corollary of Theorem 1.

Corollary 1. *Let X be a uniform random variable over some set $\mathcal{X}$ of cardinality N and let g be a deterministic function from $\mathcal{X}$ to $\mathcal{X}$. Let d be a positive integer and let $(G_i)_i$ be a d^{th}-order encoding of $g(X)$. Let $\varepsilon \in [0,1)$ and let f_0, f_1, ..., f_d be noisy functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \big| f_0(G_0), f_1(G_1), \ldots, f_d(G_d)\big) \leq \sqrt{2} N^{\frac{d+2}{2}} \varepsilon^{d+1} .$$

Corollary 1 holds as a direct consequence of the above remark and the following lemma.

Lemma 1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets. Let f be a noisy function defined over $\mathcal{Y}$ and belonging to the class $\mathcal{N}(\varepsilon)$ and let g be a deterministic function from $\mathcal{X}$ to $\mathcal{Y}$. Then the noisy function $f \circ g$ defined over $\mathcal{X}$ belongs to the class $\mathcal{N}(\sqrt{2}|\mathcal{Y}| \cdot \varepsilon)$.*

5.2 Multiplication of the Shares

The previous theorem deals with a situation where all the shares leak separately which matches the context of the secure linear functions processing. However it is not sufficient alone to deduce an upper bound for the secure multiplication processing given in Section 4. In the latter case, the secure multiplication of two variables A and B from their respective encodings $(A_i)_i$ and $(B_j)_j$ requires to compute the cross-terms $A_i \times B_j$. Hence each share A_i of A appears in $d+1$ different multiplications, one per share B_j, and *vice versa*. Our strategy is first to bound the bias on each

share A_i and B_j given the multiple leakages on each share. We can then bound the bias of A and B using a similar approach as in Theorem 1, and we finally derive a bound for the bias of the pair (A, B).

We give hereafter our result for the bias given multiple leakages, and then provide our result for the bias given the cross-term leakages.

Bias given multiple leakages. The next theorem deals with the case of repeated leakages on a variable X. We will then apply it in the secure multiplication context.

Theorem 2. *Let X be a uniform random variable defined over a finite set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0, 1)$ and let f_1, f_2, ..., f_t be t noisy functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. For any real number $\alpha \in (0, 1]$, if $\varepsilon \leq \frac{\alpha}{tN}$, then we have*

$$\beta(X|f_1(X), f_2(X), \ldots, f_t(X)) \ \leq \ \left(\left(\frac{\mathrm{e}^\alpha - 1}{\alpha}\right)t + \mathrm{e}^\alpha\right)\varepsilon \ .$$

The bound in Theorem 2 shows that the bias of X given t leakages increases linearly with t. A requirement is that the bias given a single leakage, namely ε, is t times lower than $\frac{1}{N}$ or less, namely $\varepsilon \leq \frac{\alpha}{tN}$ for some $\alpha \in (0, 1]$. Then the bias of X given t leakages is smaller that $\lambda(t)\,\varepsilon$ where λ is an affine function. The value α provides a trade-off between the constraint on ε and the coefficients of λ. If $\alpha = 1$ then $\lambda(t) = (\mathrm{e} - 1)t + \mathrm{e} \approx 1.72\,t + 2.72$, and when α approaches 0, then $\lambda(t)$ tends towards $t + 1$.

Bias given cross-term leakages. The next theorem gives an upper bound on the bias of a uniform pair (A, B) given the leakage $(f_{i,j}(A_i, B_j))_{i,j}$.

Theorem 3. *Let A and B be two random variables uniformly distributed over some finite set $\mathcal{X}$ of cardinality N. Let d be a positive integer, and let $(A_i)_i$ and $(B_j)_j$ be two d^{th}-order encodings of A and B respectively. Let ε be a real number such that $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0, 1]$ and let $(f_{i,j})_{i,j}$ be noisy functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big((A, B)|(f_{i,j}(A_i, B_j))_{i,j}\big) \leq 2N^{\frac{3d+2}{2}}\big((\lambda_1 d + \lambda_0)\varepsilon\big)^{d+1} \ ,$$

where $\lambda_1 = \frac{\mathrm{e}^\alpha - 1}{\alpha}$ and $\lambda_0 = \lambda_1 + \mathrm{e}^\alpha$.

In our context, the pair (A, B) is not uniformly distributed but it is of the form $(A, B) = (g_1(X), g_2(X))$ where X is uniform, and g_1 and g_2 are polynomial functions. We will then use the following corollary of Theorem 3.

Corollary 2. *Let X be a random variable uniformly distributed over some set $\mathcal{X}$ and let g_1 and g_2 be deterministic functions from $\mathcal{X}$ to $\mathcal{X}$. Let d be a positive integer and let $(G_i)_i$ and $(H_j)_j$ be d^{th}-order encodings of $g_1(X)$ and $g_2(X)$ respectively. Let ε be a real number such that $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0, 1]$. And let $(f_{i,j})_{i,j}$ be noisy functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \mid (f_{i,j}(G_i, H_j))_{i,j}\big) \leq 2\sqrt{2}N^{\frac{3d+6}{2}}\big((\lambda_1 d + \lambda_0)\varepsilon\big)^{d+1} \ ,$$

$\lambda_1 = \frac{\mathrm{e}^\alpha - 1}{\alpha}$ *and* $\lambda_0 = \lambda_1 + \mathrm{e}^\alpha$.

Here again, the above corollary is a direct consequence of Lemma 1 (taking $g = (g_1, g_2)$ and $\mathcal{Y} = \mathcal{X} \times \mathcal{X}$).

The bound in Corollary 2 shows that the bias of X given the leakages on all the pairs of shares (A_i, B_j) decreases exponentially with d. A requirement is that the bias given a single leakage, namely ε, is $(d+1)$ times lower than $\frac{1}{N^2}$ or less, namely $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0, 1]$. Then the bias of X given the $(d+1)^2$ leakages is smaller that $(\lambda(d)\ \varepsilon)^{d+1}$ where λ is an affine function. Once again, the value α provides a trade-off between the constraint on ε and the coefficients of λ.

In the next section, we will use the theorems and corollaries introduced above to deduce a security bound for a full masked implementation of block cipher.

6 Overall Security Proof

In this section, we formally prove the security of the physical implementation $\mathcal{I} = (\mathsf{C}_i, f_i)_i$ of a block cipher following the scheme described in Section 4 with masking order d. Our security proof is *information theoretic*: we show that the information about the cipher input (message and secret key) provided by the overall leakage within an execution of $\mathcal{I}$ is upper bounded by a *negligible function* of the masking order d.

The cipher is assumed to involve t_{lin} linear transformations, t_{nlm} nonlinear multiplications and t_{aff} affine functions (in the s-boxes evaluations). The plaintext and the secret key in input of the cipher are modeled as uniform random variables M and K respectively. The input of every elementary calculation C_i is modeled as a random variable X_i. Each elementary calculation leaks a noisy function f_i of X_i. The designer is allowed to choose a noise parameter ψ_i linear in the security parameter d, such that the leakage function f_i lies in the class of noisy functions $\mathcal{N}(1/\psi_i)$.

Theorem 4. *Let d be a positive integer and let $\mathcal{I} = (\mathsf{C}_i, f_i)_{1 \leq i \leq q}$ be the physical implementation of a block cipher under the scheme described in Section 4 with masking order d. For any positive real number ω, there exists a family of real numbers $\psi_i = O(d)\omega$ such that if f_i lies in $\mathcal{N}(1/\psi_i)$ for every i, then the mutual information between the cipher input (M, K) and the overall leakage $(f_1(X_1), f_2(X_2), \ldots, f_q(X_q))$ satisfies*

$$\mathrm{I}\big((M, K); (f_1(X_1), f_2(X_2), \ldots, f_q(X_q))\big) \ \leq \ \frac{T}{\omega^{d+1}}\ , \tag{2}$$

where $T = 2t_{\textbf{nlm}} + t_{\textbf{aff}} + t_{\textbf{lin}}$.

The rest of this section provides a proof of Theorem 4 and exhibits the noise parameters ψ_i that must be chosen for every elementary calculation C_i.

From the description of the scheme in Section 4.2, the overall sequence of elementary calculations of the protected block cipher algorithm can be split into several subsequences separated by masks-refreshing calculations. These subsequences are of four types:

1. $\big(z_i \leftarrow g(x_i)\big)_i$ where g is a linear function of the block cipher,
2. $\big(z_i \leftarrow g(x_i)\big)_i$ where g is an affine function of an s-box evaluation,

3. $\big(v_{i,j} \leftarrow a_i \times b_j\big)_{i,j}$ (first step of a secure nonlinear multiplication),
4. $\big(t_{i,j} \leftarrow t_{i,j-1} \oplus v_{i,j}\big)_{i,j}$ (fourth step of a secure nonlinear multiplication).

All the remaining elementary calculations are masks-refreshing calculations which are leak-free by assumption.

Let us denote by $\mathcal{I}_1$, $\mathcal{I}_1$, ..., $\mathcal{I}_T$ the successive subsequences of elementary calculations and by L_1, L_2, ..., L_T their respective leakages. For every $t \in [1;T]$, the leakage L_t satisfies

$$L_t = \begin{cases} \big(f_i(X_i)\big)_i & \text{if } \mathcal{I}_t \text{ is of type 1 or 2,} \\ \big(f_{i,j}(A_i, B_j)\big)_{i,j} & \text{if } \mathcal{I}_t \text{ is of type 3,} \\ \big(f_{i,j}(T_{i,j-1}, V_{i,j})\big)_{i,j} & \text{if } \mathcal{I}_t \text{ is of type 4.} \end{cases}$$

where the f_i or $f_{i,j}$ are the leakage functions associated to the elementary calculations in $\mathcal{I}_t$ and where the $(X_i)_i$, $(A_i)_i$, $(B_j)_j$, $(V_{i,j})_{i,j}$, or $(T_{i,j})_{i,j}$ are random variables modeling the elementary calculations inputs in $\mathcal{I}_t$ (note that the indexing is intra-subsequence and it differs from that in Theorem 4).

A crucial point of our security proof is that every subsequence operates on shares with fresh random masks. As a result, given a cipher input $(M, K) = (m, k)$, the encodings processed in the different subsequences are mutually independent, which in turn implies that the leakages of the different subsequences are mutually independent (since by definition, the randomness introduced by a noisy function is independent of the randomness introduced by the others). We deduce that the entropy of the overall leakage $(L_1, L_2, \ldots, L_T)$ knowing the cipher input (M, K) satisfies:

$$\mathrm{H}[(L_1, L_2, \ldots, L_T)|(M, K)] = \sum_{t=1}^{T} \mathrm{H}[L_t|(M, K)] \; .$$

The mutual information between the cipher input and the overall leakage therefore satisfies:

$$\begin{aligned} \mathrm{I}\big((M, K); (L_1, L_2, \ldots, L_T)\big) &= \mathrm{H}[(L_1, L_2, \ldots, L_T)] - \sum_{t=1}^{T} \mathrm{H}[L_t|(M, K)] \\ &\leq \sum_{i=1}^{T} \mathrm{I}((M, K); L_t) \; , \end{aligned}$$

where the inequality holds since $\mathrm{H}[(L_t)_t] \leq \sum_t \mathrm{H}[L_t]$.

For every subsequence $\mathcal{I}_t$, there exists a uniform random variable[5] $X_t = g(M, K)$ such that $\mathrm{I}((M, K); L_t) = \mathrm{I}(X_t; L_t)$. To complete the proof of Theorem 4, we now demonstrate that the mutual information $\mathrm{I}(X_t; L_t)$ is upper bounded by $(1/\omega)^{d+1}$ for some noise parameter $\psi_t = O(d)\omega$, for every $t \in \{1, 2, \ldots, T\}$. Due to Proposition 1, this is equivalent to prove that the bias of X_t given L_t satisfies

$$\beta(X_t|L_t) \leq \frac{\ln 2}{N} \left(\frac{1}{\omega}\right)^{d+1} \; , \tag{3}$$

[5] For a subsequence of type 1, this variable is simply the (unmasked) input of the corresponding linear transformation (which is indeed uniform since the cipher input is uniform by assumption, and according to the uniformity property stated in Section 4.1). For a subsequence of type 2, 3 or 4, this variable is the (unmasked) input of the corresponding s-box (which is also uniformly distributed for the same reasons).

where N is the cardinal of the definition set of X_t. In the rest of the section, we demonstrate the claim for every type of subsequence.

Security of type 1 subsequences. In a type 1 subsequence every elementary calculation processes a share of an encoding of the uniform input X_t of a linear function. The security of such a subsequence directly holds from Theorem 1. Indeed, according to Theorem 1 with $\varepsilon = \psi_t^{-1}$, choosing a noise parameter $\psi_t = c\,\omega$ with a constant c satisfying $c^{d+1} \geq (\ln 2)^{-1} N^{\frac{d+2}{2}}$ implies bound (3). In particular c can be taken equal to $(\ln 2)^{-\frac{1}{2}} N^{\frac{3}{4}} \approx 1.44 N^{\frac{3}{4}}$ for any $d \geq 1$ and equal to a value close to $1.44\sqrt{N}$ for a large d.

Security of type 2 subsequences. In a type 2 subsequence every elementary calculation processes a share G_i of an encoding of $g(X_t)$ where X_t is a uniform s-box input and g is some polynomial function.

According to Corollary 1, choosing a noise parameter $\psi_t = c\,\omega$ with a constant c satisfying $c^{d+1} \geq (\ln 2)^{-1} N^{\frac{d+5}{2}}$ implies bound (3). In particular c can be taken equal to $(\ln 2)^{-\frac{1}{2}} N^{\frac{3}{2}} \approx 1.44 N^{\frac{3}{2}}$ for any $d \geq 1$ and equal to a value close to $1.44\sqrt{N}$ for a large d.

Security of type 3 subsequences. In a type 3 subsequence every elementary calculation processes a multiplication from a pair of shares (A_i, B_j), where $(A_i)_i$ and $(B_j)_j$ are d^{th}-order encodings of two variables A and B to multiply. Both A and B rely on a uniform s-box input X_t by $A = g(X_t)$ and $B = h(X_t)$ for some polynomial functions g and h.

According to Corollary 2, choosing a noise parameter $\psi_t = \lambda(d)\,\omega$ for a subsequence $\mathcal{I}_t$ of type 3 implies bound (3), as long as there exists $\alpha \in (0;1]$ such that $\lambda(d)$ satisfies

$$\lambda(d) \geq \frac{N^2}{\alpha\,\omega}(d+1) \quad \text{and} \quad \lambda(d) \geq c(\lambda_{1,\alpha}\,d + \lambda_{0,\alpha})\,,$$

where $\lambda_{1,\alpha} = \frac{\mathrm{e}^\alpha - 1}{\alpha}$, $\lambda_{0,\alpha} = \lambda_{1,\alpha} + \mathrm{e}^\alpha$ and c is a constant satisfying $c^{d+1} \geq 2(\ln 2)^{-1} N^{\frac{3d+9}{2}}$. In particular c can be taken to $(\ln 2)^{-\frac{1}{2}} N^3 \approx 1.44 N^3$ for any $d \geq 1$ and equal to a value close to $1.44 N^{\frac{3}{2}}$ for a large d.

The first constraint aims to meet the requirement on ε for Corollary 2 and the second constraint implies bound (3). To summarize, the best choice is to take λ as

$$\lambda(d) = \min_{\alpha \in (0;1]} \max\left(\frac{N^2}{\alpha\,\omega}(d+1)\,,\; c(\lambda_{1,\alpha}\,d + \lambda_{0,\alpha})\right).$$

Security of type 4 subsequences. In a type 4 subsequence every elementary calculation processes an addition $T_{i,j} = T_{i,j-1} \oplus V_{i,j}$ for $0 \leq i \leq d$ and $1 \leq j \leq d$, and where $T_{i,0} = V_{i,0}$. For every i, the sequence of additions results in a share $Z_i = T_{i,d}$ of the underlying multiplication output. That is $(Z_0, Z_1, \ldots, Z_d)$ is an encoding of $g(X)$ where X is a uniform s-box input and g is some polynomial function over $\mathcal{X}$. Each elementary calculation takes as input a pair $(T_{i,j-1}, V_{i,j})$ and

leaks $f_{i,j}(T_{i,j-1}, V_{i,j})$ where $f_{i,j} \in \mathcal{N}(1/\psi_t)$. Our goal is to set ψ_t such that the bias $\beta\big(X|(F_0(Z_0), F_1(Z_1), \ldots, F_d(Z_d))\big)$ satisfies bound (3), where

$$F_i(Z_i) = \big(f_{i,1}(T_{i,0}, V_{i,1}), f_{i,2}(T_{i,1}, V_{i,2}) \ldots, f_{i,d}(T_{i,d-1}, V_{i,d})\big) \ .$$

Note that F_i can be seen as a noisy function applied to Z_i (and where $V_{i,1}$, $V_{i,2}$, …, $V_{i,d}$ are seen as the internal randomness of F_i).

We first analyse the bias on each Z_i given the leakages $F_i(Z_i)$. Let $f'_{i,j}$ be the noisy functions defined by $f'_{i,j} : (X, Y) \mapsto f_{i,j}(X, X \oplus Y)$. We can then rewrite $F_i(Z_i)$ as

$$F_i(Z_i) = \big(f'_{i,1}(T_{i,0}, T_{i,1}), f'_{i,2}(T_{i,1}, T_{i,2}) \ldots, f'_{i,d}(T_{i,d-1}, T_{i,d})\big) \ ,$$

and we have $f'_{i,j} \in \mathcal{N}(1/\psi_t)$ for every (i, j) by definition of the bias.[6] Moreover $T_{i,0}$, $T_{i,1}$, …, $T_{i,d}$ are independent and uniformly distributed, and $T_{i,d} = Z_i$. We then apply the following lemma.

Lemma 2. *Let T_0, T_1, …, T_d be $d+1$ independent random variables uniformly distributed over some set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0, 1)$ and let f_1, f_2, …, f_d be noisy functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(T_d \big| f_1(T_0, T_1), f_2(T_1, T_2), \ldots, f_d(T_{d-1}, T_d)\big) \leq 2N\varepsilon \ .$$

Lemma 2 implies $\beta(Z_i|F_i(Z_i)) \leq 2N/\psi_t$ for every i. Then by Corollary 1, we get

$$\beta\big(X|(F_0(Z_0), F_1(Z_1), \ldots, F_d(Z_d))\big) \leq N^{\frac{d+3}{2}} \Big(\frac{2N}{\psi_t}\Big)^{d+1} = N^{\frac{3d+5}{2}} \Big(\frac{2}{\psi_t}\Big)^{d+1} \ .$$

According to the above inequality, choosing a noise parameter $\psi_t = c\,\omega$ with a constant c satisfying $c^{d+1} \geq 2^{d+1}(\ln 2)^{-1} N^{\frac{3d+7}{2}}$ implies bound (3). In particular c can be taken equal to $2(\ln 2)^{-1} N^{\frac{5}{2}} \approx 2.89 N^{\frac{5}{2}}$ for any $d \leq 1$ and equal to a value close to $2.89 N^{\frac{3}{2}}$ for a large d.

[6] For every noisy function f defined over $\mathcal{X} \times \mathcal{X}$, and for $f' : (X, Y) \mapsto f(X, X \oplus Y)$, we have $\beta\big((X, Y)|f(X, Y)\big) = \beta\big((X, Y')|f'(X, Y')\big)$ where $Y' = X \oplus Y$.

References

1. G. Blakely. Safeguarding cryptographic keys. In *National Comp. Conf.*, volume 48, pages 313–317, New York, June 1979. AFIPS Press.
2. J. Blömer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
3. C. Carlet, L. Goubin, E. Prouff, M. Quisquater, and M. Rivain. Higher-order masking schemes for s-boxes. In A. Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.
4. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
5. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In B. Kaliski Jr., Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–29. Springer, 2002.
6. C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.
7. J.-S. Coron and I. Kizhvatov. Analysis and Improvement of the Random Delay Countermeasure of CHES 2009. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010.
8. J.-S. Coron, P. Kocher, and D. Naccache. Statistics and secret leakage. In Y. Frankel, editor, *Financial Cryptography – FC 2000*, volume 1962 of *Lecture Notes in Computer Science*. Springer, 2000.
9. G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly Secure Password Protocols in the Bounded Retrieval Model. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 225–244. Springer, 2006.
10. Y. Dodis and K. Pietrzak. Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.
11. S. Dziembowski and S. Faust. Leakage-Resilient Circuits without Computational Assumptions. In R. Cramer, editor, *Theory of Cryptography, 9th Theory of Cryptography Conference – TCC 2012*, volume 7194 of *Lecture Notes in Computer Science*, pages 230–247. Springer, 2012.
12. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.
13. S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-Resilient Signatures. In D. Micciancio, editor, *Theory of Cryptography Conference – TCC 2010*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, 2010.
14. S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, 2010.
15. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
16. L. Genelle, E. Prouff, and M. Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 240–255. Springer, 2011.
17. S. Goldwasser and G. N. Rothblum. Securing computation against continuous leakage. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 59–79. Springer, 2010.
18. S. Goldwasser and G. N. Rothblum. How to Compute in the Presence of Leakage. In *53rd Annual IEEE Symposium on Foundations of Computer Science – FOCS 2012*, pages 31–40. IEEE Computer Society, 2012.
19. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
20. P. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security – ANCS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2006.
21. Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

22. A. Juma and Y. Vahlis. Protecting Cryptographic Keys against Continual Leakage. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2010.
23. E. Kiltz and K. Pietrzak. Leakage Resilient ElGamal Encryption. In M. Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010.
24. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
25. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
26. F. Macé, F.-X. Standaert, and J.-J. Quisquater. Information Theoretic Evaluation of Side-Channel Resistant Logic Styles. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2007.
27. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, 2007.
28. T. Messerges. Securing the AES Finalists against Power Analysis Attacks. In B. Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
29. S. Micali and L. Reyzin. Physically Observable Cryptography (Extended Abstract). In M. Naor, editor, *Theory of Cryptography Conference – TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
30. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In H. Handschuh and H. Gilbert, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.
31. E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Integration*, 40(1):52–60, 2007.
32. K. Pietrzak. A Leakage-Resilient Mode of Operation. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.
33. E. Prouff and T. Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011.
34. M. Rivain and E. Prouff. Provably Secure Higher-Order Masking of AES. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
35. G. N. Rothblum. How to compute under $\mathcal{AC}^0$ leakage without secure hardware. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, 2012.
36. W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
37. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
38. F.-X. Standaert and C. Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.
39. F.-X. Standaert, S. B. Örs, and B. Preneel. Power Analysis of an FPGA: Implementation of Rijndael: Is Pipelining a DPA Countermeasure? In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2004.
40. F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. Cryptology ePrint Archive, Report 2009/341, 2009. `http://eprint.iacr.org/`.
41. K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *Design, Automation and Test in Europe Conference and Exposition – DATE 2004*, pages 246–251. IEEE Computer Society, 2004.
42. Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung. Practical leakage-resilient pseudorandom generators. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security – CCS 2010*, pages 141–151, 2010.

A Our Model in the Physically Observable Cryptography Framework

We discuss hereafter the similarities between our model and the physically observable cryptography framework introduced by Micalli and Reyzin [29].

An elementary calculation in our setting corresponds to the notion of *abstract virtual-memory Turing machine* (VTM) in [29] and what we call the state corresponds the *physical address space* in [29]. Our notion of physical elementary calculation corresponds to the notion of *physical VTM* and it is also defined as a pair composed of an elementary calculation (or VTM) and a leakage function. The notion of leakage function involved in this paper slightly differs from the one defined by Micalli and Reyzin. Whereas in their formalism the leakage function is applied to the current state of the VTM for every step in the VTM computation, the leakage function in our model only applies to the input of the current elementary calculation. This difference does not imply any loss of generality since the elementary calculations we consider process deterministic functions and do not involve any random number generation (except for the leak-free component which does not leak by definition). All the intermediate values within such an elementary calculation are therefore deterministic functions of the input and the overall leakage can hence be modeled as a function of the calculation input (and an independent parameter for the leakage randomness). Another difference with the leakage function defined in [29] is that we omit the second parameter: a binary string encoding the measuring apparatus. The role of this parameter is to enable the adversary to adaptively change the leakage function between each step of the on-going computation. Although we do not need this parameter in our formalism, we emphasize that our security proof holds for any choice of the leakage functions $(f_i)_i$ within pre-defined classes $\left(\mathcal{N}(1/\omega_i)\right)_i$ which encompasses an adaptive choice of the leakage function in $\mathcal{N}(1/\omega_i)$ at every step i.

B Proof of Proposition 1

Proposition 1. *Let X and Y be two random variables, with X uniformly distributed over a set $\mathcal{X}$ of cardinality N. The mutual information between X and Y satisfies* $\mathrm{I}(X;Y) \leq \frac{N}{\ln 2}\beta(X|Y)$.

Proof. Let $\delta_{x,y}$ denote the difference $\mathrm{P}[X = x|Y = y] - \mathrm{P}[X = x]$. By definition we have:

$$\begin{aligned}\mathrm{I}(X;Y) &= \sum_{y\in Y} \mathrm{P}[Y=y] \sum_{x\in\mathcal{X}} (\mathrm{P}[X=x] + \delta_{x,y}) \log_2 \left(\frac{\mathrm{P}[X=x]+\delta_{x,y}}{\mathrm{P}[X=x]}\right) \\ &= \sum_{y\in Y} \mathrm{P}[Y=y] \sum_{x\in\mathcal{X}} \left(\frac{1}{N} + \delta_{x,y}\right) \log_2 \left(1 + N\delta_{x,y}\right) \ .\end{aligned}$$

By definition we have $\delta_{x,y} \geq -\frac{1}{N}$ and the logarithm satisfies $\log_2(1+z) \leq \frac{z}{\ln 2}$ for every $z > -1$. We hence get

$$\mathrm{I}(X;Y) \leq \frac{1}{\ln 2} \sum_{y\in Y} \mathrm{P}[Y=y] \sum_{x\in\mathcal{X}} \left(\frac{1}{N} + \delta_{x,y}\right) N\delta_{x,y} \ .$$

As for every y, we have $\sum_{x\in\mathcal{X}} \delta_{x,y} = 0$, we deduce

$$\mathrm{I}(X;Y) \leq \frac{N}{\ln 2} \sum_{y\in Y} \mathrm{P}[Y=y] \sum_{x\in\mathcal{X}} \delta_{x,y}^2 = \frac{N}{\ln 2} \sum_{y\in Y} \mathrm{P}[Y=y] \, \|\delta_y\|^2 \,,$$

where δ_y denotes the vector $(\delta_{x,y})_{x\in\mathcal{X}}$. It can be checked[7] that the norm of δ_y satisfies $\|\delta_y\| \leq 1 - \frac{1}{N} < 1$, which finally yields

$$\mathrm{I}(X;Y) < \frac{N}{\ln 2} \sum_{y\in Y} \mathrm{P}[Y=y] \, \|\delta_y\| \,,$$

and the proposition follows. □

C Proof of Theorem 1

Theorem 1. *Let X be a uniform random variable over some set $\mathcal{X}$ of cardinality N, let d be a positive integer and let $(X_i)_i$ be a d^{th}-order encoding of X. Let $\varepsilon \in [0,1)$ and let f_0, f_1, …, f_d be noisy functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \big| f_0(X_0), f_1(X_1), \ldots, f_d(X_d)\big) \leq N^{\frac{d}{2}} \varepsilon^{d+1} \,.$$

Proof. According to the properties of a d^{th}-order encoding, the uniformity of X implies the uniformity and the mutual independence of the X_i. Let $(\ell_0, \ell_1, \ldots, \ell_d) \in \mathrm{Im}(f_0) \times \mathrm{Im}(f_1) \times \cdots \times \mathrm{Im}(f_d)$ and let define:

$$p(x) = \mathrm{P}[X = x | (f_0(X_0), f_1(X_1), \ldots, f_d(X_d)) = (\ell_0, \ell_1, \ldots, \ell_d)] \,.$$

We have:

$$p(x) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \mathrm{P}[(X_i)_{i\geq 0} = (x_i)_{i\geq 0} | (f_i(X_i))_{i\geq 0} = (\ell_i)_{i\geq 0}] \,.$$

where $x_0 = x \oplus \bigoplus_{i\geq 1} x_i$. The independence of the X_i then implies

$$p(x) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{i=0}^{d} \mathrm{P}[X_i = x_i | f_i(X_i) = \ell_i] \,.$$

After denoting $\mathrm{P}[X_i = x | f_i(X_i) = \ell_i] - \frac{1}{N}$ by $\delta_x^{(i)}$, we get

$$p(x) \;=\; \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{i=0}^{d} \Big(\frac{1}{N} + \delta_{x_i}^{(i)} \Big) \;=\; \frac{1}{N} + \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{i=0}^{d} \delta_{x_i}^{(i)} \,.$$

[7] By definition, we have $-\frac{1}{N} \leq \delta_{x,y} \leq 1 - \frac{1}{N}$ and $\sum_x \delta_{x,y} = 0$. The norm of δ_y is then lower than or equal to the norm of the vector $\big(-\frac{1}{N}, -\frac{1}{N}, \ldots, -\frac{1}{N}, (N-1)\frac{1}{N}\big)$ which equals $(N-1)\big(-\frac{1}{N}\big)^2 + \big((N-1)\frac{1}{N}\big)^2 = 1 - \frac{1}{N}$.

The latter holds since we have $\sum_{x_{i_1}} \sum_{x_{i_2}} \cdots \sum_{x_{i_t}} \prod_j \delta_{x_{i_j}}^{(i_j)} = \prod_j \left(\sum_{x_{i_j}} \delta_{x_{i_j}}^{(i_j)} \right) = 0$ for every strict subset $\{i_j\} \subset \{0, 1, \ldots, d\}$ (which does not hold for $\{i_j\} = \{0, 1, \ldots, d\}$ as $x_0 = x \oplus \bigoplus_{i \geq 1} x_i$). By definition, we then have

$$\beta\big(X \mid (f_i(X_i))_i = (\ell_i)_i\big)^2 = \sum_x \Big(p(x) - \frac{1}{N}\Big)^2 = \sum_x \Big(\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{i=0}^{d} \delta_{x_i}^{(i)} \Big)^2$$

which implies

$$\beta\big(X \mid (f_i(X_i))_i = (\ell_i)_i\big)^2 \leq \sum_x \Big(\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \Big| \prod_{i=0}^{d} \delta_{x_i}^{(i)} \Big| \Big)^2 \ .$$

Let $\delta^{(i)}$ denote the vector $(\delta_x^{(i)})_{x \in \mathcal{X}}$. By Cauchy-Shwartz we have $\Big| \sum_{x_d} \delta_{c \oplus x_d}^{(0)} \delta_{x_d}^{(d)} \Big| \leq \|\delta^{(0)}\| \, \|\delta^{(d)}\|$ for every $c = x \oplus x_1 \oplus \cdots \oplus x_{d-1}$ (*i.e.* $x_0 = c \oplus x_d$), which implies

$$\begin{aligned} \beta\big(X \mid (f_i(X_i))_i = (\ell_i)_i\big)^2 &\leq \sum_x \Big(\sum_{x_1} \sum_{x_2} \cdots \sum_{x_{d-1}} \Big| \prod_{i=1}^{d-1} \delta_{x_i}^{(i)} \Big| \, \|\delta^{(0)}\| \, \|\delta^{(d)}\| \Big)^2 \\ &= N \, \|\delta^{(0)}\|^2 \, \|\delta^{(d)}\|^2 \Big(\prod_{i=1}^{d-1} \sum_{x_i} |\delta_{x_i}^{(i)}| \Big)^2 \ . \end{aligned}$$

Inequality (1) implies $\sum_{x_i} |\delta_{x_i}^{(i)}| \leq \sqrt{N} \|\delta^{(i)}\|$. We hence deduce

$$\beta\big(X \mid (f_i(X_i))_i = (\ell_i)_i\big)^2 \leq N^d \prod_{i=0}^{d} \|\delta^{(i)}\|^2 \ ,$$

and by definition

$$\beta\big(X \mid (f_i(X_i))_i\big) \leq N^{\frac{d}{2}} \sum_{\ell_0} \sum_{\ell_1} \cdots \sum_{\ell_d} \mathrm{P}[(f_i(X_i))_i = (\ell_i)_i] \prod_{i=0}^{d} \|\delta^{(i)}\| \ .$$

Eventually the independence of the X_i implies the independence of the $f_i(X_i)$ from which we get

$$\begin{aligned} \beta\big(X \mid (f_i(X_i))_i\big) &\leq N^{\frac{d}{2}} \sum_{\ell_0} \sum_{\ell_1} \cdots \sum_{\ell_d} \prod_{i=0}^{d} \mathrm{P}[f_i(X_i) = \ell_i] \, \|\delta^{(i)}\| \\ &= N^{\frac{d}{2}} \prod_{i=0}^{d} \sum_{\ell_i} \mathrm{P}[f_i(X_i) = \ell_i] \, \|\delta^{(i)}\| \ , \end{aligned}$$

that is

$$\beta\big(X \mid (f_i(X_i))_i\big) \leq N^{\frac{d}{2}} \prod_{i=0}^{d} \beta\big(X_i \mid (f_i(X_i))_i\big) \ .$$

The theorem statement directly follows. □

D Proof of Lemma 1

Lemma 1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets. Let f be a noisy function defined over $\mathcal{Y}$ and belonging to the class $\mathcal{N}(\varepsilon)$ and let g be a deterministic function from $\mathcal{X}$ to $\mathcal{Y}$. Then the noisy function $f \circ g$ defined over $\mathcal{X}$ belongs to the class $\mathcal{N}(\sqrt{2}|\mathcal{Y}| \cdot \varepsilon)$.*

Proof. Let X and Y be independent random variables uniformly distributed over $\mathcal{X}$ and $\mathcal{Y}$ respectively. Then, for every $\ell \in \mathrm{Im}(f)$ and every $x \in \mathcal{X}$, we have

$$\mathrm{P}[X = x \mid f \circ g(X) = \ell] = \mathrm{P}[X = x \mid f(Y) = \ell, Y = g(X)] \,.$$

After denoting by $p(x)$ the probability above, the law of total probability implies

$$p(x) = \frac{\mathrm{P}[X = x, Y = g(X) \mid f(Y) = \ell]}{\mathrm{P}[Y = g(X) \mid f(Y) = \ell]} = \frac{\mathrm{P}[X = x, Y = g(x) \mid f(Y) = \ell]}{\sum_{x'} \mathrm{P}[X = x', Y = g(x') \mid f(Y) = \ell]} \,.$$

Then by independence between X and $(Y, f(Y))$, and by uniformity of X, we have

$$p(x) = \frac{\mathrm{P}[X = x]\,\mathrm{P}[Y = g(x) \mid f(Y) = \ell]}{\sum_{x'} \mathrm{P}[X = x']\,\mathrm{P}[Y = g(x') \mid f(Y) = \ell]} = \frac{\mathrm{P}[Y = g(x) \mid f(Y) = \ell]}{\sum_{x'} \mathrm{P}[Y = g(x') \mid f(Y) = \ell]} \,. \quad (4)$$

In the following, we shall denote the cardinalities of the sets $\mathcal{X}$ and $\mathcal{Y}$ by $N = |\mathcal{X}|$ and $M = |\mathcal{Y}|$ respectively. We shall further denote $\delta_y = \mathrm{P}[Y = y \mid f(Y) = \ell] - \frac{1}{M}$ and $\theta = \sum_x \delta_{g(x)}$. Then (4) can be rewritten as

$$p(x) = \frac{\frac{1}{M} + \delta_{g(x)}}{\sum_{x'} \left(\frac{1}{M} + \delta_{g(x')}\right)} = \frac{\frac{1}{M} + \delta_{g(x)}}{\frac{N}{M} + \theta} = \frac{1}{N} + \frac{\delta_{g(x)} - \frac{\theta}{N}}{\frac{N}{M} + \theta} \,.$$

And, by definition of the bias, we get

$$\begin{aligned} \beta(X \mid f \circ g(X) = \ell)^2 = \sum_x \Big(p(x) - \frac{1}{N}\Big)^2 &= \sum_x \Big(\frac{\delta_{g(x)} - \frac{\theta}{N}}{\frac{N}{M} + \theta}\Big)^2 \\ &= \frac{1}{\left(\frac{N}{M} + \theta\right)^2}\Big(\sum_x \delta_{g(x)}^2 - \frac{\theta^2}{N}\Big) \,. \quad (5) \end{aligned}$$

By definition, all the $\delta_{g(x)}$ are greater than or equal to $-\frac{1}{M}$. We are now going to prove that this implies the following inequality:

$$\beta(X \mid f \circ g(X) = \ell)^2 \leq \frac{2M^2}{N} \sum_x \delta_{g(x)}^2 \,. \quad (6)$$

Since we have $\delta_{g(x)}^2 \leq \sum_y \delta_y^2 = \beta(Y \mid f(Y) = \ell)^2$ for every $x \in \mathcal{X}$, (6) implies

$$\beta(X \mid f \circ g(X) = \ell)^2 \leq 2M^2 \beta(Y \mid f(Y) = \ell)^2 \,, \quad (7)$$

from which we deduce the lemma statement.

In order to prove (6), we will consider two possible cases:

$$-\frac{N}{M} < \theta < -\frac{N}{M} + \frac{\sqrt{N/2}}{M} \quad \text{and} \quad -\frac{N}{M} + \frac{\sqrt{N/2}}{M} \leq \theta \,.$$

Let us first consider that we have $\theta \geq -\frac{N}{M} + \frac{\sqrt{N/2}}{M}$. Then (5) directly implies

$$\beta(X \mid f \circ g(X) = \ell)^2 \leq \frac{1}{\left(\frac{N}{M} + \theta\right)^2} \sum_x \delta_{g(x)}^2 \leq \frac{1}{\left(\frac{\sqrt{N/2}}{M}\right)^2} \sum_x \delta_{g(x)}^2 = \frac{2M^2}{N} \sum_x \delta_{g(x)}^2 \ .$$

It now remains to prove that (6) holds when θ satisfies $-\frac{N}{M} < \theta < -\frac{N}{M} + \frac{\sqrt{N/2}}{M}$. Let us denote $\alpha = \theta + \frac{N}{M}$ so that we have $\theta = -\frac{N}{M} + \frac{\alpha}{M}$ with $\alpha \in (0; \sqrt{N/2})$. Let us focus on the sum $\sum_x \delta_{g(x)^2}$. It is minimal when all the $\delta_{g(x)}$ equal the mean value $\frac{\theta}{N}$. On the other hand, since all the $\delta_{g(x)}$ are greater than or equal to $-\frac{1}{M}$, it is maximal when $\delta_{g(x)} = -\frac{1}{M}$ for every x but one, say x_0, and $\delta_{g(x_0)} = -\frac{1}{M} + \frac{\alpha}{M}$. This leads to the following bounding[8] of $\sum_x \delta_{g(x)}^2$:

$$\frac{\theta^2}{N} \leq \sum_x \delta_{g(x)}^2 \leq (N-1)\Big(-\frac{1}{M}\Big)^2 + \Big(-\frac{1}{M} + \frac{\alpha}{M}\Big)^2 \ ,$$

that is

$$\frac{1}{M^2}\Big(N - 2\alpha + \frac{\alpha^2}{N}\Big) \leq \sum_x \delta_{g(x)}^2 \leq \frac{1}{M^2}(N - 2\alpha + \alpha^2) \ , \tag{8}$$

From (5) and the upper bound in (8) we get

$$\beta(X \mid f \circ g(X) = \ell)^2 \leq 1 - \frac{1}{N} \ , \tag{9}$$

and from the lower bound in (8), we get

$$\frac{2M^2}{N} \sum_x \delta_{g(x)}^2 \geq 2\Big(1 - 2\frac{\alpha}{N} + \frac{\alpha^2}{N^2}\Big) = 2\Big(1 - \frac{\alpha}{N}\Big)^2 \geq 2\Big(1 - \frac{1}{\sqrt{2N}}\Big)^2 \geq 1 - \frac{1}{N} \ , \tag{10}$$

where the last holds from $N \geq 2$. Then (9) and (10) directly implies (6). □

E Proof of Theorem 2

Theorem 2. *Let X be a uniform random variable defined over a finite set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0, 1)$ and let $f_1, f_2, \ldots, f_t$ be t noisy functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. For any real number $\alpha \in (0, 1]$, if $\varepsilon \leq \frac{\alpha}{tN}$, then we have*

$$\beta(X|f_1(X), f_2(X), \ldots, f_t(X)) \ \leq \ \left(\Big(\frac{\mathrm{e}^\alpha - 1}{\alpha}\Big)t + \mathrm{e}^\alpha\right)\varepsilon \ .$$

Lemma 3. *Let X be a uniform random variable defined over a finite set $\mathcal{X}$ and let N denote the cardinality of $\mathcal{X}$. Let ε_1 and ε_2 be positive real numbers. And let L_1 and L_2 be two random variables such that $\beta(X|L_i) \leq \varepsilon_i$ for $i = 1, 2$, and $(L_1|X = x)$ and $(L_2|X = x)$ are mutually independent for every $x \in \mathcal{X}$. We have:*

$$\beta(X|L_1, L_2) \leq \varepsilon_1 + \varepsilon_2 + N\varepsilon_1\varepsilon_2 \ .$$

[8] Note that $\delta_{g(x_0)} = -\frac{1}{M} + \frac{\alpha}{M}$ might not be reachable for some values of M, N and ε. In that case the upper bound still hold but it is just less tight.

Proof. Let $(\ell_1, \ell_2) \in \mathcal{L}_1 \times \mathcal{L}_2$ and let $p_{12}(x)$ denote the probability $\mathrm{P}[X = x | L_1 = \ell_1, L_2 = \ell_2]$. By Bayes' and total probability theorems, we have:

$$p_{12}(x) = \frac{\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2 | X = x]\mathrm{P}[X = x]}{\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2]} \ . \tag{11}$$

Then by independence of $(L_1 | X = x)$ and $(L_2 | X = x)$ we have

$$\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2 | X = x] = \mathrm{P}[L_1 = \ell_1 | X = x]\mathrm{P}[L_2 = \ell_2 | X = x] \ ,$$

and Bayes' theorem gives

$$\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2 | X = x] = \frac{\mathrm{P}[X = x | L_1 = \ell_1]\mathrm{P}[X = x | L_2 = \ell_2]\mathrm{P}[L_1 = \ell_1]\mathrm{P}[L_2 = \ell_2]}{\mathrm{P}[X = x]^2} \ . \tag{12}$$

Plugging this equality into (11) yields

$$\begin{aligned} p_{12}(x) &= \frac{\mathrm{P}[X = x | L_1 = \ell_1]\mathrm{P}[X = x | L_2 = \ell_2]\mathrm{P}[L_1 = \ell_1]\mathrm{P}[L_2 = \ell_2]}{\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2]\mathrm{P}[X = x]} \\ &= \theta_{12} \, \frac{\mathrm{P}[X = x | L_1 = \ell_1]\mathrm{P}[X = x | L_2 = \ell_2]}{\mathrm{P}[X = x]} \end{aligned}$$

where

$$\theta_{12} = \frac{\mathrm{P}[L_1 = \ell_1]\mathrm{P}[L_2 = \ell_2]}{\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2]} \ .$$

Let us denote the difference $\mathrm{P}[X = x | f_i(X) = \ell_i] - \mathrm{P}[X = x]$ by $\delta_x^{(i)}$ for $i = 1, 2$. We can then rewrite the above equality as follows:

$$p_{12}(x) \ = \ \theta_{12}\Big(\frac{1}{N} + \delta_x^{(1)}\Big)\Big(\frac{1}{N} + \delta_x^{(2)}\Big)N \ = \ \frac{\theta_{12}}{N}(1 + N\delta_x^{(1)})(1 + N\delta_x^{(2)}) \ ,$$

Let us now denote the difference $p_{12}(x) - \mathrm{P}[X = x]$ by $\delta_x^{(12)}$. We get:

$$\delta_x^{(12)} = \frac{\theta_{12}}{N}(1 + N\delta_x^{(1)})(1 + N\delta_x^{(2)}) - \frac{1}{N} = \frac{\theta_{12}}{N}\Big((1 + N\delta_x^{(1)})(1 + N\delta_x^{(2)}) - \frac{1}{\theta_{12}}\Big) \ . \tag{13}$$

Now from the law of total probability we have

$$\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2] = \sum_{x \in \mathcal{X}} \mathrm{P}[X = x]\mathrm{P}[L_1 = \ell_1, L_2 = \ell_2 | X = x] \ ,$$

and from (12), we can write

$$\frac{1}{\theta_{12}} = \sum_{x \in \mathcal{X}} \frac{\mathrm{P}[X = x | L_1 = \ell_1]\mathrm{P}[X = x | L_2 = \ell_2]}{\mathrm{P}[X = x]} = N \sum_{x \in \mathcal{X}} \Big(\frac{1}{N} + \delta_x^{(1)}\Big)\Big(\frac{1}{N} + \delta_x^{(2)}\Big) \ .$$

Since by definition $\sum_x \delta_x^{(i)} = 0$ with $i \in \{1, 2\}$, we get

$$\frac{1}{\theta_{12}} = \Big(1 + N \sum_{x \in \mathcal{X}} \delta_x^{(1)}\delta_x^{(2)}\Big) \ .$$

Equation (13) can then be rewritten as

$$\delta_x^{(12)} = \theta_{12}\Big(\delta_x^{(1)} + \delta_x^{(2)} + N\delta_x^{(1)}\delta_x^{(2)} - \sum_{x\in\mathcal{X}} \delta_x^{(1)}\delta_x^{(2)}\Big) \,.$$

In the rest of the proof, we will use the following notations:

$$e_1 := \|\delta^{(1)}\| \,, \quad e_2 := \|\delta^{(2)}\| \,, \quad e_{12} := \|\delta^{(12)}\| \,, \text{ and } \quad \rho_{12} := \sum_{x\in\mathcal{X}} \delta_x^{(1)}\delta_x^{(2)} \,,$$

where $\delta^{(i)}$ denotes the vector $(\delta_x^{(i)})_{x\in\mathcal{X}}$. Our goal is then to upper-bound e_{12}^2. By definition, we have

$$e_{12}^2 = \theta_{12}^2 \sum_{x\in\mathcal{X}} \big(\delta_x^{(1)} + \delta_x^{(2)} + N\delta_x^{(1)}\delta_x^{(2)} - \rho_{12}\big)^2 \,.$$

Then developing the $\big(\delta_x^{(1)} + \delta_x^{(2)} + N\delta_x^{(1)}\delta_x^{(2)} - \rho_{12}\big)^2$, one obtains the following terms:

$$\begin{gathered}{\delta_x^{(1)}}^2 \;+\; {\delta_x^{(2)}}^2 \;+\; N^2{\delta_x^{(1)}}^2{\delta_x^{(2)}}^2 \;+\; \rho_{12}^2 \;+\; 2\delta_x^{(1)}\delta_x^{(2)} \;+\; 2N{\delta_x^{(1)}}^2\delta_x^{(2)} \;+\; 2N\delta_x^{(1)}{\delta_x^{(2)}}^2 \\ -\; 2\rho_{12}\delta_x^{(1)} \;-\; 2\rho_{12}\delta_x^{(2)} \;-\; 2N\rho_{12}\delta_x^{(1)}\delta_x^{(2)} \,,\end{gathered}$$

and summing over $\mathcal{X}$ one gets

$$\begin{gathered}e_1^2 \;+\; e_2^2 \;+\; N^2\textstyle\sum_x {\delta_x^{(1)}}^2{\delta_x^{(2)}}^2 \;+\; N\rho_{12}^2 \;+\; 2\rho_{12} \;+\; 2N\textstyle\sum_x {\delta_x^{(1)}}^2\delta_x^{(2)} \;+\; 2N\textstyle\sum_x \delta_x^{(1)}{\delta_x^{(2)}}^2 \\ -\; 0 \;-\; 0 \;-\; 2N\rho_{12}^2 \,.\end{gathered}$$

Then, denoting $e_1' := \big\|\big({\delta_x^{(1)}}^2\big)_x\big\|$, $e_2' := \big\|\big({\delta_x^{(2)}}^2\big)_x\big\|$, and applying Causchy-Shwartz, we get

$$\rho_{12} \le e_1e_2 \,, \quad \sum_x {\delta_x^{(1)}}^2\delta_x^{(2)} \le e_1'e_2 \,, \quad \sum_x \delta_x^{(1)}{\delta_x^{(2)}}^2 \le e_1e_2' \,, \text{ and } \quad \sum_x {\delta_x^{(1)}}^2{\delta_x^{(2)}}^2 \le e_1'e_2' \,,$$

which gives

$$e_{12}^2 = \theta_{12}^2\Big(e_1^2 + e_2^2 + N^2e_1'e_2' - N\rho_{12}^2e_1^2e_2^2 + 2\rho_{12}e_1e_2 + 2Ne_1'e_2 + 2Ne_1e_2'\Big) \,.$$

And from $e_i'^2 = \sum_x {\delta_x^{(i)}}^4 \le \big(\sum_x {\delta_x^{(i)}}^2\big)^2 = e_i^4$, we get

$$e_{12}^2 \;\le\; \theta_{12}^2\Big(e_1^2 + e_2^2 + N^2e_1^2e_2^2 + 2e_1e_2 + 2Ne_1^2e_2 + 2Ne_1e_2^2\Big) \,.$$

A simple development then shows

$$e_{12}^2 \;\le\; \theta_{12}^2(e_1 + e_2 + Ne_1e_2)^2 \,,$$

that is

$$e_{12} \;\le\; \theta_{12}(e_1 + e_2 + Ne_1e_2) \,,$$

We eventually obtain

$$\beta(X|L_1,L_2) \;=\; \sum_{(\ell_1,\ell_2)} \mathrm{P}[L_1=\ell_1,L_2=\ell_2]\; e_{12}\;,$$

and by definition of θ_{12}, we get

$$\begin{aligned}\beta(X|L_1,L_2) \;\leq\; & \sum_{(\ell_1,\ell_2)} \mathrm{P}[L_1=\ell_1]\,\mathrm{P}[L_2=\ell_2](e_1+e_2+Ne_1e_2)\\ & = \beta(X|L_1)+\beta(X|L_2)+N\beta(X|L_1)\beta(X|L_2)\;.\end{aligned}$$

And Lemma 3 directly follows. □

Proof. (Theorem 2) Let ε_i denote the bias $\beta(X|L_1,L_2,\ldots,L_i)$. From $\beta(X|L_{i+1}) \leq \varepsilon$ and by Lemma 3 we have:

$$\varepsilon_{i+1} = \beta\big(X|(L_1,L_2,\ldots,L_i),L_{i+1}\big) \;\leq\; \varepsilon_i+\varepsilon+N\varepsilon_i\varepsilon\;.$$

Then from $\varepsilon \leq \frac{\alpha}{tN}$ we get

$$\varepsilon_{i+1} \;\leq\; \Big(1+\frac{\alpha}{t}\Big)\varepsilon_i+\varepsilon\;,$$

which defines a sequence with recurrence relation of the form $\varepsilon_{i+1} \leq a\,\varepsilon_i + b$. Such a sequence satisfies $\varepsilon_{i+1} \leq a^i\varepsilon_1 + \big(\frac{a^i-1}{a-1}\big)b$. Here we have $a = 1+\frac{\alpha}{t}$ and $b=\varepsilon_1=\varepsilon$, therefore we get

$$\beta(X|L_1,L_2,\ldots,L_t)=\varepsilon_t \;\leq\; \left(\Big(1+\frac{\alpha}{t}\Big)^{t-1}\Big(1+\frac{t}{\alpha}\Big)-\frac{t}{\alpha}\right)\varepsilon\;.$$

Since for every positive integer t we have $\big(1+\frac{\alpha}{t}\big)^{t-1} < \mathrm{e}^{\alpha}$, the above inequality directly implies Theorem 2. □

F Proof of Theorem 3

Theorem 3. *Let A and B be two random variables uniformly distributed over some finite set $\mathcal{X}$ of cardinality N. Let d be a positive integer, and let $(A_i)_i$ and $(B_j)_j$ be two d^{th}-order encodings of A and B respectively. Let ε be a real number such that $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0,1]$ and let $(f_{i,j})_{i,j}$ be noisy functions defined over $\mathcal{X}\times\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big((A,B)|(f_{i,j}(A_i,B_j))_{i,j}\big) \leq 2N^{\frac{3d+2}{2}}\big((\lambda_1 d+\lambda_0)\varepsilon\big)^{d+1}\;,$$

where $\lambda_1 = \frac{\mathrm{e}^{\alpha}-1}{\alpha}$ and $\lambda_0 = \lambda_1+\mathrm{e}^{\alpha}$.

Lemma 4. *Let A and B be two uniform and mutually independent random variables defined over a finite set $\mathcal{X}$ of cardinality N. Let f be a noisy function and let $L=f(A,B)$, for every $a,b\in\mathcal{X}$, and for every $\ell \in \mathrm{Im}(f)$, we have*

$$\beta(A|f(A,b)=\ell) \leq N\beta((A,B)|L=\ell)\;,$$

and (by symmetry),

$$\beta(B|f(a,B)=\ell) \leq N\beta((A,B)|L=\ell)\;.$$

Proof. We prove the bound for $\beta(A|f(A,b))$ only since the bound for $\beta(B|f(a,B))$ then directly holds by symmetry. For every $a, b \in \mathcal{X}$ and $\ell \in \mathcal{L}$, we denote:

$$\delta_{(a,b),\ell} \;:=\; \mathrm{P}[(A,B) = (a,b) \mid L = \ell] - \frac{1}{N^2}$$

and

$$\delta'_{a|b,\ell} \;:=\; \mathrm{P}[A = a \mid B = b, L = \ell] - \frac{1}{N} \,.$$

From $\mathrm{P}[A = a \mid B = b, L = \ell] = \mathrm{P}[A = a, B = b \mid L = \ell]\mathrm{P}[B = b]^{-1}$ we deduce $\delta'_{a|b,\ell} = N\delta_{(a,b),\ell}$. The independence of A and B implies

$$\mathrm{P}[A = a \mid B = b, L = \ell] = \mathrm{P}[A = a \mid B = b, f(A,b) = \ell] = \mathrm{P}[A = a \mid f(A,b) = \ell] \,.$$

By definition of the bias, we deduce

$$\beta(A \mid f(A,b) = \ell)^2 = \sum\nolimits_a \delta'^2_{a|b,\ell} = \sum\nolimits_a N^2 \delta^2_{(a,b),\ell} \,,$$

which implies

$$\beta(A \mid f(A,b) = \ell)^2 \leq \sum\nolimits_{a,b} N^2 \delta^2_{(a,b),\ell} = N^2 \beta((A,B) \mid L = \ell)^2 \,,$$

and directly yields Lemma 4 . □

Proof. (Theorem 3) Let $\boldsymbol{L}$ denote the random vector $(f_{i,j}(A_i, B_j))_{0 \leq i,j \leq d}$. For every $a, b \in \mathcal{X}$ and $\boldsymbol{\ell} \in \mathcal{L}$, we denote:

$$\begin{aligned}
\delta_{a,\boldsymbol{\ell}} \;&:=\; \mathrm{P}[A = a \mid \boldsymbol{L} = \boldsymbol{\ell}] - \tfrac{1}{N} \,, \\
\delta_{b|a,\boldsymbol{\ell}} \;&:=\; \mathrm{P}[B = b \mid A = a, \boldsymbol{L} = \boldsymbol{\ell}] - \tfrac{1}{N} \,, \\
\delta_{(a,b),\boldsymbol{\ell}} \;&:=\; \mathrm{P}[(A,B) = (a,b) \mid \boldsymbol{L} = \boldsymbol{\ell}] - \tfrac{1}{N^2} \,.
\end{aligned}$$

By definition of the conditional probability, we have

$$\delta_{(a,b),\boldsymbol{\ell}} = \mathrm{P}[A = a \mid \boldsymbol{L} = \boldsymbol{\ell}]\, \mathrm{P}[B = b \mid A = a, \boldsymbol{L} = \boldsymbol{\ell}] - \frac{1}{N^2} \;\;,$$

which implies

$$\delta_{(a,b),\boldsymbol{\ell}} = \Big(\frac{1}{N} + \delta_{a,\boldsymbol{\ell}}\Big)\Big(\frac{1}{N} + \delta_{b|a,\boldsymbol{\ell}}\Big) - \frac{1}{N^2} = \frac{1}{N}\delta_{a,\boldsymbol{\ell}} + \frac{1}{N}\delta_{b|a,\boldsymbol{\ell}} + \delta_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}} \,,$$

and thus

$$\delta^2_{(a,b),\boldsymbol{\ell}} = \frac{1}{N^2}\delta^2_{a,\boldsymbol{\ell}} + \frac{1}{N^2}\delta^2_{b|a,\boldsymbol{\ell}} + \delta^2_{a,\boldsymbol{\ell}}\delta^2_{b|a,\boldsymbol{\ell}} + \frac{2}{N^2}\delta_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}} + \frac{2}{N}\delta^2_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}} + \frac{2}{N}\delta_{a,\boldsymbol{\ell}}\delta^2_{b|a,\boldsymbol{\ell}} \;\;. \quad (14)$$

Let us now develop the sums of the different terms over $(a,b) \in \mathcal{X} \times \mathcal{X}$. By definition of the bias, we have

$$\sum_{a,b} \delta^2_{a,\boldsymbol{\ell}} = N\beta(A \mid \boldsymbol{L} = \boldsymbol{\ell})^2 \,, \quad (15)$$

and

$$\sum_{a,b} \delta_{b|a,\boldsymbol{\ell}}^2 = N\beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell})^2 \,. \tag{16}$$

Then, since the $\delta_{b|a,\boldsymbol{\ell}}$ sum to zero over $b \in \mathcal{X}$, we get

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}} = \sum_{a} \delta_{a,\boldsymbol{\ell}} \sum_{b} \delta_{b|a,\boldsymbol{\ell}} = 0 \;\;, \tag{17}$$

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}^2\delta_{b|a,\boldsymbol{\ell}} = \sum_{a} \delta_{a,\boldsymbol{\ell}}^2 \sum_{b} \delta_{b|a,\boldsymbol{\ell}} = 0 \;\;. \tag{18}$$

On the other hand, we have

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}^2\delta_{b|a,\boldsymbol{\ell}}^2 = \sum_{a} \delta_{a,\boldsymbol{\ell}}^2 \sum_{b} \delta_{b|a,\boldsymbol{\ell}}^2 = \sum_{a} \delta_{a,\boldsymbol{\ell}}^2\beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell})^2 \,,$$

giving

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}^2\delta_{b|a,\boldsymbol{\ell}}^2 \leq \beta(A \mid \boldsymbol{L}=\boldsymbol{\ell})^2\big(\max_a \beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell})^2\big) \,. \tag{19}$$

Eventually, for the last term we obtain

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}}^2 = \sum_{a} \delta_{a,\boldsymbol{\ell}} \sum_{b} \delta_{b|a,\boldsymbol{\ell}}^2 = \sum_{a} \delta_{a,\boldsymbol{\ell}}\beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell})^2 \,,$$

which implies

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}}^2 \leq \big(\max_a \beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell})^2\big) \sum_{a,\delta_{a,\boldsymbol{\ell}}>0} \delta_{a,\boldsymbol{\ell}} \,.$$

Since the $\delta_{a,\boldsymbol{\ell}}$ sum to zero over $a \in \mathcal{X}$, the sum $\sum_{a,\delta_{a,\boldsymbol{\ell}}>0} \delta_{a,\boldsymbol{\ell}}$ equals $\frac{1}{2}\sum_a |\delta_{a,\boldsymbol{\ell}}|$. Together with (1), this leads to

$$\sum_{a,b} \delta_{a,\boldsymbol{\ell}}\delta_{b|a,\boldsymbol{\ell}}^2 \leq \frac{\sqrt{N}}{2}\beta(A \mid \boldsymbol{L}=\boldsymbol{\ell})\big(\max_a \beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell})^2\big) \,. \tag{20}$$

Eventually (14) and (15)–(20) imply

$$\beta((A,B) \mid \boldsymbol{L}=\boldsymbol{\ell})^2 = \sum \delta_{(a,b),\boldsymbol{\ell}}^2 \leq \frac{1}{N}\beta_1(\boldsymbol{\ell})^2 + \frac{1}{N}\beta_2(\boldsymbol{\ell})^2 + \beta_1(\boldsymbol{\ell})^2\beta_2(\boldsymbol{\ell})^2 + \frac{1}{\sqrt{N}}\beta_1(\boldsymbol{\ell})\beta_2(\boldsymbol{\ell})^2 \,,$$

where

$$\beta_1(\boldsymbol{\ell}) = \beta(A \mid \boldsymbol{L}=\boldsymbol{\ell}) \quad \text{and} \quad \beta_2(\boldsymbol{\ell}) = \max_a \beta(B \mid A=a, \boldsymbol{L}=\boldsymbol{\ell}) \,.$$

By definition of the bias, we have $\beta_2(\boldsymbol{\ell}) \leq 1 - \frac{1}{N}$ for every $\boldsymbol{\ell}$, and in particular $\beta_1(\boldsymbol{\ell})^2\beta_2(\boldsymbol{\ell})^2 \leq \big(1-\frac{1}{N}\big)\beta_1(\boldsymbol{\ell})^2$, and $\frac{1}{\sqrt{N}}\beta_1(\boldsymbol{\ell})\beta_2(\boldsymbol{\ell})^2 \leq \frac{1}{\sqrt{N}}\beta_1(\boldsymbol{\ell})\beta_2(\boldsymbol{\ell})$. A simple development then shows

$$\beta((A,B) \mid \boldsymbol{L}=\boldsymbol{\ell}) \leq \beta_1(\boldsymbol{\ell}) + \frac{1}{\sqrt{N}}\beta_2(\boldsymbol{\ell}) \leq \beta_1(\boldsymbol{\ell}) + \beta_2(\boldsymbol{\ell}) \,.$$

We finally get

$$\beta((A,B) \mid \boldsymbol{L}) \leq \sum_{\boldsymbol{\ell}} \mathrm{P}[\boldsymbol{L}=\boldsymbol{\ell}]\beta_1(\boldsymbol{\ell}) + \sum_{\boldsymbol{\ell}} \mathrm{P}[\boldsymbol{L}=\boldsymbol{\ell}]\beta_2(\boldsymbol{\ell}) \; . \tag{21}$$

We now show how to upper-bound the expected values of $\beta_1(\boldsymbol{\ell})$ and $\beta_2(\boldsymbol{\ell})$.

By definition of the $f_{i,j}$ we have $\beta(A_i \mid f_{i,j}(A_i,B_j)) \leq \varepsilon$. Moreover, the uniformity and the independence of A and B implies the uniformity and the mutual independence of the A_i and the B_j. For every $i \in \{0,1,\ldots,d\}$, Lemma 4 then yields

$$\max_{b_j} \beta(A_i \mid f_{i,j}(A_i,b_j)) \leq N\varepsilon \; . \tag{22}$$

Let us denote by $\boldsymbol{b} = (b_0,b_1,\ldots,b_d)$ some vector over $\mathcal{X}^{d+1}$ and by $\boldsymbol{B}$ the corresponding random variable. We moreover denote by $F_i^{\boldsymbol{b}}$ the noisy function $F_i^{\boldsymbol{b}} : A_i \mapsto (f_{i,d}(A_i,b_j))_{0\leq j\leq d}$. Theorem 2 (with $t=d+1$) and (22) imply for every $i \in \{0,1,\ldots,d\}$:

$$\max_{\boldsymbol{b}} \beta(A_i \mid F_i^{\boldsymbol{b}}(A_i)) \leq (\lambda_1 d + \lambda_0) N\varepsilon \; , \tag{23}$$

where $\lambda_1 = \frac{\mathrm{e}^\alpha - 1}{\alpha}$ and $\lambda_0 = \lambda_1 + \mathrm{e}^\alpha$. In other words, for every $i \in \{0,1,\ldots,d\}$ and every $\boldsymbol{b} \in \mathcal{X}^{d+1}$, the noisy function $F_i^{\boldsymbol{b}}$ belongs to $\mathcal{N}((\lambda_1 d + \lambda_0)N\varepsilon)$.

On the other hand, the law of total probability implies

$$\begin{aligned}
\mathrm{P}[A=a \mid \boldsymbol{L}=\boldsymbol{\ell}] &= \sum_{\boldsymbol{b}} \mathrm{P}[A=a, \boldsymbol{B}=\boldsymbol{b} \mid \boldsymbol{L}=\boldsymbol{\ell}] \\
&= \sum_{\boldsymbol{b}} \mathrm{P}[\boldsymbol{B}=\boldsymbol{b} \mid \boldsymbol{L}=\boldsymbol{\ell}] \, \mathrm{P}[A=a \mid \boldsymbol{B}=\boldsymbol{b}, \boldsymbol{L}=\boldsymbol{\ell}]
\end{aligned}$$

that is

$$\mathrm{P}[A=a \mid \boldsymbol{L}=\boldsymbol{\ell}] = \sum_{\boldsymbol{b}} \mathrm{P}[\boldsymbol{B}=\boldsymbol{b} \mid \boldsymbol{L}=\boldsymbol{\ell}] \, \mathrm{P}[A=a \mid (f_{i,j}(A_i,b_j))_{i,j}=\boldsymbol{\ell}] \; . \tag{24}$$

The law of total probability further implies

$$\begin{aligned}
&\mathrm{P}[A=a \mid (f_{i,j}(A_i,b_j))_{i,j}=\boldsymbol{\ell}] \\
&\quad = \sum_{a_1}\sum_{a_2}\cdots\sum_{a_d} \mathrm{P}[A=a, A_1=a_1,\ldots,A_d=a_d \mid (f_{i,j}(A_i,b_j))_{i,j}=\boldsymbol{\ell}] \\
&\quad = \sum_{a_1}\sum_{a_2}\cdots\sum_{a_d} \mathrm{P}[A_0=a_0, A_1=a_1,\ldots,A_d=a_d \mid (f_{i,j}(A_i,b_j))_{i,j}=\boldsymbol{\ell}] \; ,
\end{aligned}$$

where $a_0 = a \oplus \bigoplus_{i=1}^d a_i$. Then by mutual independence of the A_i, we get

$$\mathrm{P}[A=a \mid (f_{i,j}(A_i,b_j))_{i,j}=\boldsymbol{\ell}] = \sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\prod_{i=0}^{d} \mathrm{P}[A_i=a_i \mid F_i^{\boldsymbol{b}}(A_i)=(\ell_{i,j})_j] \; .$$

Let us now denote the difference $\mathrm{P}[A_i = a_i \mid F_i^{\boldsymbol{b}}(A_i) = (\ell_{i,j})_j] - \frac{1}{N}$ by[9] $\delta_{a_i}^{(i)}$ for every $i \in \{0, 1, \ldots, d\}$. We get

$$\begin{aligned}\mathrm{P}[A = a \mid (f_{i,j}(A_i, b_j))_{i,j} = \boldsymbol{\ell}] &= \sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\prod_{i=0}^{d}\Big(\frac{1}{N} + \delta_{a_i}^{(i)}\Big) \\ &= \sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\Big(\frac{1}{N^{d+1}} + \prod_{i=0}^{d}\delta_{a_i}^{(i)}\Big)\,,\end{aligned}$$

where the second equality holds since we have

$$\sum_{a_{i_1}}\sum_{a_{i_2}}\cdots\sum_{a_{i_t}}\prod_{k=1}^{t}\delta_{a_{i_k}}^{(i_k)} = \prod_{k=1}^{t}\sum_{a_{i_k}}\delta_{a_{i_k}}^{(i_k)} = 0$$

for every strict subset $\{i_k\} \subset \{0, 1, \ldots, d\}$ (which does not hold for $\{i_k\} = \{0, 1, \ldots, d\}$ as $a_0 = a \oplus \bigoplus_{i\geq 1} a_i$). Therefore (24) can be rewritten as

$$\mathrm{P}[A = a \mid \boldsymbol{L} = \boldsymbol{\ell}] = \sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\Big(\frac{1}{N^{d+1}} + \prod_{i=0}^{d}\delta_{a_i}^{(i)}\Big)\,.$$

By definition of $\delta_{a,\boldsymbol{\ell}}$ the equation above becomes

$$\begin{aligned}\delta_{a,\boldsymbol{\ell}} &= \sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\Big(\sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\Big(\frac{1}{N^{d+1}} + \prod_{i=0}^{d}\delta_{a_i}^{(i)}\Big) - \frac{1}{N}\Big) \\ &= \sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\prod_{i=0}^{d}\delta_{a_i}^{(i)}\,.\end{aligned}$$

The bias of A given $\boldsymbol{L} = \boldsymbol{\ell}$, denoted $\beta_1(\boldsymbol{\ell})$, hence satisfies

$$\begin{aligned}\beta_1(\boldsymbol{\ell})^2 &= \sum_{a}\Big(\sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\sum_{a_1}\sum_{a_2}\cdots\sum_{a_d}\prod_{i=0}^{d}\delta_{a_i}^{(i)}\Big)^2 \\ &= \sum_{a}\Big(\sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\sum_{a_1}\sum_{a_2}\cdots\sum_{a_{d-1}}\prod_{i=1}^{d-1}\delta_{a_i}^{(i)}\sum_{a_d}\delta_{a_0}^{(0)}\delta_{a_d}^{(d)}\Big)^2\,.\end{aligned}$$

Let $\delta^{(i)}$ denote the vector composed of the $\delta_{a_i}^{(i)}$ for every $i \in \{0, 1, \ldots, d\}$. By Cauchy-Shwartz we have $\Big|\sum_{a_d}\delta_{a_0}^{(0)}\delta_{a_d}^{(d)}\Big| \leq \|\delta^{(0)}\|\,\|\delta^{(d)}\|$ (note that a_0 satisfies $a_0 = \big(a \oplus \bigoplus_{i=1}^{d-1} a_i\big) \oplus a_d$). It follows that the bias of A given $\boldsymbol{L} = \boldsymbol{\ell}$ satisfies

$$\begin{aligned}\beta_1(\boldsymbol{\ell})^2 &\leq \sum_{a}\Big(\sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\,\|\delta^{(0)}\|\,\|\delta^{(d)}\|\sum_{a_1}\sum_{a_2}\cdots\sum_{a_{d-1}}\Big|\prod_{i=1}^{d-1}\delta_{a_i}^{(i)}\Big|\Big)^2 \\ &\leq \sum_{a}\Big(\sum_{\boldsymbol{b}}\mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}]\,\|\delta^{(0)}\|\,\|\delta^{(d)}\|\prod_{i=1}^{d-1}\sum_{a_i}\big|\delta_{a_i}^{(i)}\big|\Big)^2\,.\end{aligned}$$

[9] Although $\delta_{a_i}^{(i)}$ depends on $\boldsymbol{b}$ and $(\ell_{i,j})_j$, we do not make them appear in the notation for the sake of clarity.

Then by (1), we have $\sum_{a_i} \left|\delta_{a_i}^{(i)}\right| \leq \sqrt{N}\|\delta^{(i)}\|$, giving

$$\begin{aligned}\beta_1(\boldsymbol{\ell})^2 &\leq \sum_a \Big(\sum_{\boldsymbol{b}} \mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}] N^{\frac{d-1}{2}} \prod_{i=0}^{d} \|\delta^{(i)}\| \Big)^2 \\ &\leq N^d \Big(\sum_{\boldsymbol{b}} \mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}] \prod_{i=0}^{d} \|\delta^{(i)}\| \Big)^2\end{aligned}$$

and we get

$$\beta_1(\boldsymbol{\ell}) \leq N^{\frac{d}{2}} \sum_{\boldsymbol{b}} \mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}] \prod_{i=0}^{d} \|\delta^{(i)}\| \,.$$

By definition we have $\|\delta^{(i)}\| = \beta(A_i \mid F_i^{\boldsymbol{b}}(A_i) = (\ell_{i,j})_j)$ for every $i \in \{0, 1, \ldots, d\}$, which gives

$$\begin{aligned}\beta_1(\boldsymbol{\ell}) &\leq N^{\frac{d}{2}} \sum_{\boldsymbol{b}} \mathrm{P}[\boldsymbol{B} = \boldsymbol{b} \mid \boldsymbol{L} = \boldsymbol{\ell}] \prod_{i=0}^{d} \beta(A_i \mid F_i^{\boldsymbol{b}}(A_i) = (\ell_{i,j})_j) \\ &\leq N^{\frac{d}{2}} \prod_{i=0}^{d} \big(\max_{\boldsymbol{b}} \beta(A_i \mid F_i^{\boldsymbol{b}}(A_i) = (\ell_{i,j})_j) \,.\end{aligned}$$

Then by computing the expectation over $\boldsymbol{\ell} \in \prod_{i,j} \mathrm{Im}(f_{i,j})$, and by applying (23), we get

$$\sum_{\boldsymbol{\ell}} \mathrm{P}[\boldsymbol{L} = \boldsymbol{\ell}] \beta_1(\boldsymbol{\ell}) \leq N^{\frac{d}{2}} \prod_{i=0}^{d} \big(\max_{\boldsymbol{b}} \beta(A_i \mid F_i^{\boldsymbol{b}}(A_i)) \leq N^{\frac{3d+2}{2}} \big((\lambda_1 d + \lambda_0)\varepsilon \big)^{d+1} \,. \tag{25}$$

Using exactly the same approach as above, it can be shown that the same holds for $\beta_2(\boldsymbol{\ell})$, namely

$$\sum_{\boldsymbol{\ell}} \mathrm{P}[\boldsymbol{L} = \boldsymbol{\ell}] \beta_2(\boldsymbol{\ell}) \leq N^{\frac{3d+2}{2}} \big((\lambda_1 d + \lambda_0)\varepsilon \big)^{d+1} \,. \tag{26}$$

The only difference is that the probability of B is taken given $A = a$ additionnaly to $\boldsymbol{L} = \ell$. But this difference does not impact the previous approach since we bound the maximum bias of B given all the possible events $A_i = a_i$. The bounds (25) and (26) together with (21) finally proove the theorem statement. □

G Proof of Lemma 2

Lemma 2. *Let $T_0, T_1, \ldots, T_d$ be $d+1$ independent random variables uniformly distributed over some set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0, 1)$ and let $f_1, f_2, \ldots, f_d$ be noisy functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(T_d \big| f_1(T_0, T_1), f_2(T_1, T_2), \ldots, f_d(T_{d-1}, T_d)\big) \leq N\varepsilon \,.$$

Proof. We denote by $F(T_{d-1}, T_d)$ the d-tuple $\big(f_1(T_0, T_1), f_2(T_1, T_2), \ldots, f_d(T_{d-1}, T_d)\big)$. Note that since the T_i are mutually independent, F can be viewed as a noisy function applied to (T_{d-1}, T_d) and where T_0, T_1, …, T_{d-2} are part of the internal randomness of F (though such representation is not mandatory for the present proof). By Lemma 4 (see Appendix F), for every $t \in \mathcal{X}$ and for every $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_d) \in \mathrm{Im}(F)$, we have

$$\beta\big(T_d \mid F(T_{d-1}, T_d) = \boldsymbol{\ell}\big) \;\leq\; N\beta\big(T_d \mid F(t, T_d) = \boldsymbol{\ell}\big) \;=\; N\beta\big(T_d \mid f_d(t, T_d) = \ell_d\big) \;,$$

where the right equality holds from the mutual independence of the T_i. Then the proof holds from $f_d \in \mathcal{N}(\varepsilon)$. □

H Proof of Lemma 2

Lemma 2. *Let T_0, T_1, …, T_d be $d+1$ independent random variables uniformly distributed over some set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0, 1)$ and let f_1, f_2, …, f_d be noisy functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(T_d \big| f_1(T_0, T_1), f_2(T_1, T_2), \ldots, f_d(T_{d-1}, T_d)\big) \leq 2N\varepsilon \;.$$

Proof. We denote by $F(T_d)$ the d-tuple $\big(f_1(T_0, T_1), f_2(T_1, T_2), \ldots, f_d(T_{d-1}, T_d)\big)$. Note that since the T_i are independent of T_d, F can be viewed as a noisy function applied to T_d and where T_0, T_1, …, T_{d-1} are part of the internal randomness of F (though such representation is not mandatory for the present proof). For every $t \in \mathcal{X}$ and for every $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_d) \in \mathrm{Im}(F)$, the total probability law implies

$$\mathrm{P}[T_d = t | F(T_d) = \boldsymbol{\ell}] = \sum_{t' \in \mathcal{X}} \mathrm{P}[T_d = t | T_{d-1} = t', F(T_d) = \boldsymbol{\ell}] \mathrm{P}[T_{d-1} = t' | F(T_d) = \boldsymbol{\ell}] \;. \tag{27}$$

Moreover, by mutual independence of the T_i, we have

$$\mathrm{P}[T_d = t | T_{d-1} = t', F(T_d) = \boldsymbol{\ell}] = \mathrm{P}[T_d = t | f_d(t', T_d) = \ell_d] \;\; .$$

For every $t \in \mathcal{X}$, we denote

$$\delta_t^- = \min_{t'} \mathrm{P}[T_d = t | f_d(t', T_d) = \ell_d] - \frac{1}{N} \;,$$

and

$$\delta_t^+ = \max_{t'} \mathrm{P}[T_d = t | f_d(t', T_d) = \ell_d] - \frac{1}{N} \;\; .$$

For every t', we then have

$$\frac{1}{N} + \delta_t^- \leq \mathrm{P}[T_d = t | f_d(t', T_d) = \ell_d] \leq \frac{1}{N} + \delta_t^+ \;\; ,$$

and (27) implies

$$\frac{1}{N} + \delta_t^- \leq \mathrm{P}[T_d = t | F(T_d) = \boldsymbol{\ell}] \leq \frac{1}{N} + \delta_t^+ \;.$$

We deduce

$$\beta(T_d \mid F(T_d) = \boldsymbol{\ell})^2 = \sum_t \Big(\mathrm{P}[T_d = t | F(T_d) = \boldsymbol{\ell}] - \frac{1}{N}\Big)^2 \leq \sum_t {\delta_t^-}^2 + \sum_t {\delta_t^+}^2$$

giving

$$\beta(T_d \mid F(T_d) = \boldsymbol{\ell}) \;\leq\; \sqrt{\|\delta^-\|^2 + \|\delta^+\|^2} \;\leq\; \|\delta^-\| + \|\delta^+\| \tag{28}$$

where δ^+ and δ^- denotes the vectors $(\delta_t^+)_{t\in\mathcal{X}}$ and $(\delta_t^-)_{t\in\mathcal{X}}$ respectively. Then by Lemma 4 (see Appendix F), for every $t' \in \mathcal{X}$ and every $\ell_d \in \mathrm{Im}(f_d)$ we have

$$\beta\big(T_d \mid f_d(t', T_d) = \ell_d\big) \;\leq\; N\beta\big((T_{d-1}, T_d) \mid f_d(T_{d-1}, T_d) = \ell_d\big) \;,$$

from which together with (28) implies

$$\beta(T_d \mid F(T_d) = \boldsymbol{\ell}) \;\leq\; 2N\beta\big((T_{d-1}, T_d) \mid f_d(T_{d-1}, T_d) = \ell_d\big) \;.$$

□

Appendix D

How to Securely Compute with Noisy Leakage in Quasilinear Complexity

Hereafter is appended the full version of our paper [GJR18], joint work with Dahmun Goudarzi and Antoine Joux, published at **ASIACRYPT 2018**.

How to Securely Compute with Noisy Leakage in Quasilinear Complexity

Dahmun Goudarzi[1,2], Antoine Joux[3], and Matthieu Rivain[1]

[1] CryptoExperts, Paris, France
[2] ENS, CNRS, INRIA and PSL Research University, Paris, France
[3] Chaire de Cryptologie de la Fondation de l'UPMC
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, Paris, France

{dahmun.goudarzi,matthieu.rivain}@cryptoexperts.com
antoine.joux@m4x.org

Abstract. Since their introduction in the late 90's, side-channel attacks have been considered as a major threat against cryptographic implementations. This threat has raised the need for formal leakage models in which the security of implementations can be proved. At Eurocrypt 2013, Prouff and Rivain introduced the *noisy leakage model* which has been argued to soundly capture the physical reality of power and electromagnetic leakages. In their work, they also provide the first formal security proof for a masking scheme in the noisy leakage model. However their work has two important limitations: (i) the security proof relies on the existence of a leak-free component, (ii) the tolerated amount of information in the leakage (aka *leakage rate*) is of $O(1/n)$ where n is the number of shares in the underlying masking scheme. The first limitation was nicely tackled by Duc, Dziembowski and Faust one year later (Eurocrypt 2014). Their main contribution was to show a security reduction from the noisy leakage model to the conceptually simpler *random-probing model*. They were then able to prove the security of the well-known Ishai-Sahai-Wagner scheme (Crypto 2003) in the noisy leakage model. The second limitation was addressed last year in a paper by Andrychowicz, Dziembowski and Faust (Eurocrypt 2016). The proposed construction achieves security in the strong adaptive probing model with a leakage rate of $O(1/\log n)$ at the cost of a $O(n^2 \log n)$ complexity. The authors argue that their result can be translated into the noisy leakage model with a leakage rate of $O(1)$ by using secret sharing based on algebraic geometric codes. They further argue that the efficiency of their construction can be improved by a linear factor using packed secret sharing but no details are provided.

In this paper, we show how to compute in the presence of noisy leakage with a leakage rate up to $\tilde{O}(1)$ in complexity $\tilde{O}(n)$. We use a polynomial encoding allowing quasilinear multiplication based on the fast Number Theoretic Transform (NTT). We first show that our scheme is secure in the random-probing model with leakage rate $O(1/\log n)$. Using the reduction by Duc *et al.* this result can be translated in the noisy leakage model with a $O(1/|\mathbb{F}|^2 \log n)$ leakage rate. However, as in the work of Andrychowicz *et al.* , our construction also requires $|\mathbb{F}| = O(n)$. In order to bypass this issue, we refine the granularity of our computation by considering the noisy leakage model on *logical instructions* that work on constant-size machine words. We provide a generic security reduction from the noisy leakage model at the logical-instruction level to the random-probing model at the arithmetic level. This reduction allows us to prove the security of our construction in the noisy leakage model with leakage rate $\tilde{O}(1)$.

1 Introduction

Side-channel attacks have been considered as a major threat against cryptographic implementations since their apparition in the late 90's. It was indeed shown that even a tiny dependence between the data processed by a device and its *side-channel leakage* (*e.g.* running time, power consumption, electromagnetic emanation) could allow devastating key-recovery attack against the implementation of any cryptosystem secure in the standard model [Koc96,KJJ99,GMO01]. The so-called physical security of cryptographic implementations has then become a very active research area and many efficient countermeasures have been proposed to mitigate these attacks. However, most of these countermeasures are only empirically validated or they are proven secure in a weak adversarial model where, for instance, an attacker only exploits a small part of the available leakage.

An important step towards a more formal treatment of side-channel security was made by Micalli and Reyzin in 2004 in their *physically observable cryptography* framework [MR04]. In particular, they formalized the assumptions that a cryptographic device can at least keep some secrets and that *only computation leaks information*. This framework was then specialized into the *leakage resilient cryptography* model introduced by Dziembowski and Pietrzak in [DP08] which gave rise to a huge amount of subsequent works. In this model, a leaking computation is divided into elementary operations that are assumed to leak some information about their inputs through a leakage function whose range is bounded (*i.e.* taking values in $\{0,1\}^\lambda$ for some parameter λ). Many new leakage-resilient cryptographic primitives were proposed as well as so-called *compilers* that can make any computation secure in this model [GR12].

While the leakage resilient literature has achieved considerable theoretical advances, the considered model does not fully capture the physical reality of power or electromagnetic leakages (see for instance [SPY+09]). In particular for a leakage function $f : \{0,1\}^n \to \{0,1\}^\lambda$, the parameter λ must be (significantly) smaller than n. This means, for instance, that the leakage of an AES computation should be smaller than 128 bits, whereas in practice an AES power trace can take several kilobytes (or even megabytes). On the other hand, it is fair to assume that the side-channel leakage is *noisy* in such a way that the information $f(x)$ leaked by an elementary operation on a variable x is not enough to fully recover x. This intuition was formalized in the *noisy leakage model* introduced by Prouff and Rivain in 2013 [PR13]. In a nutshell, this model considers that an elementary operation with some input x leaks a *noisy leakage function* $f(x)$. The noisy feature is then captured by assuming that an observation of $f(x)$ only implies a bounded bias in the probability distribution of x. Namely the statistical distance between the distributions $\Pr(x)$ and $\Pr(x|f(x))$ is bounded by some parameter δ. In particular, this model does not imply any restriction on the leakage size but only on the amount of useful information it contains.

1.1 Related Works

Probing-secure circuits. In a seminal paper of 2003, Ishai, Sahai and Wagner considered the problem of building Boolean circuits secure against *probing attacks* [ISW03]. In the so-called *probing model*, an adversary is allowed to adaptively probe up to t wires of the circuit. They show how to transform any circuit C with q logic gates into a circuit C' with $O(qt^2)$ logic gates that is secure against a t-probing adversary. Their scheme consists in encoding each Boolean variable x as a random sharing $(x_1, x_2, \ldots, x_n)$ satisfying $x_1 + x_2 + \ldots + x_n = x$ over $\mathbb{F}_2$, where $n = 2t + 1$. They show how to transform each logic gate into a *gadget* that work on encoded variables. Their construction is actually secure against an adversary that can adaptively place up to t probes per such gadget. The so-called ISW construction has since then served as a building block in many practical side-channel countermeasures known as *higher-order masking schemes* (see for instance [RP10,CPRR14,CRV14]).

Towards noisy-leakage security. In [PR13], Prouff and Rivain proposed the first formal security proof for an ISW-like masking scheme in the noisy leakage model. In particular they generalize the previous work of Chari *et al.* [CJRR99] and show that in the presence of noisy leakage on the shares x_1, x_2, ..., x_n the information on x becomes negligible as n grows. Specifically, they show that for any δ-noisy leakage function f, the mutual information between x and the leakage $(f(x_1), f(x_2), \ldots, f(x_n))$ is of order $O(\delta^n)$. They also provide a security proof for full masked computation in the noisy leakage model, however their result has two important limitations. First they assume the existence of a *leak-free component* that can refresh a sharing without leaking any information. Second, their proof can only tolerate an δ-noisy leakage with $\delta = O(1/n)$. Namely, the amount of leakage must decrease linearly with the number of shares. Note that this second limitation is inherent to masking schemes based on the ISW construction since it implies that each share leaks $O(n)$ times. Some practical attacks have recently been exhibited that exploit this issue [BCPZ16].

Avoiding leak-free components. In [DDF14], Duc, Dziembowski and Faust tackled the first of these two limitations. Namely they show how to avoid the requirement for a leak-free component with a nice and conceptually simpler security proof. Applying the Chernoff bound, they show that the ISW scheme is secure in the δ-*random probing model* in which each operation leaks its full input with a given probability $\delta = O(1/n)$ (and leaks nothing with probability $1 - \delta$). Their main contribution is then to show that any δ'-noisy leakage $f(x)$ can be simulated from a δ-random probing leakage $\phi(x)$ with $\delta' \leq \delta \cdot |\mathcal{X}|$, where

$\mathcal{X}$ denotes the definition space of x. In other words, if the δ-random probing leakage of a computation contains no significant information, then neither does any δ'-noisy leakage of this computation as long as $\delta \leq \delta' \cdot |\mathcal{X}|$. The ISW scheme is therefore secure against δ'-noisy leakage for $\delta' = O(1/n|\mathcal{X}|)$. Note that for an arithmetic program working over some field $\mathbb{F}$, each elementary operation takes up to two inputs on $\mathbb{F}$, meaning $\mathcal{X} = \mathbb{F}^2$ and $\delta' = O(1/n|\mathbb{F}|^2)$. This way, the work of Duc *et al.* avoid the strong requirement of leak-free components. However, it still requires a leakage rate of $O(1/n)$.

Towards a constant leakage rate. This second limitation was addressed by Andrychowicz, Dziembowski, and Faust [ADF16]. They describe a construction using Shamir's secret sharing [Sha79] and a refreshing algorithm from expander graphs with constant degree due to Ajtai [Ajt11]. The number of instructions in the protected program is multiplied by a factor $O(n^3)$ which can be reduced to $O(n^2 \log n)$ using the FFT. They show that this construction achieves security in the strong probing model where an adversary can adaptively place up to $O(1/\log n)$ probes per elementary operation. In the random probing model, the result is improved to a constant ratio. Applying the reduction from [DDF14] they obtain the security against δ-noisy leakage with a leakage rate $\delta = O(1/|\mathbb{F}|^2)$. (Note that they obtain a leakage rate $O(1/|\mathbb{F}|)$ in the restrictive model where input variables leak independently. In the present paper, we make the more realistic assumption that the leakage function applies to the full input of each elementary operation.) For the standard version of their scheme based on Shamir's secret sharing, the base field $\mathbb{F}$ must be of size $O(n)$ which implies a leakage rate $\delta = O(1/n^2)$ in the noisy leakage model. Fortunately, their scheme can be improved by using secret sharing based on algebraic geometric codes [CC06] (at the cost of weaker parameters). As argued in [ADF16], these codes operate over fields of constant size and hence there basic operations can be implemented by constant size Boolean circuits, which gives a $\delta = O(1)$ noisy leakage rate with the DDF reduction. As explained hereafter, we propose in this paper a generic reduction to achieve $\delta = \tilde{O}(1)$ noisy leakage rate from a random-probing secure scheme on a field $\mathbb{F} = O(n)$. This reduction could also be used to get noisy-leakage security for the ADF scheme with Shamir's secret sharing.

Towards a quasilinear complexity. Another challenging issue is to improve the efficiency of leakage-secure schemes, and in particular to bridge the gap between the current $\tilde{O}(n^2)$ complexity and the theoretically achievable $\tilde{O}(n)$ complexity. In [ADF16], the authors argue that the complexity of their scheme can be improved by using *packed secret sharing* [ADD$^+$15,DIK10]. As explained in [ADD$^+$15], the use of packed secret sharing allows to securely compute an addition or a multiplication on ℓ values in parallel at the price of what a single operation would cost with a standard secret sharing. Using the construction from [DIK10], one can improve the complexity of the ADF scheme on a circuit of size s and of width ℓ from $O(s\,n^2 \log n)$ to $O(s \log s\, n^2 \log n/\ell)$. For a circuit of width $\ell = \Theta(n)$, this approach hence yields a secure circuit with quasilinear overhead in n. For a constant-size circuit (as the AES cipher) on the other hand, only a constant factor can be saved and the complexity remains of $\tilde{O}(n^2)$. Let us also mention that recent works have proposed efficiency improvements of the ISW construction [BBP$^+$16,BBP$^+$17]. These works provide new multiplication schemes with a linear randomness consumption or a linear number of field multiplications but the overall complexity is still quadratic.

1.2 Our Contribution

In this paper we show how to securely compute in the noisy leakage model with a leakage rate of $\tilde{O}(1)$ and with complexity overhead of $\tilde{O}(n)$ (whatever the circuit size). Our scheme is conceptually very simple and also practically efficient provided that the computation relies on a base field $\mathbb{F}$ with appropriate structure.

Specifically, we consider an *arithmetic program* P that executes basic arithmetic instructions over some prime field $\mathbb{F}$ (additions, subtractions, and multiplications) satisfying $|\mathbb{F}| = \alpha \cdot n + 1$ for n being a power of 2 (in particular $|\mathbb{F}| = O(n)$ as in [ADF16]).[4] In our scheme, each element $a \in \mathbb{F}$ is encoded into a random tuple $(a_0, a_1, \ldots, a_{n-1})$ that satisfies the relation $a = \sum_{i=0}^{n-1} a_i \omega^i$ for some random element $\omega \in \mathbb{F}$. In other words, a is encoded as the coefficient of a random n-degree polynomial Q satisfying

[4] We prefer the terminology of (arithmetic) program composed of instructions to the terminology of (arithmetic) circuit composed of gates but the two notions are strictly equivalent.

$Q(\omega) = a$. It is worth noting that the security of our scheme does not rely on the secrecy of ω but on its random distribution. We then show how to transform each arithmetic instruction of P into a corresponding secure gadget that works on encoded variables. Using a fast Number Theoretic Transform (NTT), we then achieve a multiplication gadget with $O(n \log n)$ instructions.

In a first place, we show that our scheme is secure in the δ-random-probing model for a parameter $\delta = O(1/\log n)$. Specifically, we show that for any program P with a constant number of instructions $|P|$, the advantage of a δ-random-probing adversary can be upper bounded by $\mathsf{negl}(\lambda) + \mathsf{negl}'(n)$ where negl and negl' are some negligible functions and where λ denotes some security parameter that impact the size of $\mathbb{F}$ (specifically we have $\lambda = \log \alpha$ where $|\mathbb{F}| = \alpha \cdot n + 1$). This is shown at the level of a single NTT-based secure multiplication in a first place. Then we show how to achieve compositional security, by interleaving each gadget by a refreshing procedure that has some *input-output separability* property. Using the Chernoff bound as in [DDF14] we can then statistically bound the number of leaking intermediate variables in each gadget. Specifically, we show that the leakage in each gadget can be expressed as linear or quadratic combinations of the input shares that do not reveal any information with overwhelming probability (over the randomness of ω).

From our result in the random probing model, the security reduction of Duc *et al.* [DDF14] directly implies that our construction is secure in the δ'-noisy leakage model for $\delta' = O(1/|\mathbb{F}|^2 \log n)$. However, since we require $|\mathbb{F}| = O(n)$ (as in the standard ADF scheme) this reduction is not satisfactory. We then refine the granularity of our computation by considering the noisy leakage model on *logical instructions* working on constant-size machine words. In this model, we provide a generic reduction from the random-probing model over $\mathbb{F}$ to the noisy leakage model on logical instructions. Namely we show that any arithmetic program Π secure under a δ-random-probing leakage gives rise to a functionally equivalent program Π' that is secure under a δ'-noisy leakage at the logical instruction level where $\delta' = \delta / O(\log |\mathbb{F}| \log \log |\mathbb{F}|)$. Applying this reduction, our construction achieves security in the δ'-noisy leakage model with $\delta' = O\big(1/((\log n)^2 \log \log n)\big)$ for a computational overhead of $O(n \log n)$.

The paper is organized as follows. Section 2 provides background notions on the noisy leakage model and the considered adversary. In Section 3 we describe our secure quasilinear multiplication scheme and we prove its security in the random probing model. Section 4 then presents the refreshing procedure used to get compositional security and provides a security proof for a full arithmetic program. In Section 5 we give our generic reduction from the random-probing model over $\mathbb{F}$ to the noisy leakage model on logical instructions and we apply this reduction to our scheme to get our final result. We finally discuss practical aspects of our scheme and related open problems in Section 6.

2 Leakage and Adversary

In the rest of the paper, we shall denote by $x \leftarrow \mathcal{X}$ the action of picking x uniformly at random over some set $\mathcal{X}$. Similarly, for a probabilistic algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ the action of running $\mathcal{A}$ on input x with a fresh random tape and setting y to the obtained result.

2.1 Noisy Leakage Model

The noisy leakage model introduced by Prouff and Rivain in [PR13] follows the *only computation leaks* paradigm [MR04]. In this paradigm, the computation is divided into subcomputations; each works on a subpart x of the current computation state and leaks some information $f(x)$, where f is called the *leakage function*. In practice, f is a so-called *randomized function* that takes two arguments, the input variable x and a random tape ρ that is large enough to model the leakage noise. A subcomputation with input variable x hence leaks $f(x, \rho)$ for a fresh random tape ρ. For the sake of simplicity, in the sequel we shall omit the parameter ρ and see $f(x)$ as a random realization of $f(x, \rho)$. Moreover, the definition space of the input x shall be called the *domain* of f, and we shall write $f : \mathcal{X} \to \mathcal{Y}$ for a randomized function with domain $\mathcal{X}$ and image space $\mathcal{Y}$.

In the noisy leakage model [PR13], a *noisy leakage function* f is defined as a leakage function such that an observation $f(x)$ only implies a bounded bias in the probability distribution of x. Namely, the statistical distance between the distributions of x and $(x \mid f(x))$ is assumed to be bounded by some bias ε. Let X and X' be two random variables defined over some set $\mathcal{X}$. We recall that the *statistical distance*

between X and X' is defined as:

$$\Delta(X;X') = \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr(X=x) - \Pr(X'=x)| \ . \tag{1}$$

The notion of noisy leakage function is then formalized as follows:

Definition 1 ([PR13]). *A* ε-noisy leakage function *is a randomized function* $f : \mathcal{X} \to \mathcal{Y}$ *satisfying*

$$\sum_{y\in\mathcal{Y}} \Pr(f(X)=y)\cdot\Delta(X;(X \mid f(X)=y)) \leq \varepsilon \ , \tag{2}$$

where X *is a uniform random variable over* $\mathcal{X}$*.*

In practice, the leaking input x might not be uniformly distributed but one must specify a distribution to have a consistent definition, and as argued in [PR13], the uniform distribution is a natural choice. Also note that in the original paper [PR13], the L_2 norm was used for the definition of the statistical distance while, as argued in [DDF14], the L_1 norm is a more standard choice (that we also adopt in this paper).

A conceptually simpler model, known as the *random probing model*, was first used in [ISW03] and formalized in the work of Duc, Dziembowski, and Faust [DDF14]. Informally speaking, this model restricts the noisy leakage model to leakage functions that leak their entire input with a given probability. These *random-probing leakage functions* are formalized in the following definition.[5]

Definition 2. *A* ε-random-probing leakage function *is a randomized function* $\phi : \mathcal{X} \to \mathcal{X}\cup\{\perp\}$ *satisfying*

$$\phi(x) = \begin{cases} \perp & \text{with probability } 1-\varepsilon \\ x & \text{with probability } \varepsilon \end{cases} \tag{3}$$

It can be checked that such a function is a special case of ε-noisy leakage function.[6] Moreover, it has been shown by Duc, Dziembowski, and Faust [DDF14] that every noisy leakage function f can be expressed as a composition $f = f' \circ \phi$ where ϕ is a random-probing leakage function. This important result enables to reduce noisy-leakage security to random-probing security. It is recalled hereafter:

Lemma 1 ([DDF14]). *Let* $f : \mathcal{X} \to \mathcal{Y}$ *be a* ε*-noisy leakage function with* $\varepsilon < \frac{1}{|\mathcal{X}|}$*. There exists a* δ*-random-probing leakage function* $\phi : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ *and a randomized function* $f' : \mathcal{X} \cup \{\perp\} \to \mathcal{Y}$ *such that for every* $x \in \mathcal{X}$ *we have*

$$f(x) = f'(\phi(x)) \quad \text{and} \quad \delta \leq \varepsilon \cdot |\mathcal{X}| \ . \tag{4}$$

In the random-probing model, the total number of leaking operations can be statistically bounded using the Chernoff bound as suggested in [ISW03,DDF14]. We shall follow this approach in the present paper by using the following corollary.

Corollary 1 (Chernoff bound [Che52]). *The* δ*-random probing leakage of a computation composed of* N *elementary operations reveals the input of* $\ell > \delta N$ *of these elementary operations with probability lower than*

$$\psi(\ell, N) = \exp\Big(-\frac{(\ell - \delta\, N)^2}{\ell + \delta\, N}\Big) \tag{5}$$

If $\ell \leq \alpha n$ *and* $N = \beta n$*, for some* α*,* β *and* n *with* $\alpha/\beta > \delta$ *, the above gives*

$$\psi(\alpha n, \beta n) = \exp\Big(-\frac{(\alpha - \delta\beta)^2}{\alpha + \delta\beta}\ n\Big) \tag{6}$$

[5] Note that we use a different terminology from [DDF14] where these are called ε*-identity functions*.

[6] To be tighter: a ε-random-probing leakage function is a $\varepsilon\,(1 - \frac{1}{|\mathcal{X}|})$-noisy function. This can be simply checked by evaluation (2).

2.2 Leakage Adversary

We consider secure computation schemes that *encode* the data of a program in order to make the leakage on the encoded data useless. An *encoding* Enc is a randomized function that maps an element $x \in \mathbb{F}$ to a n-tuple $\mathsf{Enc}(x) \in \mathbb{F}^n$, where n is called the encoding length, and for which a deterministic function $\mathsf{Dec} : \mathbb{F}^n \to \mathbb{F}$ exists that satisfies $\Pr(\mathsf{Dec}(\mathsf{Enc}(x)) = x) = 1$ for every $x \in \mathbb{F}$ (where the latter probability is taken over the encoding randomness).

Consider an *arithmetic program* P taking a string $\boldsymbol{x} \in \mathbb{F}^s$ as input and executing a sequence of instructions of the form $\mathtt{m}_i \leftarrow \mathtt{m}_j * \mathtt{m}_k$, where $*$ denotes some operations over $\mathbb{F}$ (addition, subtraction, or multiplication) and where $[\mathtt{m}_0, \mathtt{m}_1, \ldots, \mathtt{m}_T]$ denotes the memory of the program which is initialized with $\boldsymbol{x}$ (and some constants). To achieve leakage security, the program P is transformed into a functionally equivalent arithmetic program Π taking as input an encoded input $\mathsf{Enc}(\boldsymbol{x})$ (where the encoding simply applies to each coordinate of $\boldsymbol{x}$). According to the defined leakage model, each executed instruction of Π is then assumed to leak some noisy function $f(\mathtt{m}_j, \mathtt{m}_k)$ of its pair of inputs. It is further assumed that Π includes random sampling instructions $\mathtt{m}_i \leftarrow \mathbb{F}$ that each leaks a noisy function of its output $f(\mathtt{m}_i)$. We denote the overall leakage by $\mathcal{L}(\Pi, \boldsymbol{x})$. The compiler is then said to be *leakage secure* if an observation of $\mathcal{L}(\Pi, \boldsymbol{x})$ does not reveal significant information about $\boldsymbol{x}$. More specifically, the leakage $\mathcal{L}(\Pi, \boldsymbol{x})$ must be indistinguishable from the leakage $\mathcal{L}(\Pi, \boldsymbol{x}')$ for every $\boldsymbol{x}' \in \mathbb{F}^s$. This security notion is formalized as follows:

Definition 3 (Leakage Security). *The program Π is* ε-leakage secure *if every adversary $\mathcal{A}$ has advantage at most ε of distinguishing $\mathcal{L}(\Pi, \boldsymbol{x}_0)$ from $\mathcal{L}(\Pi, \boldsymbol{x}_1)$ for chosen $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, i.e. we have:*

$$\mathrm{Adv}_{\mathcal{A}}^{\Pi} := \left| \mathrm{Succ}_{\mathcal{A}}^{\Pi} - \frac{1}{2} \right| \leq \varepsilon \tag{7}$$

where

$$\mathrm{Succ}_{\mathcal{A}}^{\Pi} = \Pr \left(\begin{array}{c} (\boldsymbol{x}_0, \boldsymbol{x}_1, \mu) \leftarrow \mathcal{A}(\perp) \\ b \leftarrow \{0,1\} \\ \boldsymbol{\ell} \leftarrow \mathcal{L}(\Pi, \boldsymbol{x}_b) \end{array} : \mathcal{A}(\boldsymbol{x}_0, \boldsymbol{x}_1, \mu, \boldsymbol{\ell}) = b \right) . \tag{8}$$

In the above definition, $\mu \in \{0,1\}^*$ denotes any auxiliary information computed by the adversary during the first round when she chooses the inputs $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. Note that for the definition to be sound, we only consider adversaries $\mathcal{A}$ such that $\mathcal{A}(\perp)$ takes values over $\mathbb{F}^s \times \mathbb{F}^s \times \{0,1\}^*$ and $\mathcal{A}(\boldsymbol{x}_0, \boldsymbol{x}_1, \mu, \boldsymbol{\ell})$ takes values over $\{0,1\}$ for every input $(\boldsymbol{x}_0, \boldsymbol{x}_1, \mu, \boldsymbol{\ell}) \in \mathbb{F}^s \times \mathbb{F}^s \times \{0,1\}^* \times \mathrm{Im}(\mathcal{L})$.

Lemma 1 provides a security reduction from the noisy leakage model to the random probing model. This is formalized in the following corollary:

Corollary 2. *Let Π be an arithmetic program that is ε-leakage secure wrt δ-random-probing leakage functions. Then Π is ε-leakage secure wrt δ'-noisy leakage functions, where $\delta' = \delta \cdot |\mathbb{F}|^2$.*

Note that in the original version of Lemma 1 (see [DDF14]), the authors need the additional requirement that f' is efficiently *decidable* so that $f'(\phi(x))$ is computable in polynomial time in $|\mathcal{X}|$. We ignore this property in the present paper since our security statements consider adversaries with unlimited computational power.

3 Secure Multiplication in Quasilinear Complexity

In this section, we describe our encoding scheme and the associated secure multiplication. An important requirement of our construction is that the size n of the underlying encoding must divide $\frac{p-1}{2}$ where p is the characteristic of $\mathbb{F}$, that is $\mathbb{F}$ must contain the $2n$-th roots of unity. This implies that the size of the elements of $\mathbb{F}$ is in $\Omega(\log n)$. Without loss of generality, we further assume that n is a power of 2.

3.1 Our Encoding

Let ξ denote a primitive $2n$th root of unity in $\mathbb{F}$. Our encoding is based on a random element $\omega \in \mathbb{F}^*$ and is defined as follows:

Definition 4. *Let $\omega \in \mathbb{F}^*$ and $a \in \mathbb{F}$. An* ω-encoding *of a is a tuple $(a_i)_{i=0}^{n-1} \in \mathbb{F}^n$ satisfying $\sum_{i=0}^{n-1} a_i \omega^i = a$.*

Our encoding function Enc maps an element $a \in \mathbb{F}$ to a random element $\omega \in \mathbb{F}^*$ and a random uniform ω-encoding of a:

$$\mathsf{Enc}(a) = \langle \omega, (a_0, a_1, \ldots, a_{n-1}) \rangle \ . \tag{9}$$

The corresponding decoding function Dec is defined as:

$$\mathsf{Dec}\big(\langle \omega, (a_0, a_1, \ldots, a_{n-1}) \rangle\big) := \mathsf{Dec}_\omega(a_0, a_1, \ldots, a_{n-1}) := \sum_{i=0}^{n-1} a_i \omega^i \tag{10}$$

It is easy to check that we have $\Pr(\mathsf{Dec}(\mathsf{Enc}(a)) = a) = 1$ for every $a \in \mathbb{F}$. It is worth noting that the security of our scheme does not rely on the secrecy of ω but on its uniformity. Besides, we will consider that ω is systematically leaked to the adversary.

3.2 Multiplication of Encoded Variables

Let $(a_i)_{i=0}^{n-1}$ be an ω-encoding of a and $(b_i)_{i=0}^{n-1}$ be an ω-encoding of b. To compute an ω-encoding $(c_i)_{i=0}^{n-1}$ of $c = a \cdot b$ we use the NTT-based polynomial multiplication.

Specifically, we first apply the NTT on $(a_i)_i$ and $(b_i)_i$ to obtain the polynomial evaluations $u_j = \sum_{i=0}^{n-1} a_i (\xi^j)^i$ and $v_j = \sum_{i=0}^{n-1} b_i (\xi^j)^i$ for $j \in [\![0, 2n-1]\!]$. These evaluations are then pairwisely multiplied to get evaluations of the product $s_j = (2n)^{-1} u_j \cdot v_j$ for $j \in [\![0, 2n-1]\!]$ (with a multiplicative factor $(2n)^{-1}$). Afterwards, we apply the inverse NTT to get coefficients t_i that satisfy $\sum_{i=0}^{2n-1} t_i \omega^i = (\sum_{i=0}^{n-1} a_i \omega^i) \cdot (\sum_{i=0}^{n-1} (b_i \omega^i))$. Eventually, we apply a compression procedure to recover an n-size ω-encoding from the $2n$-size ω-encoding $(t_i)_i$. Due to the particular form of roots of unity, an NTT can be evaluated with a divide and conquer strategy in $3\, n \log n$ arithmetic instructions (a detailed description is given in Appendix A).

The overall process is summarized as follows:

$$\begin{aligned}
(u_0, u_1, \ldots, u_{2n-1}) &\leftarrow \mathsf{NTT}_\xi(a_0, a_1, \ldots, a_{n-1}, 0, \ldots, 0) \\
(r_0, r_1, \ldots, r_{2n-1}) &\leftarrow \mathsf{NTT}_\xi(b_0, b_1, \ldots, b_{n-1}, 0, \ldots, 0) \\
(s_0, s_1, \ldots, s_{2n-1}) &\leftarrow (2n)^{-1}(u_0 \cdot r_0, u_1 \cdot r_1, \ldots, u_{2n-1} \cdot r_{2n-1})\big) \\
(t_0, t_1, \ldots, t_{2n-1}) &\leftarrow \mathsf{NTT}_{\xi^{-1}}(s_0, s_1, \ldots, s_{2n-1}) \\
(c_0, c_1, \ldots, c_{n-1}) &\leftarrow \mathsf{compress}(t_0, t_1, \ldots, t_{2n-1})
\end{aligned}$$

Compression procedure. After computing the inverse NTT, we get a double-size encoding $(t_i)_{i=0}^{2n-1}$ satisfying $\sum_{i=0}^{2n-1} t_i\, \omega^i = a \cdot b$, for some ω (randomly picked by the refresh call in Step 3). In order to obtain a standard encoding with n shares, we simply set $c_i = t_i + t_{n+i}\omega^n$ for $i \in [\![0, n-1]\!]$. It is not hard to see that the result is consistent.

3.3 Security in the Random Probing Model

We first focus on the NTT leakage security as it is the most complex part of our scheme, and then provide a security proof for the whole multiplication.

Security of the NTT. We have the following result:

Theorem 1. *Let ω be a uniform random element of $\mathbb{F}^*$, let $(a_i)_{i=0}^{n-1}$ be a uniform ω-encoding of some variable a and let $\delta < 1/(6 \log n)$. The NTT_ξ procedure on input $(a_i)_{i=0}^{n-1}$ is ε-leakage secure in the δ-random-probing leakage model, where*

$$\varepsilon = \frac{n}{|\mathbb{F}|} + \exp\Big(- \frac{(1 - 6\delta \log n)^2}{4}\ n \Big) \ . \tag{11}$$

The rest of the section gives a proof of Theorem 1. During the computation of the NTT on an ω-encoding $(a_i)_{i=0}^{n-1}$ of a, all the leaking intermediate variables (*i.e.* the inputs of arithmetic instructions) are linear combinations of the a_i's. Specifically, every intermediate variable v occurring in the NTT computation can be expressed as $v = \sum_{i=0}^{n-1} \alpha_i a_i$ where the α_i's are constant coefficients over $\mathbb{F}$. In the following, we shall use the notation $[v] = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ for the vector of coefficients of such an intermediate variable. Similarly, we shall denote $[a] = (1, \omega, \omega^2, \ldots, \omega^{n-1})$ since we have $a = \sum_{i=0}^{n-1} \omega^i a_i$ by definition. Moreover, we will denote by $[v_0, v_1, \ldots, v_\ell]$ the matrix with row vectors $[v_0]$, $[v_1]$, $\ldots$, $[v_\ell]$. In particular, we have $[a_0, a_1, \ldots, a_{n-1}] = I_n$ (where I_n stands for the identity matrix of dimension n over $\mathbb{F}$) and for $u_i = \sum_{j=0}^{n-1} a_j (\xi^i)^j$ (the output elements of the NTT), the matrix $[u_0, u_1, \ldots, u_{n-1}]$ is a Vandermonde matrix.

First consider an adversary that recovers $\ell < n$ intermediate variables in the computation of the NTT, denoted $v_1, v_2, \ldots, v_\ell$. Without loss of generality, we assume that these intermediate variables are linearly independent (otherwise the adversary equivalently gets less than ℓ intermediate variables), which means that the matrix $[v_1, v_2, \ldots, v_\ell]$ has full rank. The following lemma gives a necessary and sufficient condition for such a leakage to be statistically independent of a.

Lemma 2. *Let $v_1, v_2, \ldots, v_\ell$ be a set of $\ell < n$ intermediate variables of the NTT on input a uniform ω-encoding of a variable a. The distribution of the tuple $(v_1, v_2, \ldots, v_\ell)$ is statistically independent of a iff*

$$[a] \notin \mathrm{Im}([v_1, \ldots, v_\ell]) \ . \tag{12}$$

Proof. If $[a] \in \mathrm{Im}([v_1, \ldots, v_\ell])$ then there exists constants $\gamma_1, \gamma_2, \ldots, \gamma_\ell$ such that $[a] = \sum_i \gamma_i [v_i]$ implying $a = \sum_i \gamma_i v_i$, and the distribution $(v_1, v_2, \ldots, v_\ell)$ is hence statistically dependent on a. On the other hand, if $[a] \notin \mathrm{Im}([v_1, \ldots, v_\ell])$, then the system

$$\begin{cases} a = \sum_{j=0}^{n-1} \omega^j a_j & = \gamma_0 \\ v_1 = \sum_{j=0}^{n-1} \alpha_{1,j} a_j & = \gamma_1 \\ v_2 = \sum_{j=0}^{n-1} \alpha_{2,j} a_j & = \gamma_2 \\ \vdots & \\ v_\ell = \sum_{j=0}^{n-1} \alpha_{t,j} a_j & = \gamma_\ell \end{cases}$$

has $|\mathbb{F}|^{n-(\ell+1)}$ solutions $(a_0, a_1, \ldots, a_{n-1})$ for every $(\gamma_0, \gamma_1, \ldots, \gamma_\ell) \in \mathbb{F}^{\ell+1}$. This implies the statistical independence between a and $(v_1, v_2, \ldots, v_\ell)$. □

The following lemma gives an upper bound on the probability that the above condition is not fulfilled.

Lemma 3. *Let ω be a uniform random element in $\mathbb{F}^*$ and let $v_1, v_2, \ldots, v_\ell$ be a set of $\ell < n$ linearly independent intermediate variables of the NTT on input an ω-encoding of a variable a. We have:*

$$\Pr\left[[a] \in \mathrm{Im}([v_1, \ldots, v_\ell])\right] \leq \frac{\ell}{|\mathbb{F}| - 1} < \frac{n}{|\mathbb{F}|} \ , \tag{13}$$

where the above probability is taken over a uniform random choice of ω.

Proof. Let us denote $A(x) = \sum_{i=0}^{n-1} a_i x^i$ so that $[A(\alpha)] = (1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$ for every $\alpha \in \mathbb{F}$, and in particular $[a] = [A(\omega)]$. For any distinct $\ell+1$ elements $\alpha_1, \alpha_2, \cdots \alpha_{\ell+1} \in \mathbb{F}^*$, the matrix $[A(\alpha_1), A(\alpha_2), \ldots, A(\alpha_{\ell+1})]$ has full rank since it is a Vandermonde matrix with distinct input entries. This directly implies:

$$\underbrace{\mathrm{Im}([A(\alpha_1), A(\alpha_2), \ldots, A(\alpha_{\ell+1})])}_{\dim\ \ell+1} \nsubseteq \underbrace{\mathrm{Im}([v_1, \ldots, v_\ell])}_{\dim\ \ell} \ , \tag{14}$$

hence the set $\Omega = \{\alpha \mid [A(\alpha)] \in \mathrm{Im}([v_0, v_1, \ldots, v_\ell])\}$ contains at most ℓ elements. By the uniform distribution of ω, we then have a probability at most $\ell/(|\mathbb{F}| - 1) \leq n/|\mathbb{F}|$ to have $\omega \in \Omega$ that is to have $[a] \in \mathrm{Im}([v_1, \ldots, v_\ell])$. □

We now have all the ingredients to prove Theorem 1.

Proof. (Theorem 1) We will show that for any adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{NTT}}$ in distinguishing $\mathcal{L}(\mathsf{NTT}, \mathsf{Enc}(a^{(0)}))$ from $\mathcal{L}(\mathsf{NTT}, \mathsf{Enc}(a^{(1)}))$ for any chosen elements $a^{(0)}, a^{(1)} \in \mathbb{F}$ is lower than ε, where $\mathcal{L}(\mathsf{NTT}, \mathsf{Enc}(a))$ denotes the δ-random-probing leakage of the procedure NTT_ξ on input $\mathsf{Enc}(a) = \langle \omega, (a_i)_{i=0}^{n-1} \rangle$. Note that this leakage is a tuple in which each coordinate corresponds to an arithmetic instruction in the computation of NTT_ξ that either equals $\perp$ (with probability $1-\delta$) or the input of the instruction. We recall that the advantage is defined as $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{NTT}} = \left|\mathrm{Succ}_{\mathcal{A}}^{\mathsf{NTT}} - \frac{1}{2}\right|$ where

$$\mathrm{Succ}_{\mathcal{A}}^{\mathsf{NTT}} = \Pr\left(\begin{array}{c} (a^{(0)}, a^{(1)}, \mu) \leftarrow \mathcal{A}(\perp) \\ b \leftarrow \{0,1\} \\ \boldsymbol{\ell} \leftarrow \mathcal{L}(\mathsf{NTT}, a^{(b)}) \end{array} : \ \mathcal{A}(a^{(0)}, a^{(1)}, \mu, \boldsymbol{\ell}) = b \right) \tag{15}$$

Without loss of generality, we assume $\mathrm{Succ}_{\mathcal{A}}^{\mathsf{NTT}} \geq \frac{1}{2}$. Indeed, for any adversary with success probability $\frac{1}{2} - \mathrm{Adv}_{\mathcal{A}}^{\mathsf{NTT}}$, there exists an adversary $\mathcal{A}'$ with success probability $\frac{1}{2} + \mathrm{Adv}_{\mathcal{A}}^{\mathsf{NTT}}$ (defined as $\mathcal{A}'(a^{(0)}, a^{(1)}, \boldsymbol{\ell}) = 1 - \mathcal{A}(a^{(0)}, a^{(1)}, \mu, \boldsymbol{\ell})$).

The procedure NTT_ξ is composed of $N = 3n \log n$ arithmetic instructions. In the δ-random-probing model, each of these instructions leaks its input(s) with probability δ. The number of instructions that leak hence follows a binomial distribution with parameters N and δ. Let us denote by max_ℓ the event that ℓ or less instructions leak in the random-probing leakage $\mathcal{L}(\mathsf{NTT}, \mathsf{Enc}(a))$. Since each instruction takes at most two inputs over $\mathbb{F}$, the adversary gets the values of at most 2ℓ intermediate variables whenever max_ℓ occurs. By the Chernoff bound (see Corollary 1), the probability that more than $\ell > N\delta$ arithmetic instructions leak, namely the probability that $\neg\mathsf{max}_\ell$ occurs, satisfies:

$$\Pr(\neg\mathsf{max}_\ell) \leq \psi(\ell, N) \ . \tag{16}$$

From $N = 3n \log n$ and $\ell < \frac{n}{2}$, we get that:

$$\Pr(\neg\mathsf{max}_\ell) \leq \exp\Big(- \frac{(1 - 6\delta \log n)^2}{2 + 12\delta \log n} \ n \Big) \leq \exp\Big(- \frac{(1 - 6\delta \log n)^2}{4} \ n \Big) \ . \tag{17}$$

Now let assume that max_ℓ occurs for some $\ell < \frac{n}{2}$ and let denote $v_1, v_2, \ldots, v_{2\ell}$ the recovered intermediate variables. Without loss of generality, we assume that the recovered intermediate variables are linearly independent. Let us then denote by free the event that $[a] \notin \mathrm{Im}([v_1, \ldots, v_\ell])$. By Lemma 3, we have

$$\Pr(\neg\mathsf{free}) < \frac{n}{|\mathbb{F}|} \ . \tag{18}$$

And let finally denote by succ the event that $\mathcal{A}$ outputs the right bit b on input $(a^{(0)}, a^{(1)}, \mu, \boldsymbol{\ell})$ so that $\mathrm{Succ}_{\mathcal{A}}^{\mathsf{NTT}} = \Pr(\mathsf{succ})$. We can then write:

$$\begin{aligned} \mathrm{Succ}_{\mathcal{A}}^{\mathsf{NTT}} = \Pr(\mathsf{max}_\ell) \Pr(\mathsf{succ} \mid \mathsf{max}_\ell) + \Pr(\neg\mathsf{max}_\ell) \Pr(\mathsf{succ} \mid \neg\mathsf{max}_\ell) \\ \leq \Pr(\mathsf{succ} \mid \mathsf{max}_\ell) + \Pr(\neg\mathsf{max}_\ell) \ . \end{aligned} \tag{19}$$

In the same way, we have

$$\Pr(\mathsf{succ} \mid \mathsf{max}_\ell) \leq \Pr(\mathsf{succ} \mid \mathsf{max}_\ell \cap \mathsf{free}) + \Pr(\neg\mathsf{free}) \ . \tag{20}$$

By Lemma 2, we have that the leakage $\boldsymbol{\ell}$ is statistically independent of $a^{(b)}$ in (15) whenever $\mathsf{max}_\ell \cap \mathsf{free}$ occurs. This directly implies $\Pr(\mathsf{succ} \mid \mathsf{max}_\ell \cap \mathsf{free}) = \frac{1}{2}$, which gives

$$\mathrm{Succ}_{\mathcal{A}}^{\mathsf{NTT}} < \frac{1}{2} + \Pr(\neg\mathsf{max}_\ell) + \Pr(\neg\mathsf{free}) \ . \tag{21}$$

Hence, we finally get

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{NTT}} < \Pr(\neg\mathsf{max}_\ell) + \Pr(\neg\mathsf{free}) = \frac{n}{|\mathbb{F}|} + \exp\Big(- \frac{(1 - 6\delta \log n)^2}{4} \ n \Big) \ , \tag{22}$$

which concludes the proof. □

Security of the Full Multiplication. We now prove the security of the full multiplication. We have the following result:

Theorem 2. *Let ω be a uniform random element of $\mathbb{F}^*$, let $(a_i)_{i=0}^{n-1}$ and $(b_i)_{i=0}^{n-1}$ be uniform ω-encodings of some variables a and b, and let $\delta < 1/(21 \log n)$. The above NTT-based multiplication procedure on input $(a_i)_{i=0}^{n-1}$ and $(b_i)_{i=0}^{n-1}$ is ε-leakage secure in the δ-random-probing leakage model, where*

$$\varepsilon = \frac{2\,n}{|\mathbb{F}|} + 5 \exp\Big(- \frac{(1 - 21\delta \log n)^2}{14}\ n\Big)\ . \tag{23}$$

Proof. The full multiplication is composed of five successive steps:

1. the NTT on input $(a_i)_i$,
2. the NTT on input $(b_i)_i$,
3. the pairwise multiplications $(2n)^{-1} \cdot u_i \cdot r_i$,
4. the NTT on input $(s_i)_i$,
5. the final compression on input $(t_i)_i$.

Let us denote by ℓ_1, ℓ_2, ..., ℓ_5 the number of operations that leak at each of these steps. Since each operation takes up to 2 input variables, the adversary then gets:

- up to $2\ell_1$ variables from the first NTT, each variable providing a linear equation in the a_i's;
- up to $2\ell_2$ variables from the first NTT, each variable providing a linear equation in the b_i's;
- up to ℓ_3 pairs (u_i, r_i),[7] each pair providing a linear equation in the a_i's and a linear equation in the b_i's;
- up to $2\ell_4$ variables in the third NTT (the inverse NTT), each variable providing a linear equation in the s_j's;
- up to ℓ_5 pairs (t_i, t_{i+n}),[8] each pair providing two linear equations in the s_j's.

To sum up, the adversary gets a system composed of

- up to $\ell_1^* = 2\ell_1 + \ell_3$ linear equations of the form

$$\sum_{i=1}^{n} \alpha_{k,i} \cdot a_i = \eta_k \quad \text{for } k = 1, \ldots, \ell_1^* \tag{24}$$

- up to $\ell_2^* = 2\ell_2 + \ell_3$ linear equations of the form

$$\sum_{i=1}^{n} \beta_{k,i} \cdot b_i = \nu_k \quad \text{for } k = 1, \ldots, \ell_2^* \tag{25}$$

- up to $\ell_3^* = 2\ell_4 + 2\ell_5$ linear equations of the form

$$\sum_{j=1}^{2n} \gamma_{k,j} \cdot s_j = \chi_k \quad \text{for } k = 1, \ldots, \ell_3^* \tag{26}$$

we have $s_j = (2n)^{-1} u_j r_j$ for every j, and since u_j and r_j can be expressed as linear combinations of $(a_i)_i$ and of $(b_i)_i$ respectively, for every j, the last ℓ_3^* equations can be rewritten as:

$$\sum_{i=1}^{n} \gamma'_{k,i} \cdot b_i = \chi_k \quad \text{for } k = 1, \ldots, \ell_3^* \tag{27}$$

where the $\gamma'_{k,i}$'s are coefficients that depend on the a_i's.

From these equations, the attacker gains the knowledge that:

[7] Either a multiplication of the form $(2n)^{-1} \cdot u_i$ or a multiplication of the form $(2n)^{-1} u_i \cdot r_i$ leaks. In both cases we consider that the pair (u_i, r_i) is revealed to the adversary.

[8] Either a multiplication $\omega^n \cdot t_{i+n}$ or an addition $t_i + \omega^n t_{i+n}$ leaks. In both cases we consider that the pair (t_i, t_{i+n}) is revealed to the adversary.

1. the encoding $(a_i)_{i=0}^{n-1}$ belongs to some vectorial space

$$\mathcal{S}_1 = \{\boldsymbol{x} \in \mathbb{F}^n \ ; \ M_1 \cdot \boldsymbol{x} = \boldsymbol{\eta}\} \tag{28}$$

of dimension $n-\ell_1^*$ where M_1 is the matrix with coefficients $\alpha_{k,i}$'s, and $\boldsymbol{\eta}$ is the vector with coordinates η_k,

2. the encoding $(b_i)_{i=0}^{n-1}$ then belongs to some vectorial space

$$\mathcal{S}_2 = \{\boldsymbol{x} \in \mathbb{F}^n \ ; \ M_2 \cdot \boldsymbol{x} = (\boldsymbol{\nu}, \boldsymbol{\chi})\} \tag{29}$$

of dimension $n - \ell_2^* - \ell_3^*$ where M_2 is the matrix with coefficients $\beta_{k,i}$'s and $\gamma'_{k,i}$'s and $(\boldsymbol{\nu}, \boldsymbol{\chi})$ is the vector with coordinates ν_k and χ_k.

Following the demonstration of Lemma 2, it can be checked that if

$$(1, \omega, \ldots, \omega^{2n-1}) \notin \mathrm{Im}(M_1) \quad \text{and} \quad (1, \omega, \ldots, \omega^{2n-1}) \notin \mathrm{Im}(M_2) \ ,$$

then the full leakage of the multiplication is statistically independent of a and b, namely the leakage security holds. These two events are denoted free_1 and free_2 hereafter.

Then, following the demonstration of Lemma 3, free_1 occurs with probability at least $1 - \frac{n}{|\mathbb{F}|}$ over a random choice of ω, provided that we have $\mathrm{rank}(M_1) < n$. Then, since the vectorial space $\mathcal{S}_1$ is independent of ω, any possible choice of $(a_i)_{i=0}^{n-1} \in \mathcal{S}_1$ gives rise to some coefficients $\gamma'_{k,i}$'s independent of ω and we have that free_2 occurs with probability at least $1 - \frac{n}{|\mathbb{F}|}$ over a random choice of ω as long as we have $\mathrm{rank}(M_2) < n$. The two conditions on the ranks of M_1 and M_2 are then fulfilled whenever we have

$$\ell_1^* = 2\ell_1 + \ell_3 < n \ , \tag{30}$$

and

$$\ell_2^* + \ell_3^* = 2\ell_2 + \ell_3 + 2\ell_4 + 2\ell_5 < n \ . \tag{31}$$

Let us denote max_i the event that the number of leaking operations ℓ_i at step i is lower than $n/7$, for every i. If max_i occurs for every $i \in \{1, 2, 3, 4, 5\}$, then two above inequalities are well satisfied.

By applying the Chernoff bound, we hence get:

$$\Pr(\neg\mathsf{max}_i) \leq \psi\big(\frac{n}{7}, N_i\big) \ , \tag{32}$$

where N_i is the number of operations at step i, which satisfies $N_i \leq 3n \log n$, which gives

$$\Pr(\neg\mathsf{max}_i) \leq \psi\big(\frac{n}{7}, 3n\log n\big) \leq \exp\Big(-\frac{(1 - 21\delta \log n)^2}{14}\ n\Big) \ . \tag{33}$$

We finally get that the multiplication is ε-leakage secure with

$$\begin{aligned}\varepsilon < \Pr(\neg\mathsf{max}_1) + \Pr(\neg\mathsf{max}_2) + \cdots + \Pr(\neg\mathsf{max}_5) \\ + \underbrace{\Pr(\neg\mathsf{free}_1 \mid \mathsf{max}_1 \wedge \ldots \wedge \mathsf{max}_5)}_{<n/|\mathbb{F}|} + \underbrace{\Pr(\neg\mathsf{free}_2 \mid \mathsf{max}_1 \wedge \ldots \wedge \mathsf{max}_5)}_{<n/|\mathbb{F}|} \ .\end{aligned} \tag{34}$$

□

4 Compositional Security for Arithmetic Programs

In this section we show how to obtain leakage security for a full arithmetic program, composed of several multiplications, additions and subtractions. Since computing addition and subtraction on encoded variables is quite simple, our main contribution is to describe a refreshing procedure which allows us to achieve compositional security.

We first describe our refreshing procedure before explaining how to transform an arithmetic program into a leakage-secure equivalent arithmetic program. Then we provide our compositional security proof.

4.1 Refreshing Procedure

Our refreshing procedure is based on the common approach of adding an encoding of 0. Let $(a_i)_{i=0}^{n-1}$ be an ω-encoding of a variable a. We refresh it into an ω-encoding $(a'_i)_{i=0}^{n-1}$ of a as follows:

1. sample a random ω-encoding $(r_0, r_1, \ldots, r_{n-1}) \leftarrow \mathsf{Enc}_\omega(0)$
2. set $a'_i = a_i + r_i$ for $i = 0$ to $n-1$

The main issue with such an approach is the design of a scheme to sample an encoding of 0 which has the right features for the compositional security. As detailed later, we can prove the compositional security as long as our construction satisfies the two following properties:

- **Uniformity:** it outputs a uniform ω-encoding of 0;
- **Output linearity:** its intermediate variables (*i.e.* the input of elementary operations in the sampler) can each be expressed as a linear combination of the output shares $(r_i)_i$.

We now describe an $\mathsf{Enc}_\omega(0)$ sampler which satisfies these two properties.

Sampling Encodings of 0. At the beginning of the computation of Π, a random ω-encoding of 0 is generated. This is simply done by randomly picking $n-1$ of the n shares and computing the last one accordingly. We will denote by $(e_i)_{i=0}^{n-1}$ this encoding. Note that just as for ω, this encoding can be fully leaked to the adversary. Our sampler then works as follows:

1. pick $n-1$ random values u_0, u_1, …, u_{n-2} over $\mathbb{F}$,
2. output $(r_i)_{i=0}^{n-1} = \mathsf{NTTMult}((u_0, u_1, \ldots, u_{n-2}, 0), (e_0, e_1, \ldots, e_{n-1}))$

where $\mathsf{NTTMult}$ is the NTT-based multiplication described in Section 3.

It is not hard to see that the result is indeed an encoding of 0: since the $(e_i)_i$ encode a 0, then the encoded product is also a 0. The uniformity is slightly more tricky to see. We claim that with overwhelming probability (over the random choice of $(e_i)_i$), the function:

$$(u_0, u_1, \ldots, u_{n-2}) \mapsto \mathsf{NTTMult}((u_0, u_1, \ldots, u_{n-2}, 0), (e_0, e_1, \ldots, e_{n-1})) \;, \tag{35}$$

is invertible. This function is indeed linear and it can be seen as a multiplication by an $(n-1) \times n$ matrix. We empirically validated that this matrix is of rank $n-1$ with overwhelming probability.[9] By discarding one column we can get a full-rank square matrix of dimension $n-1$, allowing the recovery of the $(u_0, u_1, \ldots, u_{n-2})$ from output encoding. Therefore, we have a one-to-one mapping between the vectors $(u_0, u_1, \ldots, u_{n-2}) \in \mathbb{F}^{n-1}$ and the ω-encodings of 0, $(r_i)_{i=0}^{n-1} \in \mathbb{F}^n$ with $\mathsf{Dec}_\omega((r_i)_{i=0}^{n-1}) = 0$.

The output linearity is a direct consequence of the above. Since the u_i's can be expressed as linear combinations of the r_i's, then all the intermediate variables of the sampling procedure can be expressed as such linear combinations as well.

4.2 Arithmetic Program Compiler

We consider an arithmetic program P processing variables defined over a prime field $\mathbb{F}$. We show how to transform such a program into a leakage-secure arithmetic program Π. Each arithmetic instruction of P gives rise to a corresponding *gadget* in Π that works on encodings. We describe these different gadgets hereafter.

Copy gadget. The copy gadget simply consists in applying a refreshing procedure to copy an encoded variable into the same freshly encoded variable. Let $(a_i)_{i=0}^{n-1}$ be an ω-encoding of a. The copy gadget compute an ω-encoding $(a'_i)_{i=0}^{n-1}$ of a as:

$$(a'_0, a'_1, \ldots, a'_{n-1}) \leftarrow \mathsf{refresh}(a_0, a_1, \ldots, a_{n-1})$$

[9] To avoid to rely on an empirical assumption, one could easily check whether the generated encoding $(e_i)_i$ gives rise to a full-rank linear transformation.

The copy gadget is used whenever an output ω-encoding $(a_i)_{i=0}^{n-1}$ from some previous gadget is used as an input of several following gadgets. If $(a_i)_{i=0}^{n-1}$ is to be used in input of N following gadgets, one makes $N-1$ extra copies (in such a way that each new copy enters the next copy gadget):

$$(a_i)_{i=0}^{n-1} \to (a_i^{(2)})_{i=0}^{n-1} \to \cdots \to (a_i^{(N)})_{i=0}^{n-1}$$

This way, each fresh encoding $(a_i^{(j)})_{i=0}^{n-1}$ enters at most two different gadgets: the copy gadget and one of the N computation gadgets.

Addition gadget. Let $(a_i)_{i=0}^{n-1}$ be an ω-encoding of a and $(b_i)_{i=0}^{n-1}$ be an ω-encoding of b. To compute an ω-encoding $(c_i)_{i=0}^{n-1}$ of $c = a + b$, we simply compute:

$$(c_0, c_1, \ldots, c_{n-1}) \leftarrow \mathsf{refresh}(a_0 + b_0, a_1 + b_1, \ldots, a_{n-1} + b_{n-1})$$

Subtraction gadget. Let $(a_i)_{i=0}^{n-1}$ be an ω-encoding of a and $(b_i)_{i=0}^{n-1}$ be an ω-encoding of b. To compute an ω-encoding $(c_i)_{i=0}^{n-1}$ of $c = a + b$, we simply compute:

$$(c_0, c_1, \ldots, c_{n-1}) \leftarrow \mathsf{refresh}(a_0 - b_0, a_1 - b_1, \ldots, a_{n-1} - b_{n-1})$$

Multiplication gadget. Let $(a_i)_{i=0}^{n-1}$ be an ω-encoding of a and $(b_i)_{i=0}^{n-1}$ be an ω-encoding of b. To compute an ω-encoding $(c_i)_{i=0}^{n-1}$ of $c = a \cdot b$, we simply compute:

$$(c_0, c_1, \ldots, c_{n-1}) \leftarrow \mathsf{refresh}\big(\mathsf{NTTMult}((a_0, a_1, \ldots, a_{n-1}), (b_0, b_1, \ldots, b_{n-1}))\big)$$

where $\mathsf{NTTMult}$ denotes the NTT-based multiplication described in Section 3.

4.3 Compositional Security

The compositional security of our construction is based on the two following properties of the refreshing procedure:

- **Uniformity:** for a given $\omega \in \mathbb{F}^*$ and a given value $a \in \mathbb{F}$, the ω-encoding $(a'_i)_{i=0}^{n-1}$ in output of the refreshing procedure is uniformly distributed and independent of the input ω-encoding $(a_i)_{i=0}^{n-1}$;
- **I/O linear separability:** the intermediate variables of the refreshing procedure can each be expressed as a deterministic function of a linear combination of the $(a_i)_i$ and a linear combination of the $(a'_i)_i$.

The uniformity property is a direct consequence of the uniformity of the $\mathsf{Enc}_\omega(0)$ sampler. The I/O linear separability holds from the output linearity of the $\mathsf{Enc}_\omega(0)$ sampler since the shares $(r_i)_i$ output by the sampler satisfy $r_i = a'_i - a_i$ for every i, implying that any linear combination $\sum_i \gamma_i r_i$ equals $\sum_i \gamma_i a'_i - \sum_i \gamma_i a_i$ and is hence a deterministic function of a linear combination $\sum_i \gamma_i a'_i$ and a linear combination $\sum_i \gamma_i a_i$.

The I/O linear separability of the refreshing procedure implies that its leakage can be split into some leakage depending only on its input encoding, which is the output (before refreshing) from a previous gadget, and some leakage depending only on its output encoding, which is the input of a next gadget. This way, the full leakage can be split into subleakages each depending on the input/output of one gadget. Moreover, the uniformity property implies that all these subleakages are mutually independent. They can hence be analyzed separately: if none of them reveal information, then the full leakage does not reveal information either. This is illustrated on Figure 1, where the input/output encodings of each gadget (and the corresponding separated leakage) is represented by a different color.

The compositional security of our construction is formalized in the following theorem.

Theorem 3. *Let P be an arithmetic program taking some input $\boldsymbol{x} \in \mathbb{F}^s$ and let Π denotes the corresponding program protected with n-size encodings as described above. For every $\delta < 1/(33 \log n)$, Π is ε-leakage secure in the δ-random-probing model where*

$$\varepsilon = 3|P| \cdot \left(2 \exp\left(-\frac{(1 - 33\delta \log n)^2}{22}\, n\right) + \frac{2n}{|\mathbb{F}|}\right) , \tag{36}$$

where $|P|$ denotes the size of P i.e. its number of arithmetic instructions.

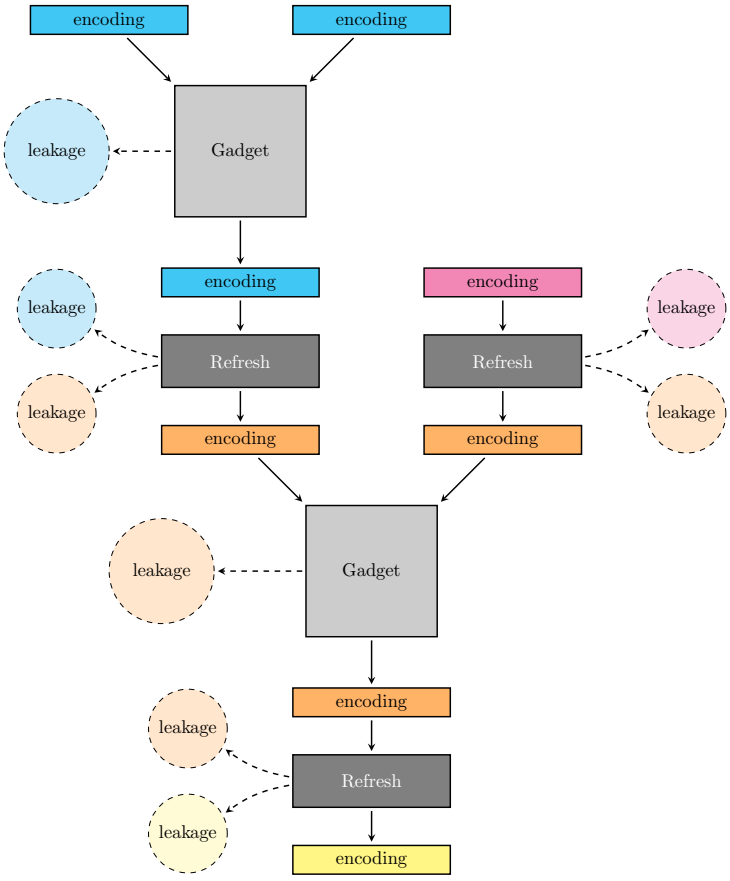

Fig. 1. Illustration of our compositional security.

Proof. Let $|\Pi|$ denotes the number of gadgets in Π. Since the output of each gadget is refreshed (and nothing more), the number of call to the refreshing procedure is also $|\Pi|$. Each arithmetic instruction in P gives rise to one associated gadget, plus up to 2 copy gadgets if necessary. We hence deduce $|\Pi| \leq 3|P|$.

Let us denote by rmax the event that at most $\frac{n}{11}$ operations leak in each refreshing. By applying the Chernoff bound (see Corollary 1), we have

$$\Pr(\neg\mathsf{rmax}) \leq |\Pi| \cdot \psi\big(\frac{n}{11}, N_{\mathrm{ref}}\big) \ , \tag{37}$$

where N_{ref} denotes the number of elementary operations in the refreshing procedure. Let us further denote by gmax the event that at most $\frac{n}{11}$ operations leak in each gadget (without refreshing). In the same way as above, we have

$$\Pr(\neg\mathsf{gmax}) \leq \sum_{i=1}^{|\Pi|} \psi\big(\frac{n}{11}, N^{(i)}\big) \leq |\Pi| \cdot \psi\big(\frac{n}{11}, N_{\mathrm{gad}}\big) \ , \tag{38}$$

where $N^{(i)}$ denotes the number of elementary operations in the ith gadget and where N_{gad} denotes the max (which is reached by the multiplication gadget).

In the following, we shall denote by $(a_j^{(i)})_j$ and $(b_j^{(i)})_j$ the input encodings of the ith gadget of Π and by $(c_j^{(i)})_j$ the output encoding (before refreshing) of the ith gadget of Π. Let us further denote by $\mathcal{L}$ the full δ-random-probing leakage of Π, so that we have:

$$\mathcal{L} = \bigcup_{i=1}^{|\Pi|} \mathcal{G}^{(i)} \ \cup \ \bigcup_{i=1}^{|\Pi|} \mathcal{R}^{(i)} \tag{39}$$

where $\mathcal{G}^{(i)}$ denotes the leakage from the ith gadget (without refreshing) and where $\mathcal{R}^{(i)}$ denotes the leakage of the ith refresh. Specifically, $\mathcal{G}^{(i)}$ and $\mathcal{R}^{(i)}$ are families of intermediate variables (inputs of elementary operations) that are revealed by the δ-random-probing leakage. If rmax and gmax occurs, we have $|\mathcal{G}^{(i)}| \leq \frac{2n}{11}$ and $|\mathcal{R}^{(i)}| \leq \frac{2n}{11}$.

According the the I/O linear separability property of the refreshing procedure, we can define a *separated leakage* $\mathcal{L}'$ as

$$\mathcal{L}' = \bigcup_{i=1}^{|\Pi|} \big(\mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}\big) \tag{40}$$

where $\mathcal{A}^{(i)}$ is a set of linear combinations of $(a_j^{(i)})_j$, $\mathcal{B}^{(i)}$ is a set of linear combinations of $(b_j^{(i)})_j$, $\mathcal{C}^{(i)}$ is a set of linear combinations of $(c_j^{(i)})_j$, such that $\mathcal{L}$ is a deterministic function of $\mathcal{L}'$. This implies that if $\mathcal{L}'$ is statistically independent of the program input $\boldsymbol{x}$, then so is $\mathcal{L}$. The remaining of the proof consists in showing that the former occurs with overwhelming probability (for a sound choice of the parameters).

We shall bound the probability (over the distribution of ω) that the family $\mathcal{L}'$ is statistically dependent on $\boldsymbol{x}$, hereafter denoted $\boldsymbol{x} \triangledown \mathcal{L}'$. We have

$$\boldsymbol{x} \triangledown \mathcal{L}' = \bigvee_{i=1}^{|\Pi|} \big(\boldsymbol{x} \triangledown \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}\big) \ . \tag{41}$$

By the uniformity property of the refreshing, we have that, given the program input $\boldsymbol{x}$, the different families of input/output shares $\big\{(a_j^{(i)})_{j=0}^{n-1}, (b_j^{(i)})_{j=0}^{n-1}, (c_j^{(i)})_{j=0}^{n-1}\big\}$ are mutually independent. We hence get

$$\Pr(\boldsymbol{x} \triangledown \mathcal{L}') \leq \sum_{i=1}^{|\Pi|} \Pr\big(\boldsymbol{x} \triangledown \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}\big) \ . \tag{42}$$

We can then upper bound the probability $\Pr\big(\boldsymbol{x} \triangledown \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}\big)$ when the ith gadget is a secure multiplication by following the proof of Theorem 2. The only difference is that the attacker gets additional linear combinations of the input/output shares from the refreshing procedures. Specifically, we would have

– up to $\ell_1^* = 2\ell_1 + \ell_3 + 2\ell_1'$ linear combinations of the form $\sum_i \alpha_{k,i} a_i$, for $1 \le k \le \ell_1^*$;
– up to $\ell_2^* = 2\ell_2 + \ell_3 + 2\ell_2'$ linear combinations of the form $\sum_i \beta_{k,i} b_i$, for $1 \le k \le \ell_2^*$;
– up to $\ell_3^* = 2\ell_4 + 2\ell_5 + 2\ell_3'$ linear combinations of the form $\sum_{i,j} \gamma_{k,j} s_j$, for $1 \le k \le \ell_3^*$;

where ℓ_1', ℓ_2' and ℓ_3', are the number of leaking operations in the input/output refreshing procedures. Taking the constraint $\ell_i < \frac{n}{11}$ and $\ell_i' < \frac{n}{11}$ for every i, we still get $\ell_1^* + \ell_3^* < n$ and $\ell_2^* + \ell_3^* < n$. That is, if rmax and gmax occurs, we get

$$\Pr\left(\boldsymbol{x} \triangledown \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)} \mid \mathsf{rmax} \wedge \mathsf{gmax}\right) \le \frac{2n}{|\mathbb{F}|} \,. \tag{43}$$

For copy, addition and subtraction gadgets, the proof is quite simple. When an operation leaks in such a gadget, it reveals one shares from each input encoding. We hence get less that $\frac{n}{11}$ linear combinations on each input encoding (from the gadget leakage), plus $\frac{2n}{11}$ linear combinations on each input encoding (from their respective refreshing), plus $\frac{2n}{11}$ linear combinations on the output encoding, which can be split into independent linear combinations on the two input encodings. We clearly get less than n linear combinations on each encoding, which allows us to apply Lemma 3 and to obtain (43) for every kind of gadget.

We finally get

$$\begin{aligned}\Pr(\boldsymbol{x} \triangledown \mathcal{L}') &\le \Pr(\neg\mathsf{rmax}) + \Pr(\neg\mathsf{gmax}) + \sum_{i=1}^{|\Pi|} \Pr\left(\boldsymbol{x} \triangledown \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)} \mid \mathsf{rmax} \wedge \mathsf{gmax}\right) \\ &\le |\Pi| \cdot \left(\psi\big(\frac{n}{11}, N_{\mathrm{ref}}\big) + \psi\big(\frac{n}{11}, N_{\mathrm{gad}}\big) + \frac{2n}{|\mathbb{F}|}\right) ,\end{aligned}$$

which together with $N_{\mathrm{ref}}, N_{\mathrm{gad}} < 3n \log n$ concludes the proof. □

5 From Arithmetic Random Probing to Noisy Leakage

5.1 Logical Programs

The definition of a *logical program* is analogous to the definition of an arithmetic program but it is composed of logical instructions over $\{0,1\}^w$ such as the bitwise AND, OR, XOR, logical shifts and rotations, as well as the addition, subtraction, and multiplication modulo 2^w (namely typical instructions of a w-bit processor). In the ε-noisy leakage model, a logical program leaks an ε-noisy leakage function $f(\mathtt{m}_j, \mathtt{m}_k)$ of the pair of inputs of each logical instruction $\mathtt{m}_i \leftarrow \mathtt{m}_j * \mathtt{m}_k$.

The security reduction of Duc *et al.* (Lemma 1) then implies that a logical program Π that is secure against δ-random-probing leakage is also secure against δ'-noisy leakage with $\delta' = \delta/2^{2w}$.

5.2 A Generic Reduction

We then have the following reduction of random-probing model for a logical programs, to the random-probing model for an arithmetic programs:

Lemma 4. *Let Π be a ε-leakage secure arithmetic program in the δ-random-probing model, then there exists a functionally equivalent logical program Π' that is ε-leakage secure in the δ'-random-probing model for some δ' satisfying*

$$\delta' = 1 - (1-\delta)^{1/N} \ge \frac{\delta}{N} \quad \textit{with} \quad N = O\Big(\frac{1}{w} \log|\mathbb{F}| \log\big(\frac{1}{w}\log|\mathbb{F}|\big)\Big) \,. \tag{44}$$

Proof. The logical program Π' is simply the program Π where arithmetic instructions are built from several w-bit logical instructions. It is well known that the addition and subtraction on $\mathbb{F}$ can be computed in $N = O\big(\frac{1}{w}\log|\mathbb{F}|\big)$ elementary (w-bit) operations, and that the multiplication on $\mathbb{F}$ can be computed from $N = O\big(\frac{1}{w}\log|\mathbb{F}| \log\big(\frac{1}{w}\log|\mathbb{F}|\big)\big)$ elementary (w-bit) operations.

Assume that there exists an adversary $\mathcal{A}'$ with advantage ε that makes use of a δ'-random-probing leakage on Π', then we show that there exists an adversary $\mathcal{A}$ with advantage ε that makes use of a δ-random-probing leakage on Π. Since by assumption no such adversary $\mathcal{A}$ exists, then by contraposition neither does such adversary $\mathcal{A}'$, meaning that Π' is indeed ε-leakage secure in the δ'-random-probing model.

We construct an adversary $\mathcal{A}$ that is given the full input to an arithmetic instruction of Π whenever at least one of the corresponding logical instruction leaks in Π'. Informally, it is clear that this can only increase the success probability. To make this reasoning formal, we need to construct an adversary $\mathcal{A}$ that receives the strengthened leakage, resamples it to make its distribution identical to that of the δ'-random-probing leakage on Π' and then call $\mathcal{A}'$. When $\mathcal{A}$ receives $\perp$ as leakage for an arithmetic instruction, it simply sends $\perp$ to $\mathcal{A}'$ for all the corresponding logical instructions. When it receives the full input of the arithmetic instruction (meaning that at least one corresponding logical instruction of Π' must leak), it can compute all the inputs of the corresponding logical instructions in Π', and reveal each of them to $\mathcal{A}'$ with some (biased) given probability. Since we do not consider the computational complexity of the adversaries, the easiest way to achieve a perfect simulation is to use rejection sampling. Namely, for every logical instruction in the group, the input is revealed with probability δ'. If at the end of the group, no input was revealed, simply restart the revealing process for the same group. This way, we have constructed an adversary $\mathcal{A}$ using a δ-random-probing leakage on Π where

$$\delta = 1 - (1 - \delta')^N,$$

for $N = O\big(\log|\mathbb{F}|\log\log|\mathbb{F}|\big)$. Since by assumption no such adversary exists, this means that no adversary $\mathcal{A}'$ exists with advantage ε that makes use of a δ'-random-probing leakage on Π'. □

Combining the above lemma with Lemma 1, and considering a constant word-size w, we get a tight reduction of the security in the noisy leakage model for logical program to the security in the random-probing model for arithmetic program:

Lemma 5. *Let Π be a ε-leakage secure arithmetic program in the δ-random-probing model, then there exists a functionally equivalent logical program Π' that is ε-leakage secure in the δ'-noisy leakage model for some δ' satisfying*

$$\delta' = \frac{\delta}{O(\log|\mathbb{F}|\log\log|\mathbb{F}|)} \ . \tag{45}$$

5.3 Application to Our Scheme

In the previous section we have shown that for $\delta = O(1/\log n)$ our construction is ε-leakage secure in the δ-random-probing model with

$$\varepsilon = \mathsf{negl}(\lambda) + \mathsf{negl}'(n) \tag{46}$$

where negl and negl' are some negligible functions and where λ is some security parameter such that $\log|\mathbb{F}| = \lambda + \log n$.

By applying the above reduction to our construction (and recalling that we have $|\mathbb{F}| = O(n)$), we obtain the following corollary of Theorem 3:

Corollary 3. *Let Π' denotes the secure logical program corresponding to our construction (see Section 4). Π' is ε-leakage secure in the δ-noisy leakage model where $\varepsilon = \mathsf{negl}(\lambda) + \mathsf{negl}'(n)$ and $\delta' = O\big(1/((\log n)^2\log\log n)\big)$.*

6 Practical Aspects and Open Problems

Securing arbitrary computation. Although our scheme is described to work on a finite field $\mathbb{F}$ with specific structure, it can be used to secure any arbitrary computation represented as a Boolean circuit. Indeed, it is possible to embed a Boolean circuit into an arithmetic program over $\mathbb{F}$. Each bit is simply represented by an element $a \in \{0,1\} \subseteq \mathbb{F}$. The binary multiplication then matches with the $\mathbb{F}$-multiplication over this subset. Regarding the binary addition $\oplus$, it can be implemented with operations over $\mathbb{F}$ as:

$$a \oplus b = a + b - 2ab \ , \tag{47}$$

for every $a, b \in \{0,1\} \subseteq \mathbb{F}$. Of course such an embedding comes at a high cost in practice and our scheme would not be efficient to protect *e.g.* an AES computation. However, our scheme is asymptotically more efficient than previous ISW-based schemes meaning that there exists some masking order n for which an implementation of our scheme would be more efficient than an implementation of a previous scheme. Moreover and as discussed hereafter, we think that our scheme could be practically improved in many ways.

Practical efficiency. For any cryptographic computation on a base field $\mathbb{F}$ with appropriate structure, our scheme should be very efficient in practice. We recall that the field should be such that $|\mathbb{F}| = \alpha \cdot n + 1$, for n being a power of 2 and α being large enough so that n/α is negligible. A 256-bit prime field such as those used in Elliptic Curve Cryptography could for instance satisfy these criteria. An interesting open issue would be to extend our scheme to work on other algebraic structures and in particular on binary fields (*e.g.* to efficiently secure the AES) or on rings used in lattice-based cryptography.

On the size of the field. We note that we need a 'big' field (typically of size $128 + 2\log n$) in order to have enough randomness when picking ω. However this might be a proof artefact and the scheme could be secure for some constant ω and/or using smaller fields. Another direction of improvement would be to mitigate or remove this constraint with an improved construction and/or proof technique.

Packing encodings. Finally our scheme could also probably be improved by using the principle of packed secret sharing as suggested in [ADF16,ADD+15] since our encoding is a kind of randomized Shamir's secret sharing.

References

[ADD+15] Marcin Andrychowicz, Ivan Damgård, Stefan Dziembowski, Sebastian Faust, and Antigoni Polychroniadou. Efficient leakage resilient circuit compilers. pages 311–329, 2015.

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. *Circuit Compilers with O(1/log(n)) Leakage Rate*, pages 586–615. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

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

[BBP+16] 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, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, 2016.

[BBP+17] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. *Private Multiplication over Finite Fields*, pages 397–426. Springer International Publishing, Cham, 2017.

[BCPZ16] 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, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. pages 521–536, 2006.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23(4):493–507, 12 1952.

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

[CPRR14] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. pages 410–424, 2014.

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. pages 170–187, 2014.

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

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. pages 445–465, 2010.
[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. pages 293–302, 2008.
[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. pages 251–261, 2001.
[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. pages 31–40, 2012.
[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. pages 463–481, 2003.
[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. pages 388–397, 1999.
[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. pages 104–113, 1996.
[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). pages 278–296, 2004.
[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. pages 142–159, 2013.
[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. pages 413–427, 2010.
[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.
[SPY+09] Francois-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. Cryptology ePrint Archive, Report 2009/341, 2009. `http://eprint.iacr.org/2009/341`.

A Number Theoretic Transform

The Number Theoretic Transform (NTT) is essentially a (Fast) Fourier Transform defined in a finite field (or ring) where inaccurate floating point or complex arithmetic can be avoided. The NTT can be used to multiply two polynomials over a finite field in quasilinear complexity. Let $\mathbb{F}_p$ be a prime finite field such that $d \mid p-1$ for some integer d ($\mathbb{F}_p$ contains d-th roots of unity) and let A be a $(d-1)$-degree polynomial over $\mathbb{F}_p[x]$ such that $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{d-1} x^{d-1}$. For a given primitive d-th root of unity ξ, the NTT maps the coefficients of A to the evaluations $A(\xi^i)$ with $1 \leq i \leq d$:

$$\mathsf{NTT}_\xi : (a_0, a_1, \ldots, a_{d-1}) \mapsto (A(\xi^1), A(\xi^2), \ldots, A(\xi^d)) \ . \tag{48}$$

For d being a power of two, the NTT can be computed in time complexity $O(d \log d)$. To show this, let us define A_0 and A_1, the two $(\frac{d}{2}-1)$-degree polynomials

$$A_0(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{d-2} x^{\frac{d}{2}-1}$$
$$A_1(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{d-1} x^{\frac{d}{2}-1}$$

which satisfy

$$A(x) = A_0(x^2) + x A_1(x^2).$$

The problem of evaluating $A(x)$ at each d-th root of unity ξ^i, for $1 \leq i \leq d$, is reduced to the problem of evaluating $A_0(x)$ and $A_1(x)$ at the points ξ^{2i}, for $1 \leq i \leq \frac{d}{2}$, and we can combine the results with $A(\xi) = A_0(\xi^2) + \xi A_1(\xi^2)$. The polynomials $A_0(x)$ and $A_1(x)$ can also be evaluated at the points ξ^{2i} with the same divide and conquer strategy, using the polynomials A_{00}, A_{01}, A_{10}, A_{11} satisfying

$$A_0(x) = A_{00}(x^2) + x A_{01}(x^2) \ \text{ and } \ A_1(x) = A_{01}(x^2) + x A_{11}(x^2) \ .$$

This divide and conquer strategy can be iterated $\log_2(d)$ times. At the t-th step we have 2^t polynomials $A_{\boldsymbol{u}}$ of degree $\frac{d}{2^t}$ for $\boldsymbol{u} \in \{0,1\}^t$ that must be evaluated in ξ^j for $j = 2^t, 2 \cdot 2^t, \ldots, \frac{d}{2^t} \cdot 2^t$, which makes a total of $2^t \cdot \frac{d}{2^t} = d$ evaluations. Moreover, from $\xi^{j+\frac{d}{2}} = -\xi^j$ we have

$$A_{\boldsymbol{u}}(\xi^j) = A_{\boldsymbol{u}|0}(\xi^{2j}) + \xi^j A_{\boldsymbol{u}|1}(\xi^{2j}) \ \text{ and } \ A_{\boldsymbol{u}}(\xi^{j+\frac{d}{2}}) = A_{\boldsymbol{u}|0}(\xi^{2j}) - \xi^j A_{\boldsymbol{u}|1}(\xi^{2j}) \tag{49}$$

implying that the number of evaluations can be merely divided by two.

In practice, we start with $t = \log_2(d)$, where we have $2^t = d$ constant polynomials $A_{\boldsymbol{u}} = a_{\varphi(\boldsymbol{u})}$ with $\varphi(\boldsymbol{u})$ denoting the integer corresponding to the binary expansion $\boldsymbol{u} \in \{0,1\}^{\log_2(d)}$. Then we iterate (49) for t from $\log_2(d)$ down to 1 where we have our d evaluations of A. The overall process is summarized hereafter:

1. $(c_0, c_1, \ldots, c_{d-1}) \leftarrow (a_0, a_1, \ldots, a_{d-1})$
2. for $t = \log_2(d) - 1$ down to 1:
3. $\quad j = 2^t$; $k = 2^{d-t-1}$
4. $\quad$ for $i \in \bigcup_{\ell=0}^{k-1} U_{j,\ell}$
5. $\quad\quad (c_i, c_{i+j}) \leftarrow (c_i + \xi^j c_{i+j}, c_i - \xi^j c_{i+j})$

where $U_{j,\ell} = \{(2\ell j, \ldots, (2\ell+1)j - 1)\}$ and where the index shiftings of c are done modulo d, *i.e.* $c_{i+j} = c_{i+j \mod d}$. It can be checked that the above evaluation of NTT_ξ takes a total of $\frac{d \log d}{2}$ multiplications, $\frac{d \log d}{2}$ additions and $\frac{d \log d}{2}$ subtractions.

Using the NTT with a dth root of unity, we can efficiently compute the product $C(x) = A(x) \cdot B(x)$ for any two polynomials $A, B \in \mathbb{F}_p[x]$ of degree up to $n-1$ with $d = 2n$. We first apply the NTT to get d evaluations of both polynomials:

$$(A(\xi^1), A(\xi^2), \ldots, A(\xi^d)) = \mathsf{NTT}_\xi(a_0, a_1, \ldots, a_{n-1}, 0, \ldots, 0)$$
$$(B(\xi^1), B(\xi^2), \ldots, B(\xi^d)) = \mathsf{NTT}_\xi(b_0, b_1, \ldots, b_{n-1}, 0, \ldots, 0)$$

from which we get d evaluations of C by $C(\xi^i) = A(\xi^i) \cdot B(\xi^i)$ for $1 \leq i \leq d$. Finally, we can recover the coefficients of the output polynomial C by computing the inverse NTT on $(C(\xi), C(\xi^1), \ldots, C(\xi^d))$, which satisfies

$$(c_0, c_1, \ldots, c_d) = \mathsf{NTT}_\xi^{-1}(C(\xi^1), C(\xi^2), \ldots, C(\xi^d)) = \mathsf{NTT}_{\xi^{-1}}\big(\frac{1}{d}C(\xi^1), \frac{1}{d}C(\xi^2), \ldots, \frac{1}{d}C(\xi^d)\big)\ .$$

Appendix E

Probing Security through Input-OutputSeparation & Revisited Quasilinear Masking

Hereafter is appended a revised version of our paper [GPRV21], joint work with Dahmun Goudarzi, Thomas Prest and Damien Vergnaud, published at **TCHES 2021**.

Probing Security through Input-Output Separation and Revisited Quasilinear Masking*

Dahmun Goudarzi[1], Thomas Prest[2], Matthieu Rivain[3] and Damien Vergnaud[4,5]

[1] Independent researcher
dahmun.goudarzi@gmail.com

[2] PQShield, Oxford, United Kingdom
thomas.prest@pqshield.com

[3] CryptoExperts, Paris, France
matthieu.rivain@cryptoexperts.com

[4] Sorbonne Université, CNRS, LIP6, Paris, France

[5] Institut Universitaire de France, Paris, France

Abstract. The probing security model is widely used to formally prove the security of masking schemes. Whenever a masked implementation can be proven secure in this model with a reasonable *leakage rate*, it is also provably secure in a realistic leakage model known as the *noisy leakage model*. This paper introduces a new framework for the composition of probing-secure circuits. We introduce the security notion of *input-output separation* (IOS) for a refresh gadget. From this notion, one can easily compose gadgets satisfying the classical probing security notion –which does not ensure composability on its own– to obtain a *region probing secure* circuit. Such a circuit is secure against an adversary placing up to t probes in each gadget composing the circuit, which ensures a tight reduction to the more realistic noisy leakage model. After introducing the notion and proving our composition theorem, we compare our approach to the composition approaches obtained with the (Strong) Non-Interference (S/NI) notions as well as the Probe-Isolating Non-Interference (PINI) notion. We further show that any uniform SNI gadget achieves the IOS security notion, while the converse is not true. We further describe a refresh gadget achieving the IOS property for any linear sharing with a quasilinear complexity $\Theta(n \log n)$ and a $O(1/\log n)$ leakage rate (for an n-size sharing). This refresh gadget is a simplified version of the quasilinear SNI refresh gadget proposed by Battistello, Coron, Prouff, and Zeitoun (ePrint 2016). As an application of our composition framework, we revisit the quasilinear-complexity masking scheme of Goudarzi, Joux and Rivain (Asiacrypt 2018). We improve this scheme by generalizing it to any base field (whereas the original proposal only applies to field with nth powers of unity) and by taking advantage of our composition approach. We further patch a flaw in the original security proof and extend it from the random probing model to the stronger region probing model. Finally, we present some application of this extended quasilinear masking scheme to AES and MiMC and compare the obtained performances.

Keywords: Probing Security · Composition · Quasilinear Masking · IOS Notion

*Revised version of a paper published in IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(3). The current version is dated from June 2022.

1 Introduction

In cryptography, side-channel attacks are all attacks based on extracting information from a physical implementation of a cryptosystem. Rather than exploiting some weakness in the underlying cryptographic algorithm, the leakage information is exploited by attackers to extract the secret key from a specific implementation.

Probing security is a notion put forward by Ishai, Sahai and Wagner in [31] to evaluate the security of a circuit against a class of physical attacks. Specifically, they consider *t-probing attacks* in which the adversary has the ability to place some probes on t wires of a circuit processing some secrets. The circuit is said to be *t-probing secure* if no information leaks from the values of the t probed wires. More formally, one should be able to perfectly simulate the distribution of the probed wires without any knowledge on the secrets. In their paper, Ishai *et al.* propose a scheme, the so-called ISW scheme, to compile a circuit into a new randomized circuit (*i.e.* a circuit featuring random generation gates) which is resistant to t-probing attacks. Their scheme used some additive secret sharing (a.k.a. Boolean masking) of the processed variables. Specifically, each variable x is split into $n \geq 2$ variables $x_1, x_2, \ldots, x_n$, called the *shares*, which are uniformly distributed among n-tuples satisfying $x = x_1 + x_2 + \cdots + x_n$ (where $+$ is the addition on $\mathbb{F}_2$ in the original scheme).

Using such an additive sharing to protect a cryptographic computation was already proposed in 1999 as a protection against side-channel attacks [20, 28]. Many *masking schemes* describing efficient implementations of ciphers protected at some given (low) orders were published in the early 2000's, see *e.g.* [35, 3, 36]. In this context, the probing security notion is analogous to the security against so-called *higher-order* side-channel attacks. In such an attack, an adversary uses t leakage points from a power consumption trace (or electromagnetic trace) to extract information on the secret. If properly implemented, a t-probing secure scheme achieves provable security against this kind of attacks. The ISW scheme had hence a strong impact on the side-channel research community and it was used as a building block in many popular masking schemes, see *e.g.* [41, 33, 18, 24, 22, 40, 44, 30, 11, 12, 32].

Although an ISW-based masking scheme can achieve some level of resistance against side-channel attacks, the probing security notion is not fully satisfactory in this context. In practice a side-channel adversary gets some leakage on the full computation and has no reason to limit herself to t leakage points. Nevertheless, the side-channel leakage is often (or can be made) *noisy* and the noise is known to be amplified by the masking order [20]. This was the motivation behind the formal *noisy leakage model* introduced by Prouff and Rivain in [39]. In this model, every variable (or wire) x in the computation leaks a noisy function $f(x)$. The noisy property is captured by assuming that the bias introduced in the distribution of x by an observation of $f(x)$ is smaller than some bound δ.

Subsequently, Duc, Dziembowski and Faust showed that the security in the noisy leakage model could be obtained for a probing-secure scheme through a security reduction [25]. In a nutshell, the so-called DDF reduction considers an intermediate leakage model called the *random-probing model*, which was already considered by Ishai *et al.* in [31] and formalized by Ajtai in [2], in which each variable (or wire) is leaked to the adversary with a given probability p. By applying the Chernoff bound, one gets that a t-probing secure circuit $\widehat{C}$ is also p-random probing secure with $p = O(t/|\widehat{C}|)$ (where $|\widehat{C}|$ denotes the number of wires of $\widehat{C}$). Duc *et al.* could then show a transition from the p-random probing security to the δ-noisy leakage security with $\delta = O(p/|\mathbb{K}|)$ where $|\mathbb{K}|$ is the base field of the computation. It was recently shown that the impact of the field size can be relaxed by refining the granularity of the computation [29] or considering alternative definitions of the noisy leakage model [38].

The DFF reduction and the obtained security in the noisy leakage model is thus mainly

impacted by the *leakage rate* (or *probing rate*) which is the ratio between the number of tolerated probes and the size of the circuit [6]. In order to tolerate a significant leakage parameter $\delta = O(t/|\widehat{C}|)$, the leakage rate should be as close as possible to 1. In particular, one should be able to tolerate a number of probes that grows linearly with the circuit. To this aim, the circuit should achieve the stronger notion of *region probing security* formalized by Andrychowicz, Dziembowski, and Faust in [6], namely it should be separable into regions that each tolerate some amount of probes independently of the total size of the circuit. This notion was already considered in the work of Ishai *et al.* and their scheme was shown to be region probing secure. Specifically, it can tolerate up to $t < n/2$ probes per protected gate, or *gadget*, for a masking order n. Since the ISW gadgets require $O(n^2)$ operations, the obtained leakage rate is of $O(1/n)$. Such a leakage rate is not fully satisfactory since it implies that the leakage noise should decrease linearly with the number of shares. In particular, no security can be obtained for the ISW gadgets in the context of a constant leakage rate (*i.e.* on a given target device) and some practical attacks were exhibited to underline this issue [9].

Fortunately, some schemes are known that achieve constant (or quasi-constant) leakage rates. Such a scheme was first proposed by Ajtai in [2] which achieves random probing security with leakage rate $O(1)$. Another scheme, partly based on Ajtai's work, was proposed by Andrychowicz, Dziembowski, and Faust in [6] which achieves probing security with leakage rate $O(1/\log n)$, and random-probing security with leakage rate $O(1)$. More recently, Ananth, Ishai and Sahai [5] have proposed a conceptually simpler approach to achieve random-probing security with leakage rate $O(1)$. This approach has been further improved by Belaïd, Coron, Prouff, Rivain and Taleb in [13]. In terms of complexity, all these proposals imply a size of the protected circuit of $O(|C|n^2)$ or larger, where $|C|$ is the size of the original circuit. This was recently improved by Goudarzi, Joux and Rivain who proposed a scheme making use of a Fast Fourier Transform-based (FFT) polynomial multiplication to obtain the first construction achieving a $O(|C|n \log n)$ complexity with a $O(1/\log n)$ leakage rate. Unfortunately, their security proof has a flaw that we exhibit in this paper. Moreover, their scheme is restricted to working on base fields including the nth powers of unity, which notably excludes fields of characteristic 2 that are yet essential in some cryptographic primitives (such as the AES block cipher).

In [8], Barthe, Belaïd, Dupressoir, Fouque, Grégoire, Strub and Zucchini formalized the notion of *composable gadgets* which notably allows to prove region probing security. More precisely, they introduced the notion of *Strong Non-Interference* (SNI) which refines the notion of probing security, by separating between external and internal probes in the circuits. SNI security allows composing masked gadgets since the notion implies that gadgets stop the propagation of dependencies. However, compared to classical probing-secure gadgets, SNI gadgets are usually less efficient than probing-secure ones and require more randomness. Another approach consists in composing SNI gadgets and NI gadgets (a relaxation of the SNI notion) in a careful way to achieve security with better performances (see [14] and references therein). In [19], Cassiers and Standaert introduced the notion of *Probe Isolating Non-Interference* (PINI) that allows secure composition and efficient implementations. It relies on the position of probes in a target implementation. Thanks to this notion, linear functions are directly composable and do not require to be refreshed and non-linear operations remain efficient. A circuit achieves PINI-security (and is consequently probing secure) if all its gadgets are PINI but the notion is not sufficient to achieve region-probing security.

In this paper, we introduce a new composition framework to construct circuits (or masked implementations) satisfying the region probing security notion. For this purpose, we formalize the property of *input-output separation* (IOS) for a refresh gadget and we show that it allows to simply construct region probing circuits from (weaker) probing secure gadgets and in particular more efficient gadgets which are only proven probing-secure but

not SNI, e.g. [29, 11]. We show that this notion can be obtained from uniform SNI or PINI refresh gadgets but also with a simpler design, namely a variant of the refreshing algorithm due to Battistello, Coron, Prouff and Zeitoun [10]. It is worth mentioning that the original refreshing gadget from [10] was proven SNI but for our purposes, we simplify and extend it and show that it achieves our new IOS security notion. The proposed variant can be used to refresh any kind of linear sharing with a quasilinear complexity $\Theta(n \log n)$ and a $O(1/\log n)$ leakage rate (for an n-size sharing).

We then revisit the quasilinear masking scheme of Goudarzi, Joux and Rivain [29] (which we shall call the GJR scheme hereafter). This scheme is based on a polynomial sharing of the form $a = \sum_i a_i \omega^i$, where a is the plain variable and the a_i's are the corresponding shares, and it uses an FFT-based polynomial multiplication to achieve a quasilinear complexity. We describe an improved version of the GJR scheme which works on any base field, including binary fields, and which relies on our composition framework. We further patch a flaw in the original security proof and extend it from the random probing model to the stronger region probing model. Specifically, our improved GJR scheme is secure in the region probing model provided that the underlying FFT algorithm is probing secure. From this ground, we obtain a probing-secure FFT using the approach of [29], that is by relying on a large field $|\mathbb{K}| = \Theta(2^\lambda)$ and taking ω at random. We hence get a region-probing-secure scheme for large fields. For smaller fields, our result is essentially a security reduction from the region probing security of the full scheme to the probing security of the FFT. Finally, we present an application of our extended GJR scheme and compare it with a more standard scheme based on SNI gadgets for two different ciphers: the Advanced Encryption Standard (AES) [1] and MiMC [4]: a cipher with efficient arithmetic representation on a large field. We show that this masking scheme significantly improves the efficiency of the masked cipher for a masking order $n \geq 64$ for MiMC and $n \geq 512$ for the AES. For the AES instantiation, we present a variant of Gao-Mateer additive FFT [27] with improved efficiency and which may be of independent interest.

2 Background on Probing Secure Circuits

2.1 Notations

In this paper, $\mathbb{K}$ shall denote a finite field. Vectors shall be denoted with bold letters, *e.g.* $\boldsymbol{x}$. For any two random variables (or random vectors) X and Y, we shall write $X \overset{\text{id}}{=} Y$ whenever X and Y are identically distributed. For some positive integer $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For any two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{K}^n$, $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ denotes their inner product. For any finite set I, we denote by $|I|$ the cardinality of I. Let $I \subseteq [n]$ and $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{K}^n$, we denote by $\boldsymbol{v}_{|I}$ the $|I|$-tuple $(v_i)_{i \in I}$. We shall denote $x \leftarrow \mathcal{X}$ the action of picking x uniformly at random in some set $\mathcal{X}$, and $y \leftarrow \mathcal{A}(x)$ the action of defining y as the output of an algorithm $\mathcal{A}$ on input x. If $\mathcal{A}$ is a probabilistic algorithm, then $y \leftarrow \mathcal{A}(x)$ is a random assignment of y on input x and for a uniform random tape.

2.2 Basic Definitions

Arithmetic circuits. Given a finite field $\mathbb{K}$, an *arithmetic circuit* is a circuit processing elements of $\mathbb{K}$ through simple arithmetic operations. Formally, it is modeled as a directed acyclic graph whose vertices are *gates* that belong to the following types:

- input gate (fan-in 0, fan-out 1) which holds an input value of the circuit,
- output gate (fan-in 1, fan-out 0) which receives an output value of the circuit,
- constant gate (fan-in 0, fan-out 1) which outputs a constant value of $\mathbb{K}$,

– addition gate (fan-in 2, fan-out 1) which outputs the sum (on $\mathbb{K}$) of the two input values,
– subtraction gate (fan-in 2, fan-out 1) which outputs the difference (on $\mathbb{K}$) of the two input values,
– multiplication gate (fan-in 2, fan-out 1) which outputs the product (on $\mathbb{K}$) of the two input values,
– copy gate (fan-in 1, fan-out 2) which outputs two copies of the input.

The addition, subtraction and multiplication gates are further called *operation gates*. The edges of an arithmetic circuit are called the *wires*. A *randomized arithmetic circuit* is a arithmetic circuit augmented with a

– random gate (fan-in 0, fan-out 1) which outputs a fresh uniform random value of $\mathbb{K}$.

Given some assignment of the input gates, all the wires of a circuit can be assigned subsequently following the input-output behavior of the gates, which finally leads to an assignment of the output gates. For an arithmetic circuit C with n input gates and m output gates, we denote $\boldsymbol{y} = C(\boldsymbol{x}) \in \mathbb{K}^m$ the output of C (*i.e.* the assignment of the output gates of C) on input $\boldsymbol{x} \in \mathbb{K}^n$ (*i.e.* when the input gates are assigned to $\boldsymbol{x}$). For a randomized arithmetic circuit C with q random gates, we denote $\boldsymbol{y} = C^{\boldsymbol{\rho}}(\boldsymbol{x})$ the output of C on input $\boldsymbol{x}$ and such that each random gate outputs a coordinate of $\boldsymbol{\rho} \in \mathbb{K}^q$. The parameter $\boldsymbol{\rho}$ is then called the *random tape* of C. Whenever $\boldsymbol{\rho}$ is omitted, $\boldsymbol{y} = C(\boldsymbol{x})$ denotes the random vector obtained for a uniform distribution of $\boldsymbol{\rho}$.

Let C be a randomized arithmetic circuit with n input gates, m output gates, q random gates, and let consider that the wires of C are labeled from 1 to s (where s is the total number of wires in C). Then for any set $\mathcal{W} \subseteq [s]$ with $|\mathcal{W}| = t$, we shall denote by $C^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x}) \in \mathbb{K}^t$ the tuple composed of the assignments of the wires with labels in $\mathcal{W}$ on input $\boldsymbol{x} \in \mathbb{K}^n$ and random tape $\boldsymbol{\rho} \in \mathbb{K}^q$. In particular, each coordinate of $C^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x})$ is a deterministic function of $\boldsymbol{x}$ and $\boldsymbol{\rho}$. Here again, whenever $\boldsymbol{\rho}$ is omitted, $C_{\mathcal{W}}(\boldsymbol{x})$ denotes the random vector obtained for a uniform distribution of $\boldsymbol{\rho}$ on $\mathbb{K}^q$.

Circuit compilers. We now recall the definition of *circuit compilers* as formalized in [5] (but adapted to arithmetic circuits). We shall call a $\mathbb{K}$-*string* any tuple of elements from the base field $\mathbb{K}$.

Definition 1 (Circuit Compiler)**.** A circuit compiler is a triplet of algorithms $(\mathsf{Compile}, \mathsf{Encode}, \mathsf{Decode})$ defined as follows:

- $\mathsf{Compile}$ (circuit compilation) is a deterministic algorithm that takes as input an arithmetic circuit C and outputs a randomized arithmetic circuit $\widehat{C}$.
- Encode (input encoding) is a probabilistic algorithm that takes as input a $\mathbb{K}$-string $\boldsymbol{x}$ and outputs a $\mathbb{K}$-string $\widehat{\boldsymbol{x}}$.
- Decode (output decoding) is a deterministic algorithm that takes as input a $\mathbb{K}$-string $\widehat{\boldsymbol{y}}$ and outputs a $\mathbb{K}$-string $\boldsymbol{y}$.

These three algorithms satisfy the following properties:

- **Correctness:** For every arithmetic circuit C of input length ℓ, and for every $x \in \mathbb{K}^\ell$, we have
$$\Pr\left(\mathsf{Decode}\big(\widehat{C}(\widehat{\boldsymbol{x}})\big) = C(\boldsymbol{x}) \;\middle|\; \widehat{\boldsymbol{x}} \leftarrow \mathsf{Encode}(\boldsymbol{x})\right) = 1\ ,$$
where $\widehat{C} = \mathsf{Compile}(C)$.
- **Efficiency:** For some parameter called the *encoding order* $n \in \mathbb{N}$, the running time of $\mathsf{Compile}(C)$ is $\mathrm{poly}(n, |C|)$, the running time of $\mathsf{Encode}(\boldsymbol{x})$ is $\mathrm{poly}(n, |\boldsymbol{x}|)$ and the running time of $\mathsf{Decode}\big(\widehat{\boldsymbol{y}}\big)$ is $\mathrm{poly}(n, |\widehat{\boldsymbol{y}}|)$, where $\mathrm{poly}(n, q) = O(n^{k_1} q^{k_2})$ for some constants k_1, k_2.

Sharings and gadgets. Let $n \in \mathbb{N}$ and let $\boldsymbol{v} \in (\mathbb{K}^*)^n$. A $\boldsymbol{v}$-*linear sharing* of $x \in \mathbb{K}$ is a vector $\boldsymbol{x} \in \mathbb{K}^n$ such that $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = x$. The coordinates of a linear sharing $\boldsymbol{x} \in \mathbb{K}^n$ are called

the *shares* of $\boldsymbol{x}$. A random vector $\boldsymbol{x}$ is a *uniform* $\boldsymbol{v}$*-linear sharing* of x if $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = x$ and $\boldsymbol{x}_{|I}$ is uniformly distributed over $\mathbb{K}^t$ for any $I \subset [n]$ with $|I| < n$.

Let $\boldsymbol{v}$-Enc denote a probabilistic algorithm that on input x outputs a uniform $\boldsymbol{v}$-linear sharing of x. For instance $\boldsymbol{v}$-$\mathsf{Enc}(x)$ performs the following:

$$x_1 \leftarrow \mathbb{K};\ x_2 \leftarrow \mathbb{K};\ \cdots\ x_{n-1} \leftarrow \mathbb{K};$$
$$x_n \leftarrow v_n^{-1}\big(x - \langle \boldsymbol{v}, (x_1, \ldots, x_{n-1}, 0) \rangle\big)$$

and returns the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. We further denote $\boldsymbol{v}$-Dec the deterministic algorithm that on input of a $\boldsymbol{v}$-linear sharing of x outputs x. This algorithm simply computes the inner product $\boldsymbol{v}$-$\mathsf{Dec}(\boldsymbol{x}) = \langle \boldsymbol{v}, \boldsymbol{x} \rangle$.

For any operation $g : (x, y) \in \mathbb{K}^2 \mapsto z \in \mathbb{K}$ and for any vector $\boldsymbol{v} \in \mathbb{K}^n$, a $\boldsymbol{v}$*-gadget* of g is a randomized arithmetic circuit with $2n$ input gates and n output gates, which, on input of a $\boldsymbol{v}$-linear sharing of x and a $\boldsymbol{v}$-linear sharing of y, outputs a $\boldsymbol{v}$-linear sharing of $z = g(x, y)$, for any $x, y \in \mathbb{K}$. In particular, G is a $\boldsymbol{v}$*-gadget* of g if and only if for every random tape $\boldsymbol{\rho}$, $\boldsymbol{v}$-$\mathsf{Dec}(G^{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{y})) = g(x, y)$. A $\boldsymbol{v}$*-refresh gadget* is a randomized arithmetic circuit with n input gates and n output gates, which, on input of a $\boldsymbol{v}$-linear sharing of x outputs a $\boldsymbol{v}$-linear sharing of x, for any $x \in \mathbb{K}$.

Standard circuit compilers. Consider a family of vectors $\mathcal{V} = \{\boldsymbol{v}_n \in \mathbb{K}^n\}_{n \in \mathbb{N}}$ and three families of gadgets $\mathcal{G}^{\oplus} = \{G_n^{\oplus}\}_{n \in \mathbb{N}}$, $\mathcal{G}^{\otimes} = \{G_n^{\otimes}\}_{n \in \mathbb{N}}$ and $\mathcal{G}^{\mathsf{R}} = \{G_n^{\mathsf{R}}\}_{n \in \mathbb{N}}$ such that for every $n \in \mathbb{N}$, $G_n^{\oplus}$ is a $\boldsymbol{v}_n$-gadget for the addition on $\mathbb{K}$, $G_n^{\otimes}$ is a $\boldsymbol{v}_n$-gadget for the multiplication on $\mathbb{K}$, and G_n^{R} is a $\boldsymbol{v}_n$-refresh gadget.

The *standard circuit compiler* for $(\mathcal{V}, \mathcal{G}^{\oplus}, \mathcal{G}^{\otimes}, \mathcal{G}^{\mathsf{R}})$ with encoding order n is the circuit compiler for which

- Encode applies $\boldsymbol{v}_n$-Enc to each coordinate of the input $\mathbb{K}$-string;
- Decode applies $\boldsymbol{v}_n$-Dec to each coordinate of the input $\mathbb{K}$-string;
- $\mathsf{Compile}$ takes an arithmetic circuit C and outputs the randomized arithmetic circuit $\widehat{C}$ such that each addition gate is replaced by an addition gadget $G_n^{\oplus}$ followed by a refresh gadget G_n^{R}, each multiplication gate is replaced by a multiplication gadget $G_n^{\otimes}$ followed by a refresh gadget G_n^{R}, each constant gate outputting α is replaced by n constant gates with constants $(\alpha \cdot v_1^{-1}, 0, \ldots, 0)$ followed by a refresh gadget G_n^{R} and each copy gate is replaced by a copy of the input sharing (through n copy gates) followed by a refresh gadget G_n^{R} per output sharing.

It is not hard to see that such a circuit compiler achieves correctness and efficiency, provided, for the latter, that the sizes of the gadgets $G_n^{\oplus}$, $G_n^{\otimes}$ and G_n^{R} are polynomial in n.

To ease the presentation, we restrict the notion of standard circuit compiler to three types of gadgets (addition, multiplication and refresh) but in practice we consider compilers for which the addition gadget is replaced by a broader class of *sharewise* gadgets. These gadgets apply a linear operation (addition, subtraction, multiplication by a constant, or any $\mathbb{K}_0$-linear operation if $\mathbb{K}$ is an $\mathbb{K}_0$-module) sharewisely to the input linear sharing(s).

2.3 Probing Security

Throughout the paper, the notion of *simulator* will refer to a polynomial-time probabilistic algorithm. We will say that a random vector $\boldsymbol{w}$ can be *perfectly simulated* (possibly given some input $\boldsymbol{in}$) if there exists a simulator $\mathcal{S}$ that (given $\boldsymbol{in}$) outputs a vector which is identically distributed as $\boldsymbol{w}$ (over the internal randomness of the simulator), which shall be denoted $\mathcal{S}(\boldsymbol{in}) \overset{\text{id}}{=} \boldsymbol{w}$.

Informally speaking, a randomized arithmetic circuit achieves *t-probing security*, if leaking the value of t arbitrary wires (*i.e.* allowing t *probes* on the circuit) does not reveal any information about the input (provided that the latter has been properly encoded). This is formally define hereafter.

Definition 2 (Probing Security)**.** A randomized arithmetic circuit $\widehat{C}$ is *t-probing secure* w.r.t. an encoding algorithm Encode if for every plain input $\boldsymbol{x}$ and for every set $\mathcal{W} \subseteq [|\widehat{C}|]$, with $|\mathcal{W}| \leq t$, there exists a simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ such that

$$\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) \overset{\text{id}}{=} \widehat{C}_{\mathcal{W}}(\mathsf{Encode}(\boldsymbol{x})) \ .$$

A circuit compiler $(\mathsf{Compile}, \mathsf{Encode}, \mathsf{Decode})$ is said to achieve t-probing security if for every arithmetic circuit C, the randomized arithmetic circuit $\widehat{C} = \mathsf{Compile}(C)$ is t-probing secure w.r.t. Encode. Note that factually, the parameter t is a function of the encoding order n. For instance, the first probing-secure scheme due to Ishai, Sahai and Wagner achieves t-probing security with $t \leq (n-1)/2$ and an efficiency $|\widehat{C}| = \Theta(n^2|C|)$.

Most probing-secure circuit compilers are based on the composition of gadgets. These gadgets are themselves probing-secure w.r.t. the underlying encoding scheme but they must also satisfy *composition* properties so that the overall compiled circuit is probing secure. In particular, the notions of *(strong) non-interference*, or (S)NI and *probe isolating non-interference*, or PINI have been proposed and studied in [8, 14, 7, 19]. In this paper, we introduce another notion called *input-output separation* (see Section 3) which is aimed to enable the composition for a stronger notion of probing security, namely the *region probing security*. In a nutshell, a circuit is *region probing secure* if it is composed of several sub-circuits (*e.g.* several gadgets) that can each tolerate some constant amount of probes (irrespective of the total number of sub-circuits). We shall then consider the *probing rate* (or *leakage rate*) of such a circuit as the maximum ratio between the number of tolerable probes over the size of a sub-circuit. Region probing security is formalized hereafter.

Let us first introduce the notion of circuit partition. For any (randomized) arithmetic circuit C, we call $C \equiv (C_1, C_2, \ldots, C_m)$ a *circuit partition* where each C_i is a sub-circuit of C such that the gates of the C_i's form a partition of the gates of C. We further denote by $\mathcal{W}_{C_i}$ the set of wires with source gate in C_i, so that $\mathcal{W}_{C_1}, \ldots, \mathcal{W}_{C_m}$ is a partition of $[|C|]$.

Definition 3 (Region Probing Security)**.** A randomized circuit $\widehat{C}$ is *r-region probing secure* (*i.e.* with probing rate r) w.r.t. an encoding algorithm Encode if there exists a circuit partition $\widehat{C} \equiv (C_1, C_2, \ldots, C_m)$ such that for every plain input $\boldsymbol{x}$ and for every set $\mathcal{W}_1 \subseteq \mathcal{W}_{C_1}$, $\mathcal{W}_2 \subseteq \mathcal{W}_{C_2}$, $\ldots$, $\mathcal{W}_m \subseteq \mathcal{W}_{C_m}$, with $|\mathcal{W}_i| \leq \lceil r|C_i| \rceil$, there exists a simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ such that

$$\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) \overset{\text{id}}{=} \widehat{C}_{\mathcal{W}}(\mathsf{Encode}(\boldsymbol{x})) \ ,$$

where $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2 \cup \ldots \cup \mathcal{W}_m$. A circuit compiler $(\mathsf{Compile}, \mathsf{Encode}, \mathsf{Decode})$ is *r-region probing secure* if for every circuit C the compiled circuit $\widehat{C} = \mathsf{Compile}(C)$ is r-region probing secure w.r.t. Encode (where r might be a function of the encoding order and the circuit size).

We shall further say that a circuit $\widehat{C}$ is (r, ε)*-region probing secure* (*i.e.* with probing rate r and simulation failure ε), if the simulator fails (*i.e.* returns $\perp$) with probability

$$\Pr\left(\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) = \perp\right) \leq m \cdot \varepsilon \ ,$$

(m being the number of regions) and returns a perfect simulation otherwise:

$$\left(\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) \mid \mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) \neq \perp\right) \overset{\text{id}}{=} \widehat{C}_{\mathcal{W}}(\mathsf{Encode}(\boldsymbol{x})) \ .$$

The region probing security is a relevant security property for a cryptographic implementation while considering side-channel attacks. Indeed, security in the so-called *noisy leakage model* which captures the physical reality of power and electromagnetic side-channel leakages can be reduced to region probing security. These notions and reductions are recalled in Appendix B.

3 Composability from Input-Output Separation

3.1 Input-Output Separation

We introduce hereafter the *input-output separation* security notion for a refresh gadget. Such a property has originally been used in the GJR scheme to achieve composition in the random probing model [29]. We formalize this notion hereafter as general composition property to achieve region probing security. For the sake of simplicity, the definition given in this section only considers refresh gadgets but it can be generalized to any kind of gadgets (see Appendix A for a general definition).

We first introduce the notion of uniformity for a gadget which will be a requirement for our new security notion.

Definition 4 (Uniformity)**.** Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$. A $\boldsymbol{v}$-refresh gadget G is *uniform*, if for every $\boldsymbol{x} \in \mathbb{K}^n$, the output $G(\boldsymbol{x})$ is a uniform $\boldsymbol{v}$-linear sharing of $\langle \boldsymbol{v}, \boldsymbol{x} \rangle$.

In the following, we shall say that a pair of vector $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathbb{K}^n)^2$ is *admissible* for a gadget G if there exists a random tape $\boldsymbol{\rho}$ such that $\boldsymbol{y} = G^{\boldsymbol{\rho}}(\boldsymbol{x})$. For an admissible pair $(\boldsymbol{x}, \boldsymbol{y})$ and a set $\mathcal{W} \subseteq [|G|]$, the wire distribution of G in $\mathcal{W}$ *induced* by $(\boldsymbol{x}, \boldsymbol{y})$, denoted $G_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y})$, is the random vector $G^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x})$, *i.e.* the tuple of wire values for the wire indexes in $\mathcal{W}$, obtained for a uniform drawing of $\boldsymbol{\rho}$ among the set $\{\boldsymbol{\rho} \in \mathbb{K}^q \,;\, G^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x}) = \boldsymbol{y}\}$.

Definition 5 (IOS)**.** Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$ and let G be a $\boldsymbol{v}$-refresh gadget with s wires. G is said *t-input-output separative (t-IOS)*, if it is uniform and if for every admissible pair $(\boldsymbol{x}, \boldsymbol{y})$ and every set of wires $\mathcal{W} \subseteq [s]$ with $|\mathcal{W}| \leq t$, there exists a (two-stage) simulator $\mathcal{S}_{G,\mathcal{W}} = \left(\mathcal{S}^{(1)}_{G,\mathcal{W}}, \mathcal{S}^{(2)}_{G,\mathcal{W}}\right)$ such that

1. $\mathcal{S}^{(1)}_{G,\mathcal{W}}(\perp) = (I, J)$ where $I, J \subseteq [n]$, with $|I| \leq |\mathcal{W}|$ and $|J| \leq |\mathcal{W}|$;
2. $\mathcal{S}^{(2)}_{G,\mathcal{W}}(\boldsymbol{x}_{|I}, \boldsymbol{y}_{|J}) \overset{\text{id}}{=} G_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y})$.

A $\boldsymbol{v}$-refresh gadget is simply said to be *IOS* if it is *n-IOS*.

The above definition generalizes the notion of *input-output linear separability* used in the GJR scheme [29]. Our definition has two differences with the GJR notion:

- the GJR notion requires a deterministic (functional) relation between the probed wires and the input/output shares whereas we only require the ability of simulating the probed wires from some input/output shares;
- the GJR notion requires the knowledge of arbitrary linear combinations of the input/output shares whereas we require the knowledge of some input/output shares.

The first difference makes our definition easier to achieve without impacting the composability. Indeed, in any probing security context, the ability of achieving a perfect simulation is sufficient to prove the security. The second difference makes our definition harder to achieve[1] but more useful to different composition contexts (where the probing security might not rely on linear algebra). Moreover, we describe in Section 4 a refresh gadget achieving our version of input-output separation.

The intuition behind the IOS notion can be understood as follows. Any probing leakage from an IOS refresh gadget can be simulated given a subset of its input shares and output shares. We can therefore reduce the standard region probing security game to a game in which the refresh gadget does not leak anything but its surrounding gadgets leak more. The uniformity property then implies that the leakages from two gadgets separated by a refresh gadget are mutually independent. One can then achieve a perfect simulation of the full leakage through independent simulations of the *separated* leakages from the two gadgets.

[1] A set of shares being a particular case of a set of linear combinations of shares.

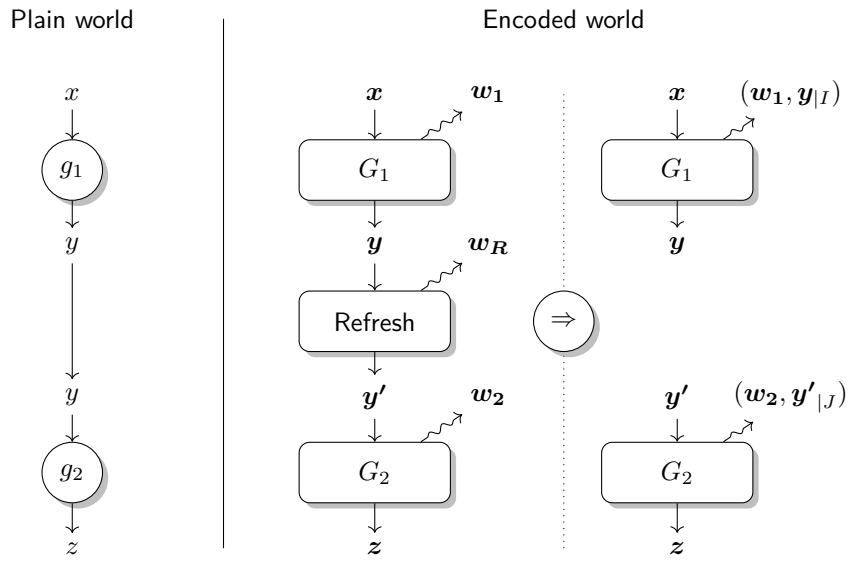

Figure 1: Illustration of the IOS property.

This is illustrated on Figure 1. The full probing leakage $(\boldsymbol{w}_1, \boldsymbol{w}_\mathsf{R}, \boldsymbol{w}_2)$ can be simulated from $(\boldsymbol{w}_1, \boldsymbol{y}_{|I}, \boldsymbol{y}'_{|J}, \boldsymbol{w}_2)$. Moreover, the refresh uniformity implies that, given x, the separated leakages $(\boldsymbol{w}_1, \boldsymbol{y}_{|I})$ and $(\boldsymbol{w}_2, \boldsymbol{y}'_{|J})$ are mutually independent. Therefore, if one can simulate $(\boldsymbol{w}_1, \boldsymbol{y}_{|I})$ on the one hand and $(\boldsymbol{w}_2, \boldsymbol{y}'_{|J})$ on the other hand, then one can simulate the full leakage.

3.2 Composition Theorem

We now provide a formal proof of composition based on the IOS property defined above. Specifically, we show that a standard circuit compiler interleaving operation gadgets and refresh gadgets is region probing secure provided that its operation gadgets are probing secure, and its refresh gadgets are IOS.

As introduced in Section 2, we consider hereafter a family of vectors $\mathcal{V} = \{\boldsymbol{v}_n \in \mathbb{K}^n\}_{n\in\mathbb{N}}$ and three families of gadgets $\mathcal{G}^\oplus = \{G_n^\oplus\}_{n\in\mathbb{N}}$, $\mathcal{G}^\otimes = \{G_n^\otimes\}_{n\in\mathbb{N}}$ and $\mathcal{G}^\mathsf{R} = \{G_n^\mathsf{R}\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$, $G_n^\oplus$ is a $\boldsymbol{v}_n$-gadget for the addition on $\mathbb{K}$, $G_n^\otimes$ is a $\boldsymbol{v}_n$-gadget for the multiplication on $\mathbb{K}$, and G_n^R is a $\boldsymbol{v}_n$-refresh gadget. The following theorem gives our composition result for the standard circuit compiler for $(\mathcal{V}, \mathcal{G}^\oplus, \mathcal{G}^\otimes, \mathcal{G}^\mathsf{R})$.

Theorem 1. *If for every $n \in \mathbb{N}$,*

- *$G_n^\oplus$ is $t_n^\oplus$-probing secure (w.r.t. $\boldsymbol{v}_n$-Enc),*
- *$G_n^\otimes$ is $t_n^\otimes$-probing secure (w.r.t. $\boldsymbol{v}_n$-Enc),*
- *G^{R_n} is t_n^R-IOS,*

then the standard circuit compiler for $(\mathcal{V}, \mathcal{G}^\oplus, \mathcal{G}^\otimes, \mathcal{G}^\mathsf{R})$ is r_n-region probing secure with

$$r_n = \max_{t \le t_n^\mathsf{R}} \min\left(\frac{t_n^\oplus - 3t}{|G_n^\oplus|}, \frac{t_n^\otimes - 3t}{|G_n^\otimes|}, \frac{t}{|G_n^\mathsf{R}|} \right). \tag{1}$$

Proof. Let $n \in \mathbb{N}$ and let $t \le t_n^\mathsf{R}$. Let C be an arithmetic circuit composed of m operation gates, and let $\widehat{C}$ be the randomized arithmetic circuit obtained by calling the standard

circuit compiler for $(\mathcal{V}, \mathcal{G}^{\oplus}, \mathcal{G}^{\otimes}, \mathcal{G}^{\mathsf{R}})$ on C. We shall denote by G_1, G_2, ..., G_m the operation gadgets of $\widehat{C}$ and by G_1^{R}, G_2^{R}, ..., G_m^{R} the refresh gadgets of $\widehat{C}$ where G_i^{R} is placed in output of G_i for every i. We further denote by $\mathcal{W}_{G_i}$ and $\mathcal{W}_{G_i^{\mathsf{R}}}$ the set of wires with source gate in G_i and G_i^{R} respectively. Finally, we denote t_i the integer such that $t_i = t_n^{\oplus} - 3t$ if $G_i = G_n^{\oplus}$ and $t_i = t_n^{\otimes} - 3t$ otherwise (*i.e.* if $G_i = G_n^{\otimes}$) for $i \in \{1, \ldots, m\}$.

Let

$$\mathcal{W} = \bigcup_{i=1}^{m} \mathcal{W}_i \cup \bigcup_{i=1}^{m} \mathcal{W}_i^{\mathsf{R}} \subseteq [|\widehat{C}|]$$

where $\mathcal{W}_i \subseteq \mathcal{W}_{G_i}$, with $\mathcal{W}_i \leq t_i$, and $\mathcal{W}_i^{\mathsf{R}} \subseteq \mathcal{W}_{G_i^{\mathsf{R}}}$, with $\mathcal{W}_i^{\mathsf{R}} \leq t$ for every $i \in [m]$. We will show that for any input $\boldsymbol{in}$ of C, there exists a simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ such that

$$\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) \stackrel{\text{id}}{=} \widehat{C}_{\mathcal{W}}(\mathsf{Encode}(\boldsymbol{in})) \ ,$$

which directly implies the r_n-region probing security of the standard circuit compiler with

$$r_n = \min\Big(\frac{t_n^{\oplus} - 3t}{|G_n^{\oplus}|} \ , \ \frac{t_n^{\otimes} - 3t}{|G_n^{\otimes}|} \ , \ \frac{t}{|G_n^{\mathsf{R}}|} \Big) \ .$$

The above shall hold for every $t \leq t_n^{\mathsf{R}}$ which yields the maximum in (1).

The simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ is simply obtained by running the simulators inherited from the probing security of the G_i's and the IOS property of the G_i^{R}'s. Specifically:

- The IOS property of the G_i^{R}'s implies that, for every $i \in [m]$, there exists a (two-stage) simulator $\mathcal{S}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}} = \big(\mathcal{S}^{(1)}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}}, \mathcal{S}^{(2)}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}}\big)$ such that for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{K}^n \times \mathbb{K}^n$ admissible for G_i^{R}:

  1. $\mathcal{S}^{(1)}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}}(\perp) = (I, J)$ where $I, J \subseteq [n]$, with $|I| \leq |\mathcal{W}|$ and $|J| \leq |\mathcal{W}|$;

  2. $\mathcal{S}^{(2)}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}}(\boldsymbol{x}_{|I}, \boldsymbol{y}_{|J})$ outputs a perfect simulation of $\widehat{C}_{\mathcal{W}_i^{\mathsf{R}}}(\mathsf{Encode}(\boldsymbol{in}))$ given that the pair of input/output sharings of G_i^{R} equals $(\boldsymbol{x}, \boldsymbol{y})$.

  Here $\boldsymbol{x}_{|I}$ corresponds to $|I| \leq t$ (output) wires of the gadget G_i and $\boldsymbol{y}_{|J}$ corresponds to $|J| \leq t$ (input) wires of the gadget G_j subsequent to the refresh G_i^{R}. In particular, there exist two sets $I_i \subseteq \mathcal{W}_{G_i}$ and $J_i \subseteq \mathcal{W}_{G_j}$ such that

  $$\boldsymbol{x}_{|I} = \widehat{C}_{I_i}(\mathsf{Encode}(\boldsymbol{in})) \ \text{ and } \ \boldsymbol{y}_{|J} = \widehat{C}_{J_i}(\mathsf{Encode}(\boldsymbol{in})) \ .$$

- Let ϕ and ψ be the index-mapping functions such that the two input sharings of gadget G_i are output sharings of refresh gadgets $G_{\phi(i)}^{\mathsf{R}}$ and $G_{\psi(i)}^{\mathsf{R}}$. By defining $\overline{\mathcal{W}}_i := \mathcal{W}_i \cup I_i \cup I_{\phi(i)} \cup I_{\psi(i)}$, we get

  $$\bigcup_{i=1}^{m} \overline{\mathcal{W}}_i = \bigcup_{i=1}^{m} \mathcal{W}_i \cup \bigcup_{i=1}^{m} I_i \cup \bigcup_{i=1}^{m} J_i \ \text{ with } \ \overline{\mathcal{W}}_i \subseteq \mathcal{W}_{G_i} \ \text{ and } \ |\overline{\mathcal{W}}_i| \leq t_i + 3t \ .$$

- The probing security of the G_i's implies that, for every $i \in [m]$, there exists a simulator $\mathcal{S}_{G_i,\overline{\mathcal{W}}_i}$ such that

  $$\mathcal{S}_{G_i,\overline{\mathcal{W}}_i}(\perp) \stackrel{\text{id}}{=} \widehat{C}_{\overline{\mathcal{W}}_i}(\mathsf{Encode}(\boldsymbol{in})) \ .$$

We now have all the ingredients to describe the simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$. It proceeds as follows:

1. $\mathcal{S}_{\widehat{C},\mathcal{W}}$ first call the simulators $\mathcal{S}^{(1)}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}}(\perp)$ to get the sets I_i's and J_i's.
2. $\mathcal{S}_{\widehat{C},\mathcal{W}}$ then calls the simulators $\mathcal{S}_{G_i,\overline{\mathcal{W}}_i}(\perp)$ to get tuples

$$\boldsymbol{z}_i = (\boldsymbol{w}_i, \boldsymbol{x}_i, \boldsymbol{y}_{\phi(i)}, \boldsymbol{y}_{\psi(i)}) \overset{\text{id}}{=} \widehat{C}_{\overline{\mathcal{W}}_i}(\mathsf{Encode}(\boldsymbol{in})) \ ,$$

   where $\boldsymbol{w}_i$, $\boldsymbol{x}_i$, $\boldsymbol{y}_{\phi(i)}$, $\boldsymbol{y}_{\psi(i)}$ corresponds to the indexes $\mathcal{W}_i$, I_i, $I_{\phi(i)}$ and $I_{\psi(i)}$ respectively. By the uniformity property of the refresh the distributions $\widehat{C}_{\mathcal{W}_{G_i}}(\mathsf{Encode}(\boldsymbol{in}))$, given $\boldsymbol{in}$, are mutually independent which implies

$$(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m) \overset{\text{id}}{=} \widehat{C}_{\bigcup_i \overline{\mathcal{W}}_i}(\mathsf{Encode}(\boldsymbol{in})) \ .$$

3. $\mathcal{S}_{\widehat{C},\mathcal{W}}$ finally calls the simulators $\mathcal{S}^{(2)}_{G_i^{\mathsf{R}},\mathcal{W}_i^{\mathsf{R}}}$ on inputs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to get tuples $\boldsymbol{w}_i^{\mathsf{R}}$ such that

$$\boldsymbol{w}_i^{\mathsf{R}} \overset{\text{id}}{=} \widehat{C}_{\mathcal{W}_i^{\mathsf{R}}}(\mathsf{Encode}(\boldsymbol{in})) \ ,$$

   and outputs $(\boldsymbol{w}_1, \boldsymbol{w}_1^{\mathsf{R}}, \ldots, \boldsymbol{w}_m, \boldsymbol{w}_m^{\mathsf{R}})$ as simulation.

The IOS property finally implies

$$\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) = (\boldsymbol{w}_1, \boldsymbol{w}_1^{\mathsf{R}}, \ldots, \boldsymbol{w}_m, \boldsymbol{w}_m^{\mathsf{R}}) \overset{\text{id}}{=} \widehat{C}_{\mathcal{W}}(\mathsf{Encode}(\boldsymbol{in})) \ ,$$

which concludes the proof.

□

3.3 Comparison with Non-Interference Security Notions

It is well-known that composition of probing secure gadgets is not always probing secure [23]. Stronger security definitions were previously proposed to analyse the security of large circuits viewed as the composition of simple gadgets. The first such notion, (strong) non-interference, or (S)NI, was proposed in [8]. The notion of *Probe Isolating Non-Interference* (PINI) was also recently introduced in [19]. In this section, we compare our composition approach with the ones underlying the (S)NI and PINI notions and then show some implications between these notions and our IOS notion.

We first recall the (S)NI and PINI definitions while extending them from standard Boolean sharing to the general case of $\boldsymbol{v}$-sharings. For a $\boldsymbol{v}$-refresh gadget, the (S)NI notion is defined as follows:

Definition 6 (NI and SNI)**.** Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$ and let G be a $\boldsymbol{v}$-refresh gadget with s wires. G is said t-Non-Interferent (t-NI) (*resp.* t- Strong Non-Interferent (t-SNI)), if for every $\boldsymbol{x}$ and every set of internal wires $\mathcal{W} \subseteq [s]$ with $|\mathcal{W}| \leq t_1$ and every set of output wires $\mathcal{O} \subseteq [s]$ with $|\mathcal{O}| \leq t_2$ and $t_1 + t_2 \leq t$, there exists a (two-stage) simulator $\mathcal{S}_{G,\mathcal{W},\mathcal{O}} = \left(\mathcal{S}^{(1)}_{G,\mathcal{W},\mathcal{O}}, \mathcal{S}^{(2)}_{G,\mathcal{W},\mathcal{O}}\right)$ such that

1. $\mathcal{S}^{(1)}_{G,\mathcal{W},\mathcal{O}}(\perp) = I$ where $I \subseteq [n]$, with $|I| \leq t_1 + t_2$ (*resp.* with $|I| \leq t_1$);
2. $\mathcal{S}^{(2)}_{G,\mathcal{W},\mathcal{O}}(\boldsymbol{x}_{|I}) \overset{\text{id}}{=} G_{\mathcal{W}\cup\mathcal{O}}(\boldsymbol{x})$.

A $\boldsymbol{v}$-refresh gadget is simply said to be *NI* (*resp. SNI*) if it is $(n-1)$-*NI* (*resp.* $(n-1)$-*SNI*).

If a gadget achieves NI-security, then a probe of an internal wire or an output wire can be simulated using one probe on each of the input sharings of the gadget. If it achieves the stronger SNI-security notion then only probes of internal wires are propagated to inputs (and it thus guarantees independence between the inputs and outputs even with access to the internal wires).

For a $\boldsymbol{v}$-refresh gadget, the PINI notion is defined as follows:

Definition 7 (PINI)**.** Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$ and let G be a $\boldsymbol{v}$-refresh gadget with s wires. G is said t-Probe Isolating Non-Interferent (t-PINI), if for every $\boldsymbol{x}$ and every set of internal wires $\mathcal{W} \subseteq [s]$ with $|\mathcal{W}| \leq t_1$ and every set of output wires $\mathcal{O} \subseteq [s]$ with $|\mathcal{O}| \leq t_2$ and $t_1 + t_2 \leq t$, there exists a (two-stage) simulator $\mathcal{S}_{G,\mathcal{W},\mathcal{O}} = \left(\mathcal{S}^{(1)}_{G,\mathcal{W},\mathcal{O}}, \mathcal{S}^{(2)}_{G,\mathcal{W},\mathcal{O}}\right)$ such that

1. $\mathcal{S}^{(1)}_{G,\mathcal{W},\mathcal{O}}(\perp) = I$ where $I \subseteq [n]$, with $|I| \leq t_1$;
2. $\mathcal{S}^{(2)}_{G,\mathcal{W},\mathcal{O}}(\boldsymbol{x}_{|I \cup J}) \overset{\text{id}}{=} G_{\mathcal{W}\cup\mathcal{O}}(\boldsymbol{x})$;

where $J \subseteq [n]$ is the set of indices of output shares in $\mathcal{O}$. A $\boldsymbol{v}$-refresh gadget is simply said to be *PINI* if it is $(n-1)$-*PINI*.

Comparison of the composition approaches. We discuss hereafter the composition approaches related to the (S)NI notion, the PINI notion and our new IOS notion.

(S)NI composition approach. The NI and SNI notions were proposed in [8] as composition notions for probing-secure gadgets. The authors show how to compose t-NI and t-SNI gadgets to achieve t-probing security, which was further generalized in [14]. Theses results can actually be extended to region probing security. Let us consider the standard circuit compiler as defined in Section 2. If the underlying refresh gadget is SNI and the underlying addition and multiplication gadgets are NI, then it can be checked that the compiled circuit can tolerate up to $t/2$ probes per gadget. In other words, from an SNI refresh gadget, one simply needs NI operation gadgets to obtain a region probing-secure composition.

PINI composition approach. The PINI notion was introduced to allow trivial composition of probing-secure gadgets [19]. Specifically composing any number of PINI gadgets in any way results in a circuit achieving PINI security which further implies probing security. Another advantage of the PINI notion is that it is satisfied by any sharewise gadget (*i.e.* a gadget which simply applies an operation sharewisely) without requiring any refreshing or randomness. Although the PINI notion enables simpler composition, it is limited to probing security (or PINI security) and cannot be extended to region probing security. To illustrate this impossibility, let us consider the following simple example. Suppose that some circuit compiler applies a single-input sharewise gadget G (for instance squaring on $\mathbb{F}_{256}$) successively many times to an input n-sharing $\boldsymbol{x}$. After N gadgets each leaking t probes, all the shares can be recovered whenever $N > n/t$.

IOS composition approach. Our composition approach consists in interleaving an IOS refresh gadget between any pair of successive operation gadgets of the compiled circuit (as in the definition of the standard circuit compiler). Doing so, we can lower the requirement on the operation gadgets: they simply needs to achieve the weaker notion of probing security (see Theorem 1 above).

Comparison. We compare the three composition approaches for the standard circuit compiler as introduced in Section 2. This compiler basically replaces each gate by the corresponding operation gadget and it interleaves a refresh gadget in each connection between two operation gadgets. Assuming that the refresh gadget satisfies a given notion in $\{\text{SNI}, \text{PINI}, \text{IOS}\}$, we look at *(i)* what is the security notion required for the operation gadgets? *(ii)* what is the obtained security notion for the composed circuit?

- **SNI:**
  - *(i)* The **NI** notion is sufficient for the operation gadgets.
  - *(ii)* The composition of NI operation gadgets and SNI refresh gadgets implies the **region probing security** of the composed circuit.
- **PINI:**

(i) The **PINI** notion is sufficient for the operation gadgets.

(ii) The composition of PINI gadgets implies the **probing security** of the composed circuit. Let us stress that with PINI operation gadgets, PINI refresh gadgets are actually useless.

- **IOS:**

(i) The **probing security** is sufficient for the operation gadgets.

(ii) The composition of probing-secure operation gadgets and IOS refresh gadgets implies the **region probing security** of the composed circuit.

Our composition approach, hence achieves the stronger notion of region probing security from the weaker notion of probing security for operation gadgets based on the IOS security of the refresh gadget.

Relations between (S)NI, PINI and IOS. Besides their differences in terms of composition approach, one might question the relation between usual non-interference notions and IOS. Can we show some form of equivalence, one-way implication, or separation? We leave this issue open for further research.

4 An Input-Output Separative Refresh Gadget

Battistello, Coron, Prouff and Zeitoun describe in [9] so-called (template) horizontal side-channel attacks against the ISW [31] and the Rivain-Prouff [41] secure multiplication schemes. These attacks exploit the fact that, for those schemes, the leaking information on each share increases with the number of shares in the presence of a constant leakage rate. Battistello *et al.* describe a variant of the ISW multiplication with probing-security that is heuristically secure against this kind of attacks. In the full version of their paper [10], they further propose a new refreshing gadget with complexity $O(n \log n)$, which we shall refer to as the BPCZ gadget hereafter. In this section, we simplify and extend this gadget for any $\boldsymbol{v}$-linear sharing and we prove that the obtained variant achieves the IOS security notion.

4.1 Refresh Gadget Description

Starting from the BPCZ gadget, our approach consists in

- using a single (post-processing) randomization layer instead of two (pre-processing and post-processing) in the algorithm recursion,
- introducing necessary multiplication by constants to support $\boldsymbol{v}$-linear sharings,
- calling the obtained variant of the BPCZ gadget to generate a fresh sharing of 0 which is then used to refresh the input sharing by addition.

The procedure ZeroEncoding which generates a fresh $\boldsymbol{v}$-linear sharing of 0 is described in Algorithm 1. It is defined recursively: for $n = 2$, it outputs $\boldsymbol{y} = (z_1, z_2) = (r, -(v_1 v_2^{-1}) \cdot r)$ such that $\langle \boldsymbol{z}, \boldsymbol{v} \rangle = 0$. For $n \geq 4$ a power of 2, ZeroEncoding is called recursively to produce two halves of the sharing (Steps 4-5) and a post-processing layer is applied to the whole sharing (Steps 6-9). Note that the original refresh gadget proposed in [9] makes use of an additional and similar pre-processing layer before the two recursive calls. It results that our variant is twice more efficient in terms of computation and randomness generation.

Algorithm 1 ZeroEncoding

Require: $\boldsymbol{v} = (v_1, \ldots, v_n)$
Ensure: $\boldsymbol{z} = (z_1, \ldots, z_n)$ such that $\langle \boldsymbol{z}, \boldsymbol{v} \rangle = 0$
1: **if** $n = 2$ **then**
2: $\quad r \leftarrow \mathbb{K}$
3: $\quad$ **return** $(r, -(v_1 v_2^{-1}) \cdot r)$
4: $(s_1, \ldots, s_{\frac{n}{2}}) \leftarrow \mathsf{ZeroEncoding}(v_1, \ldots, v_{\frac{n}{2}})$ ▷ Recursive call
5: $(s_{\frac{n}{2}+1}, \ldots, s_n) \leftarrow \mathsf{ZeroEncoding}(v_{\frac{n}{2}+1}, \ldots, v_n)$ ▷ Recursive call
6: **for** $i = 1, \ldots, \frac{n}{2}$ **do**
7: $\quad r_i \leftarrow \mathbb{K}$
8: $\quad z_i \leftarrow s_i + r_i$
9: $\quad z_{i+\frac{n}{2}} \leftarrow s_{i+\frac{n}{2}} - (v_i v_{i+\frac{n}{2}}^{-1}) \cdot r_i$

Algorithm 2 RefreshGadget

Require: $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{v} = (v_1, \ldots, v_n)$,
Ensure: $\boldsymbol{y} = (y_1, \ldots, y_n)$ such that $\langle \boldsymbol{y}, \boldsymbol{v} \rangle = \langle \boldsymbol{x}, \boldsymbol{v} \rangle$
1: $\boldsymbol{z} \leftarrow \mathsf{ZeroEncoding}(\boldsymbol{v})$
2: $\boldsymbol{y} \leftarrow \boldsymbol{x} + \boldsymbol{z}$

Let us denote $R(n)$, $A(n)$ and $M(n)$ the randomness complexity, the number of additions and the number of scalar multiplications of the ZeroEncoding algorithm for length-n linear sharing. We have $R(2) = 1$, $A(2) = 0$ and $M(2) = 1$ and $R(n) = 2R(\frac{n}{2}) + \frac{n}{2}$, $A(n) = 2A(\frac{n}{2}) + n$ and $M(n) = 2M(\frac{n}{2}) + \frac{n}{2}$ for all $n \geq 2$. By induction, we thus have for any $n \geq 2$, a power of 2,

$$R(n) = M(n) = \frac{n}{2}\log(n) \quad \text{and} \quad A(n) = n \log \frac{n}{2} \ . \tag{2}$$

We have n further additions in RefreshGadget.

Remark 1. In the original version of this paper, we suggest to directly apply the BPCZ variant (without pre-processing layer) to the input sharing $\boldsymbol{x}$ and provide a proof of IOS for this refresh gadget. However, a flaw in this proof was reported to us by Gaëtan Cassiers. This flaw is solved while using the BPCZ variant to generate a sharing of 0 which is then added to the input sharing $\boldsymbol{x}$. We note that the obtained refresh gadget (BPCZ without pre-processing layer, generating a sharing of 0) was also considered by Mathieu-Mahias in [34]. The author shows that this gadget achieves the SNI property.

4.2 Proof of Input-Output Separation

Theorem 2. *The refresh gadget from Algorithm 2 is input-output separative.*

Proof. Throughout the proof, we denote by $L = \left[\frac{n}{2}\right]$ and $H = [n] \setminus L$.

Uniformity. Let $\boldsymbol{v} \in (\mathbb{K}^*)^n$. We show that RefreshGadget is *uniform*, namely that if for every $\boldsymbol{x} \in \mathbb{K}^n$, the output of $\mathsf{RefreshGadget}(\boldsymbol{x}, \boldsymbol{v})$ is a uniform $\boldsymbol{v}$-linear sharing of $\langle \boldsymbol{v}, \boldsymbol{x} \rangle$. For this purpose, we simply need to show that $\mathsf{ZeroEncoding}(\boldsymbol{v})$ outputs a uniform $\boldsymbol{v}$-linear sharing of 0. The proof is by induction on n.

For $n = 2$, given $\boldsymbol{v} = (v_1, v_2)$, ZeroEncoding outputs $\boldsymbol{z} = (r, -(v_1 v_2^{-1}) \cdot r)$ where r is picked uniformly at random in $\mathbb{K}$. This is clearly a uniform sharing satisfying $\langle \boldsymbol{v}, \boldsymbol{z} \rangle$. For $n \geq 4$, ZeroEncoding first computes $(s_1, \ldots, s_{n/2})$ and $(s_{n/2+1}, \ldots, s_n)$ as outputs of $\mathsf{ZeroEncoding}(v_1, \ldots, v_{n/2})$ and $\mathsf{ZeroEncoding}(v_{n/2+1}, \ldots, v_n)$. By the induction hypothesis, $\boldsymbol{s}_{|L} = (s_1, \ldots, s_{n/2})$ and $\boldsymbol{s}_{|H} = (s_{n/2+1}, \ldots, s_n)$ are uniform and independent $\boldsymbol{v}_{|L} =$

$(v_1, \ldots, v_{n/2})$-linear sharing and $\boldsymbol{v}_{|H} = (v_{n/2+1}, \ldots, v_n)$-linear sharing of 0. ZeroEncoding then picks uniformly at random r_i in $\mathbb{K}$ for $i \in L$ and sets $z_i = s_i + r_i$ and $z_{i+n/2} = s_{i+n/2} - (v_i v_{i+n/2}^{-1}) \cdot r_i$ for $i \in L$. Denoting $r'_i = s_i + r_i (= z_i)$ for $i \in L$, the vector $(r'_1, \ldots, r'_{n/2})$ is uniformly distributed in $\mathbb{K}^{n/2}$ and we have $z_{i+n/2} = s_{i+n/2} - (r'_i - s_i) \cdot v_i \cdot v_{i+n/2}^{-1}$ for $i \in L$ where $\boldsymbol{s}_{|L}$ and $\boldsymbol{s}_{|H}$ are uniform and independent $\boldsymbol{v}_{|L}$-linear sharing and $\boldsymbol{v}_{|H}$-linear sharing of 0. We obtain that $\boldsymbol{z}$ is uniformly distributed among the vectors of $\mathbb{K}^n$ satisfying $\langle \boldsymbol{v}, \boldsymbol{z} \rangle = 0$, namely $\boldsymbol{z}$ is a uniform $\boldsymbol{v}$-linear sharing of 0.

IOS. For the sake of simplicity, we show the IOS property for the particular case of $\boldsymbol{v} = (1, 1, \ldots, 1)$. This way we can ignore the multiplications by constant factors from the vector coefficients. The argument applies in the exact same way (but with heavier notations) for the general case.

Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ denote some random vectors. In the scope of this proof, we shall say that a random vector $\boldsymbol{x} \in \mathbb{K}^\ell$ is *ℓ-free* with respect to $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$, if any $(\ell - 1)$-subtuple of $\boldsymbol{x}$ is uniformly distributed on $\mathbb{K}^{\ell-1}$ and mutually independent of the joint distribution of $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m)$. The core of the proof consists in showing the following property of the ZeroEncoding gadget:

> *For every set $\mathcal{W}$ of probed wires in ZeroEncoding, with $|\mathcal{W}| = t$, and denoting $\boldsymbol{w}$ the corresponding wire distribution, there exists a set K, with $\ell := |K| = \max(n - t, 0)$, such that $\boldsymbol{z}_{|K}$ is ℓ-free w.r.t. $\boldsymbol{z}_{|[n] \notin K}$ and $\boldsymbol{w}$.* (3)

Property (3) is direct for $n = 2$. If $t = 0$, it holds from the uniformity of the sharing produced by ZeroEncoding, while for $t \geq 1$, the property always holds for the case $n = 2$. Let us now show this property by induction: we assume that it holds for $n/2$ and show that it then holds for n.

We denote ZE_1 the gadget corresponding to the first recursive call to ZeroEncoding (Step 4), ZE_2 the gadget corresponding to the second recursive call to ZeroEncoding (Step 5) and M the gadget corresponding to the post-processing layer (Steps 6-9). We denote by $\mathcal{W}_1$, $\mathcal{W}_2$, and $\mathcal{W}_3$, the subset of $\mathcal{W}$ corresponding to wire indexes from ZE_1, ZE_2, and M respectively, so that $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2 \cup \mathcal{W}_3$. Without loss of generality, all the outputs of ZE_1 and ZE_2, which are also inputs of M, are included to $\mathcal{W}_1$ and $\mathcal{W}_2$, but not to $\mathcal{W}_3$. We denote $t_1 = |\mathcal{W}_1|$, $t_2 = |\mathcal{W}_2|$, and $t_3 = |\mathcal{W}_3|$, and consider the case

$$t_1 + t_2 + t_3 = |\mathcal{W}| < n$$

(since for $|\mathcal{W}| \geq n$ the property is trivial). We denote $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, and $\boldsymbol{w}_3$, the wire distributions corresponding to $\mathcal{W}_1$, $\mathcal{W}_2$, and $\mathcal{W}_3$ so that

$$(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3) = \boldsymbol{w} \ .$$

As in Algorithm 1, $\boldsymbol{s} = (s_1, \ldots, s_n)$ denotes the linear sharing in output of the block $(\mathsf{ZE}_1 \parallel \mathsf{ZE}_2)$, and $\boldsymbol{s}_{|L} = (s_1, \ldots, s_{\frac{n}{2}})$ and $\boldsymbol{s}_{|H} = (s_{\frac{n}{2}+1}, \ldots, s_n)$ are the respective output of ZE_1 and ZE_2. These notations are illustrated on Figure 2.

Without loss of generality, we assume $t_1 \leq t_2$. Applying Property (3) to ZE_1 (which holds for $n/2$ by assumption), we obtain that there exists a set $K \subseteq L$, with $\ell := |K| = \frac{n}{2} - t_1$, such that $\boldsymbol{s}_{|K}$ is ℓ-free w.r.t. $\boldsymbol{s}_{|L \setminus K}$ and $\boldsymbol{w_1}$. Without loss of generality, we further assume that $K = [\ell]$ (this does not change the argument but eases the notations).

Let us define K' as the sumset

$$K' = K + \{0, \frac{n}{2}\} = \{1, \ldots, \ell, \frac{n}{2} + 1, \ldots, \frac{n}{2} + \ell\} \ .$$

We show hereafter that $\boldsymbol{z}_{|K'}$ is (2ℓ)-free w.r.t. $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{s}_{|[n] \setminus K}$ and $\boldsymbol{z}_{|[n] \setminus K'}$. We denote by c_0, c_1, $\ldots$, c_ℓ some coefficients solely depending on $\boldsymbol{s}_{|[n] \setminus K}$ (those can be thought of

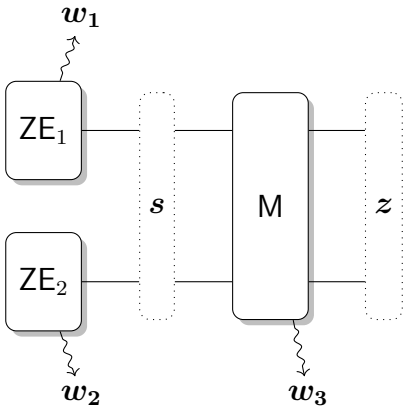

Figure 2: IOS refresh gadget with probes $(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3)$.

as constants for a given assignment of $\boldsymbol{s}_{|[n]\setminus K}$). Since the randomness of ZE_1 and the randomness of ZE_2 are mutually independent, $\boldsymbol{s}_{|K}$ can be seen as a random vector

$$\boldsymbol{s}_{|K} = (s_1, \ldots, s_{\ell-1}, s_\ell)$$

with $(s_1, \ldots, s_{\ell-1})$ uniformly distributed on $\mathbb{K}^{\ell-1}$ and mutually independent of $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{s}_{|[n]\setminus K}$, and

$$s_\ell = c_0 - \sum_{i=1}^{\ell-1} s_i \ .$$

Then, $\boldsymbol{z}_{|K'}$ can be expressed as

$$\boldsymbol{z}_{|K'} = \Big(s_1 + r_1, \ \ldots, \ s_{\ell-1} + r_{\ell-1}, \ c_0 + r_\ell - \sum_{i=1}^{\ell-1} s_i, \ c_1 - r_1, \ \ldots, \ c_\ell - r_\ell\Big) \tag{4}$$

where $r_1, \ldots, r_\ell, s_1, \ldots, s_{\ell-1}$ are fresh uniform random variables, mutually independent of $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{s}_{|[n]\setminus K}$ and $\boldsymbol{z}_{|[n]\setminus K'}$ (where we recall that the c_i's are coefficients which solely depend on $\boldsymbol{s}_{|[n]\setminus K}$ and where we ignore the constant factors from $\boldsymbol{v}$ for the sake of simplicity). By defining $u_1 := s_1 + r_1, \ldots, u_{\ell-1} := s_{\ell-1} + r_{\ell-1}, u_\ell := c_1 - r_1, \ldots, u_{2\ell-1} := c_\ell - r_\ell$, Equation 4 rewrites as

$$\boldsymbol{z}_{|K'} = \Big(u_1, \ \ldots, \ u_{\ell-1}, \ \sum_{i=0}^{\ell} c_i - \sum_{i=1}^{2\ell-1} u_i, \ u_\ell, \ \ldots, \ u_{2\ell-1}\Big) \tag{5}$$

with $u_1, \ \ldots, \ u_{2\ell-1}$, fresh uniform random variables, mutually independent of $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{s}_{|[n]\setminus K}$ and $\boldsymbol{z}_{|[n]\setminus K'}$. We thus get that $\boldsymbol{z}_{|K'}$ is (2ℓ)-free w.r.t. $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{s}_{|[n]\setminus K}$ and $\boldsymbol{z}_{|[n]\setminus K'}$.

We shall now explain how to update the set K' to take into account the probes from $\mathcal{W}_3$, *i.e.* while ensuring that $\boldsymbol{z}_{|K'}$ is further free w.r.t. $\boldsymbol{w}_3$. Each variable in $\boldsymbol{w}_3$ is either a random r_i or an output share z_i. For each r_i in $\boldsymbol{w}_3$, with $i \in K$, we remove $\frac{n}{2} + 1$ from K'. This amounts to removing the coordinates $c_i - r_i$ from $\boldsymbol{z}_{|K'}$ (see Equation 4). One can check that doing so, and treating r_i as a "constant" c'_i, a change of variables still yields an expression like Equation 5, where the random u_i's are independent of $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{s}_{|[n]\setminus K}$, $\boldsymbol{z}_{|[n]\setminus K'}$ and the probed r_i's from $\boldsymbol{w}_3$. For each z_i in $\boldsymbol{w}_3$, with $i \in K'$, we further remove i from K'. This amounts to removing z_i from $\boldsymbol{z}_{|K'}$. From Equation 5, it is clear that removing one of the coordinates does not change the form of the distribution. Moreover the subtuples of $\boldsymbol{z}_{|K''}$, where K'' denotes the updated set, are mutually independent of

the removed coordinates z_i. We hence get that the updated tuple $\boldsymbol{z}_{|K''}$ is $|K''|$-free w.r.t. $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{z}_{|[n]\setminus K''}$ and $\boldsymbol{w}_3$. Moreover, the update process removes at most t_3 elements from K', which implies

$$|K''| \geq |K'| - t_3 = 2|K| - t_3 = n - 2t_1 - t_3 \geq n - (t_1 + t_2 + t_3) \ . \tag{6}$$

In case the last inequality is strict, one can remove additional coordinates from K'' to get $|K''| = n - (t_1 + t_2 + t_3)$. We conclude that Property (3) is satisfied for n.

It remains to show that Property (3) for ZeroEncoding implies IOS for RefreshGadget. We consider a set of probed wires $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2 \cup \mathcal{W}_3 \cup \mathcal{W}_4$, where $\mathcal{W}_1$, $\mathcal{W}_2$, and $\mathcal{W}_3$ are the three sets of wires from ZeroEncoding as considered above, and $\mathcal{W}_4$ are wires from RefreshGadget (excluding ZeroEncoding). The corresponding wire distribution $\boldsymbol{w}_4$ only contains input shares x_i or output shares y_i (since the z_i coordinates are included to $\boldsymbol{w}_3$ w.l.o.g.). Let us denote $\mathcal{W}_3'$ the wires corresponding to the z_i's for indexes i such that x_i or y_i is in $\boldsymbol{w}_4$. The IOS sets $(I, J) = \mathcal{S}^{(1)}_{G,\mathcal{W}}(\perp)$ are defined as $I = J = [n] \setminus K$ where K is the set obtained from Property (3) with probing set $\mathcal{W}_1 \cup \mathcal{W}_2 \cup \mathcal{W}_3 \cup \mathcal{W}_3'$.

We now explain how to perform a perfect simulation $\mathcal{S}^{(2)}_{G,\mathcal{W}}(\boldsymbol{x}_{|I}, \boldsymbol{y}_{|J}) \overset{\text{id}}{=} G_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y})$, *i.e.* how to perfectly simulate

$$\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4) \sim G_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y})$$

for any admissible pair of sharings $(\boldsymbol{x}, \boldsymbol{y})$. We first note that for any x_i or y_i in $\boldsymbol{w}_4$, the corresponding z_i is included to $\boldsymbol{w}_3'$ (from set $\mathcal{W}_3'$), hence by construction the index i is excluded from the set K and is thus included in the sets I and J. We can then trivially simulate all the probed input / output shares, *i.e.* the coordinates of $\boldsymbol{w}_4$. We can further perfectly simulate

$$\boldsymbol{z}_{|[n]\setminus K} = \boldsymbol{z}_{|I} = \boldsymbol{y}_{|I} - \boldsymbol{x}_{|I} \tag{7}$$

from the input shares $\boldsymbol{x}_{|I}$ and the output shares $\boldsymbol{y}_{|J}$.

Now by definition of ZeroEncoding all the wire values are defined as linear combinations of random elements sampled from $\mathbb{K}$ (Steps 2 and 7 of Algorithm 1). We can then write

$$(\boldsymbol{z}_{|[n]\setminus K} \,|\, \boldsymbol{w_1} \,|\, \boldsymbol{w_2} \,|\, \boldsymbol{w_3}) = \boldsymbol{r} \cdot M \ , \tag{8}$$

where $\boldsymbol{r}$ denotes the vector of randomly sampled elements from $\mathbb{K}$ and M is a matrix with coefficients in $\mathbb{K}$. We can perfectly simulate $\boldsymbol{w_1}$, $\boldsymbol{w_2}$ and $\boldsymbol{w_3}$ by picking a random $\boldsymbol{r}$ for which Equation 8 matches the simulated $\boldsymbol{z}_{|[n]\setminus K}$ from Equation 7. Now according to Property (3), any subtuple of $\boldsymbol{z}_K$ is mutually independent of the above simulation, while the last degree of freedom is defined such that $\boldsymbol{z}$ is a sharing of 0. This ensures that the above simulation is consistent with any value of $\boldsymbol{z}_{|K} := \boldsymbol{y}_{|K} - \boldsymbol{x}_{|K}$ for an admissible pair of sharings $(\boldsymbol{x}, \boldsymbol{y})$. □

5 Revisiting the GJR Masking Scheme

In this section, we revisit the quasilinear-complexity Goudarzi-Joux-Rivain (GJR) masking scheme [29]. We first describe a variant of this scheme making use of the IOS refresh gadget described above and which is more general than the original scheme in the sense that it works on any base field $\mathbb{K}$ equipped with an Fast Fourier Transform (FFT) for multiple-point polynomial evaluation. We then show that the use of our refresh allows to patch a flaw in the security proof of the original scheme. We shall refer to the improved GJR scheme as the GJR$^+$ scheme hereafter.

5.1 The GJR$^+$ Scheme

As in the original scheme, the GJR$^+$ scheme is based on so called ω-encodings which are $\boldsymbol{v}_\omega$-linear sharings with

$$\boldsymbol{v}_\omega = (1, \omega, \ldots, \omega^{n-1}) \ . \tag{9}$$

For such a vector, a sharing $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ of a plain value $x \in \mathbb{K}$ can be seen as the coefficients of a polynomials $P_{\boldsymbol{x}} = \sum_{i=1}^{n} x_i \cdot \alpha^{i-1} \in \mathbb{K}[\alpha]$ such that $P_{\boldsymbol{x}}(\omega) = x$. The quasilinear complexity can then be achieved by using efficient FFT-based multiplication for the multiplication gadget. Note that such encoding is close to but different from *Shamir's secret sharing* [42]. In the latter the shares are defined as evaluations of a polynomial in fixed points and for which the plain value is the degree-0 coefficient.

We assume the existence of a Fast Fourier Transform (FFT) algorithm that, given any polynomial $P \in \mathbb{K}[\alpha]$ of degree $< 2n$, maps the coefficients of P to the evaluations of P over $2n$ points of $\mathbb{K}$, with a complexity of $\widetilde{O}(n)$ operations. That is:

$$\mathsf{FFT}_{\boldsymbol{\alpha}} : (x_1, x_2, \ldots, x_{2n}) \mapsto (u_1, u_2, \ldots, u_{2n}) \quad \text{with} \quad u_j = \sum_{i=1}^{2n} x_i \cdot \alpha_j^{i-1}$$

for every $j \in [2n]$, for some $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{2n}) \in \mathbb{K}^{2n}$. We further assume that this FFT algorithm can be written as an arithmetic circuit on $\mathbb{K}$ solely composed of additions, subtractions and multiplication by constants in $\mathbb{K}$, and that it features an inverse FFT algorithm with the same properties (in terms of type and number of operations).

The GJR$^+$ scheme is a standard circuit compiler for $(\mathcal{V}, \mathcal{G}^{\oplus}, \mathcal{G}^{\otimes}, \mathcal{G}^{\mathsf{R}})$ (see definition in Section 2), with $\mathcal{V} = \left\{\boldsymbol{v}_\omega^{(n)}\right\}_{n \in \mathbb{N}}$ where $\boldsymbol{v}_\omega^{(n)} \in \mathbb{K}^n$ is the vector defined in (9). As in the original scheme, we assume in the following that the order n is a power of two. The scheme could be easily extended to deal with non-power of two at the cost of a small constant efficiency factor.

We now give the description of the associated $\boldsymbol{v}_\omega^{(n)}$-gadgets. For the sake of clarity we shall omit the superscript and simply note $\boldsymbol{v}_\omega$ in what follows.

Refresh Gadget. We use the refresh gadget of Section 4 (see Algorithm 2) for $\boldsymbol{v}_\omega$-sharings *i.e.* with encoding vector $\boldsymbol{v}$ assigned to $\boldsymbol{v}_\omega$. This refresh gadget is applied in output of each operation gadget (in accordance to the definition of the standard circuit compiler). We recall that this gadget achieves the uniformity and IOS properties defined in Subsection 3.1.

Addition Gadget. Given two $\boldsymbol{v}_\omega$-sharings $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, the addition gadget outputs

$$\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, \ldots, x_n + y_n) \ .$$

This is done *via* n addition gates processing each share separately. Hence this addition gadget achieves $(n-1)$-probing security.

Subtraction Gadget. Given two $\boldsymbol{v}_\omega$-sharings $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, the subtraction gadget outputs

$$\boldsymbol{x} - \boldsymbol{y} = (x_1 - y_1, \ldots, x_n - y_n) \ .$$

This is done *via* n subtraction gates processing each share separately. Hence this subtraction gadget achieves $(n-1)$-probing security.

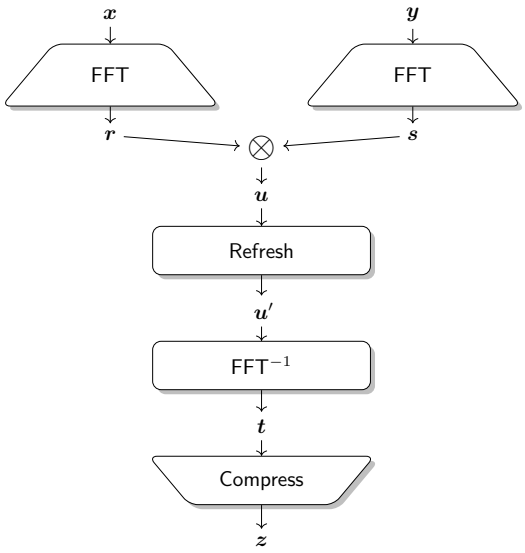

Figure 3: Multiplication gadget.

Multiplication Gadget. Let $\boldsymbol{v}'_\omega \in \mathbb{K}^{2n}$ be the vector defined as

$$\boldsymbol{v}'_\omega = \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}(1, \omega, \omega^2, \ldots, \omega^{2n-1}) \ .$$

Let $\mathsf{Compress}$ be the $\mathbb{K} \times \mathbb{K}^{2n} \to \mathbb{K}^n$ mapping defined as

$$\mathsf{Compress}(\omega; t_1, t_2, \ldots, t_{2n}) = (t_1 + \omega^n \cdot t_{n+1},\ t_2 + \omega^n \cdot t_{n+2},\ \ldots,\ t_n + \omega^n \cdot t_{2n})$$

Let $\mathbf{0}$ denotes the n-dimensional all-0 vector and $\|$ denote the concatenation operator. Given two $\boldsymbol{v}_\omega$-sharings $\boldsymbol{x}$ and $\boldsymbol{y}$, the multiplication gadget proceeds as

1. $\boldsymbol{r} \leftarrow \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{x} \,\|\, \mathbf{0})$
2. $\boldsymbol{s} \leftarrow \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{y} \,\|\, \mathbf{0})$
3. $\boldsymbol{u} \leftarrow \boldsymbol{r} \cdot \boldsymbol{s}$
4. $\boldsymbol{u}' \leftarrow \mathsf{Refresh}(\boldsymbol{v}'_\omega; \boldsymbol{u})$
5. $\boldsymbol{t} \leftarrow \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{u}')$
6. $\boldsymbol{z} \leftarrow \mathsf{Compress}(\omega; \boldsymbol{t})$

and outputs $\boldsymbol{z}$. Note that $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{u}$, $\boldsymbol{u}'$, $\boldsymbol{t}$ are $(2n)$-dimensional vectors. Only the input/output sharings $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are n-dimensional vectors. The procedure is depicted on Figure 3 for illustration.

Remark. This multiplication gadget is similar to the GJR multiplication gadget but we introduce a refreshing in Step 4. This refreshing is done using Algorithm 2 (see Section 4) where the encoding vector $\boldsymbol{v}'_\omega$ and the input sharing are of size $2n$.

Correctness. Let x and y be the values encoded by $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively and let $P_{\boldsymbol{x}} \in \mathbb{K}[\alpha]$ and $P_{\boldsymbol{y}} \in \mathbb{K}[\alpha]$ be the degree-$(n-1)$ polynomials whose coefficients are the coordinates of $\boldsymbol{x}$ and $\boldsymbol{y}$, so that we have $P_{\boldsymbol{x}}(\omega) = x$ and $P_{\boldsymbol{y}}(\omega) = y$.

Let us first assume that Step 4 applies an identity mapping, *i.e.* $\boldsymbol{u}' = \boldsymbol{u}$. Then Steps 1–5 perform a classical FFT-based polynomial multiplication. Namely, the coordinates of $\boldsymbol{t}$ are the coefficients of the polynomial $P_{\boldsymbol{t}} \in \mathbb{K}[\alpha]$ such that $P_{\boldsymbol{t}}(\alpha) = P_{\boldsymbol{x}}(\alpha) \cdot P_{\boldsymbol{y}}(\alpha)$, and in

particular $P_{\boldsymbol{t}}(\omega) = x \cdot y$. Then Step 6 outputs a vector $\boldsymbol{z}$ such that $\langle \boldsymbol{v}_\omega, \boldsymbol{z} \rangle = P_{\boldsymbol{t}}(\omega) = x \cdot y$, *i.e.* a $\boldsymbol{v}_\omega$-sharing of $x \cdot y$.

Let $\boldsymbol{v}''_\omega = (1, \omega, \omega^2, \ldots, \omega^{2n-1})$, then we have

$$P_{\boldsymbol{t}}(\omega) = \langle \boldsymbol{v}''_\omega, \boldsymbol{t} \rangle = x \cdot y \quad \Leftrightarrow \quad \langle \boldsymbol{v}'_\omega, \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{t}) \rangle = x \cdot y \ .$$

By correctness of the FFT-based polynomial multiplication, we hence have that $\boldsymbol{u} = \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{t})$ is a $\boldsymbol{v}'_\omega$-sharing of $x \cdot y$. Let us now consider the actual multiplication gadget with refreshing at Step 4. By correctness of the refresh algorithm, $\boldsymbol{u}'$ is also a $\boldsymbol{v}'_\omega$-sharing of $x \cdot y$, and by the above relation we have that $\langle \boldsymbol{v}'_\omega, \boldsymbol{u}' \rangle = x \cdot y$ implies $\langle \boldsymbol{v}''_\omega, \mathsf{FFT}^{-1}_{\boldsymbol{\alpha}}(\boldsymbol{u}') \rangle = x \cdot y$, which is $\langle \boldsymbol{v}''_\omega, \boldsymbol{t} \rangle = x \cdot y$. We hence get the correctness of the multiplication gadget.

Scalar Multiplication Gadget. For the particular case of a multiplication by a constant, a dedicated scalar multiplication gadget can be used which is much more efficient than a regular multiplication gadget. Given a $\boldsymbol{v}_\omega$-sharing $\boldsymbol{x} = (x_1, \ldots, x_n)$ and a constant $\alpha \in \mathbb{K}$, the scalar multiplication gadget outputs

$$\alpha \cdot \boldsymbol{x} = (\alpha \cdot x_1, \ldots, \alpha \cdot x_n) \ .$$

This is done *via* n multiplication gates processing each share separately. Hence this scalar multiplication gadget achieves $(n-1)$-probing security.

Square Gadget. For the particular case of a field $\mathbb{K}$ of characteristic 2, a square can be computed through a dedicated gadget much more efficiently than with a regular multiplication gadget. Given a $\boldsymbol{v}_\omega$-sharing $\boldsymbol{x} = (x_1, \ldots, x_n)$ of x, the square gadget outputs

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \quad \text{with} \quad y_i = x_i^2 \cdot \omega^{i-1}$$

for every $i \in [n]$. We then have $\langle \boldsymbol{v}_\omega, \boldsymbol{y} \rangle = \langle \boldsymbol{v}_\omega, \boldsymbol{y} \rangle^2$ by linearity of the squaring on a field of characteristic 2, which implies that $\boldsymbol{y}$ is indeed a $\boldsymbol{v}_\omega$-sharing of x^2. The square gadget involves $2n$ multiplication gates processing each share separately. Hence this square gadget achieves $(n-1)$-probing security. More generally, a sharewise gadget can compute any q^k-th power on a field of characteristic q (*i.e.* compute the k-th Frobenius map).

Note that, extending the standard circuit compiler to include such gadget is straightforward but it would make the formalism heavier so we skip this extension from our presentation.

5.2 Field Extension and FFT Algorithm

In order to instantiate the GJR$^+$ scheme, it is necessary to consider an implementation of secure multiplication at order n over a finite field $\mathbb{K}$ and an element ω such that there exists an FFT algorithm which allows quasilinear multiplication of polynomials of degree at most n and coefficients in $\mathbb{K}$ and which can be written as an arithmetic circuit on $\mathbb{K}$ solely composed of additions, subtractions and multiplication by constants.

A possible approach (which was used in [29]), is to consider finite fields $\mathbb{K} = \mathbb{F}_q$ that contain a $(2n)$-th root of unity ω (*i.e.* such that $2n \mid q-1$). However, most of the time, we cannot choose the underlying algebraic structure and we have to consider a specific cryptographic primitive with a given structure and to implement it securely. In order to extend the original scheme to any finite field $\mathbb{F}_p^m$ for some prime number p (with $m \geq 1$), we can use the general *additive* FFT proposed by Cantor in [45, 17]. In this case we can instantiate it at order n over $\mathbb{F}_p^\ell$ where ℓ is the minimum even value greater than m such that $p^\ell \geq 2n$.

In particular, for most symmetric cryptographic schemes, the underlying structure is a finite field of characteristic 2 and over such a binary field, the approach from [29] does not

apply at all. For this case of utmost practical importance, we can use the Gao-Mateer additive FFT [27] for secure implementation of multiplications at order n over binary fields $\mathbb{F}_{2^m}$ for $m \geq 2$. The Gao-Mateer additive FFT is a variant of Cantor additive FFT that works over finite fields of characteristic 2. Using this transform, if m is even with $2^m \geq 2n$, then we can use directly our technique over $\mathbb{K} = \mathbb{F}_{2^m}$ and otherwise we can simply instantiate it over $\mathbb{K} = \mathbb{F}_{2^\ell}$ where ℓ is the smallest even integer for which $2^\ell \geq 2n$ and $m \mid \ell$.

5.3 Security Reduction

This section provides a security reduction for the GJR$^+$ scheme. We show that under the probing security of the FFT, the scheme achieves region probing security. More formally, the reduction is based on the following hypothesis on the FFT algorithm.

Hypothesis 1 (FFT Probing Security)**.** *The circuits processing*

$$\mathsf{FFT}_{\boldsymbol{\alpha}} : (\boldsymbol{x} \parallel \mathbf{0}) \mapsto \boldsymbol{r} \quad \textit{and} \quad \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1} : \boldsymbol{u}' \mapsto \boldsymbol{t}$$

are t_n^{FFT}-probing secure w.r.t. the $\boldsymbol{v}_\omega$-encoding and the $\boldsymbol{v}'_\omega$-encoding respectively.

We can then state our reduction theorem. A discussion of the practical meaning of Hypothesis 1 is given after the theorem proof.

Theorem 3. *Under the FFT Probing Security hypothesis and the t_n^{R}-IOS property of the refresh gadget, the GJR$^+$ compiler is r_n-region probing secure with*

$$r_n = \max_{t \leq t_n^{\mathsf{R}}} \min \left(\frac{t_n^{\mathsf{FFT}} - 6t}{2 \cdot |\mathsf{FFT}_n|} , \frac{t}{|G_n^{\mathsf{R}}|} \right) \tag{10}$$

where $|\mathsf{FFT}_n|$ denotes the (maximum) number of wires in the FFT circuits for $2n$ input sharings.

Note that the refresh gadget described in Section 4 satisfies $t_n^{\mathsf{R}} = n - 1$ and $|G_n^{\mathsf{R}}| = 3n \log n$. Assuming the FFT algorithm is quasilinear and that it can tolerate a linear number of probes (in the encoding order n) and denoting

$$\begin{aligned} |\mathsf{FFT}_n| &= \alpha \cdot n \log n \\ |G_n^{\mathsf{R}}| &= \beta \cdot n \log n \\ t_n^{\mathsf{FFT}} &= \gamma \cdot n \end{aligned}$$

for some constants α, β and γ (with $\gamma < 1$), one can check that the minimum in Equation 19 is reached for

$$t = \left(\frac{\beta\gamma}{2(\alpha + 3\beta)} \right) \cdot n \quad \Longrightarrow \quad r_n = \left(\frac{\gamma}{2(\alpha + 3\beta)} \right) \cdot \frac{1}{\log n} \ . \tag{11}$$

In particular, we obtain a probing rate $r_n = \Theta(1/\log n)$.

The proof of Theorem 3 is based on the two following lemmas.

Lemma 1. *Under the FFT Probing Security hypothesis the circuit processing*

$$(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{u} = \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{x} \parallel \mathbf{0}) \otimes \mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{y} \parallel \mathbf{0})$$

is t_n^{FFT}-probing secure w.r.t. the $\boldsymbol{v}_\omega$-encoding.

Proof. Let us denote by $\widehat{C}$ the considered circuit and $\mathcal{W}$ the set of probed wires from $\widehat{C}$ such that $|\mathcal{W}| \leq t_n^{\mathsf{FFT}}$. We show how to construct the simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ that outputs a perfect distribution of $\widehat{C}_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are uniform $\boldsymbol{v}_\omega$-linear sharings. The simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ first call the simulators $\mathcal{S}_{\mathsf{FFT},\mathcal{W}_1}$ and $\mathcal{S}_{\mathsf{FFT},\mathcal{W}_2}$ by constructing $\mathcal{W}$ as follows: for every $w \in \mathcal{W}$, w is added to $\mathcal{W}_1$ if it corresponds to a wire in the first FFT (*i.e.* applying to $\boldsymbol{x}$) and w is added to $\mathcal{W}_1$ if it corresponds to a wire in the second FFT (*i.e.* applying to $\boldsymbol{y}$). Whenever w corresponds to a product $u_i = r_i \cdot s_i$, then the wire corresponding to r_i is added to $\mathcal{W}_1$ and the wire corresponding to s_i is added to $\mathcal{W}_2$. By construction, we have $|\mathcal{W}_1|, |\mathcal{W}_2| \leq |\mathcal{W}| \leq t_n^{\mathsf{FFT}}$ which ensures that $\mathcal{S}_{\mathsf{FFT},\mathcal{W}_1}$ and $\mathcal{S}_{\mathsf{FFT},\mathcal{W}_2}$ output perfect simulations of all the wires in $\mathcal{W}$ pertaining to the two FFT circuits (by FFT Probing Security hypothesis). Moreover, by construction, they also output the pairs (r_i, s_i) for all the wires in $\mathcal{W}$ corresponding to a product $u_i = r_i \cdot s_i$ which can then be perfectly simulated as well. □

Lemma 2. *Under the FFT Probing Security hypothesis the circuit processing*

$$\boldsymbol{u}' \mapsto \boldsymbol{z} = \mathsf{Compress}(\omega; \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{u}'))$$

is $(t_n^{\mathsf{FFT}}/2)$*-probing secure w.r.t. the* $\boldsymbol{v}'_\omega$*-encoding.*

Proof. The proof follows the same lines as the proof of Lemma 1. Let $\widehat{C}$ denote the considered circuit and $\mathcal{W}$ the set of probed wires from $\widehat{C}$ such that $|\mathcal{W}| \leq t_n^{\mathsf{FFT}}/2$. The simulator $\mathcal{S}_{\widehat{C},\mathcal{W}}$ essentially relies on the simulator $\mathcal{S}_{\mathsf{FFT},\mathcal{W}'}$ where $\mathcal{W}'$ is constructed as follows: for every $w \in \mathcal{W}$, w is added to $\mathcal{W}'$ if it corresponds to a wire in the FFT. Otherwise, w correspond to a wire in the computation $z_i = t_i + \omega^n \cdot t_{n+i}$ for some i, in which case we add the wires corresponding to t_i and t_{n+i} to $\mathcal{W}'$. By construction, we have $|\mathcal{W}'| \leq 2 \cdot |\mathcal{W}| \leq t_n^{\mathsf{FFT}}$ which ensures that $\mathcal{S}_{\mathsf{FFT},\mathcal{W}'}$ outputs a perfect simulation of all the wires in $\mathcal{W}'$, *i.e.* of all the wires in $\mathcal{W}$ pertaining to the FFT plus the pairs (t_i, t_{n_i}) for every i such that a wire in the computation $z_i = t_i + \omega^n \cdot t_{n+i}$ appears in $\mathcal{W}$. The latter wires can then also be perfectly simulated from the pairs (t_i, t_{n_i}), which concludes the proof. □

Proof. (Theorem 3) The proof simply holds from Lemma 1 and Lemma 2 by applying the composition theorem (Theorem 1). We further note that the term depending on the addition gadget can be removed from the expression of the probing rate since the latter satisfies $t_n^{\oplus} = n - 1$ and $|G_n^{\oplus}| = 2n$ which clearly makes it greater than the term depending on the FFT. □

Theorem 3 formally shows that if probing security can be demonstrated for the FFT algorithm, then we obtain region probing security for the GJR$^+$ scheme. Unfortunately, it is not clear whether the classical FFT algorithms are probing secure or not. To some extent, this open issue is related to the choice of ω: some choices lead to probing insecurity[2] while it is not clear whether some choices can provide probing security. Nevertheless, following the approach of [29] it is possible to obtain random probing security by picking ω randomly on a large enough field $\mathbb{K}$. This is formally stated in Subsection 5.4.

Discussion on Hypothesis 1. We now provide some insights about whether Hypothesis 1 is verified in practice. Given input values $\mathbb{K}, \alpha$ and ω, there exists an effectively computable function that checks whether Hypothesis 1 is verified. Indeed, given a $\boldsymbol{v}_\omega$-encoding $\boldsymbol{x}$ of the value x, there exists a circuit C of size $\Theta(n \log n)$ that takes $\boldsymbol{x}$ as input and computes $\mathsf{FFT}_{\boldsymbol{\alpha}}(\boldsymbol{x} \| \boldsymbol{0})$. Probing a node i of the circuit reveals $\langle \boldsymbol{u}_i, \boldsymbol{x} \rangle$, where the value of the vector

[2] For instance, taking ω to 0 or to some nth power of unity when the FFT algorithm is the NTT (as considered in [29]) can be shown to lead to some obvious probing security flaw.

$\boldsymbol{u}_i \in \mathbb{K}^n$ depends only of $\boldsymbol{\alpha}$ and i. The adversary can recover $x = \langle \boldsymbol{v}_\omega, \boldsymbol{x} \rangle$ if and only if he can probe a subset S such that $\boldsymbol{v}_\omega$ is in the span of $(\boldsymbol{u}_i)_{i \in S}$. Indeed, according to Lemma 1 of [29] (see Appendix C):

$$\left(\exists (a_i)_{i \in S} \in \mathbb{K}^{|S|} \text{ s.t. } \boldsymbol{v}_\omega = \sum_i a_i \boldsymbol{u}_i \right) \Leftrightarrow \left(x = \sum_i a_i \cdot \langle \boldsymbol{u}_i, \boldsymbol{x} \rangle \right). \tag{12}$$

Therefore Hypothesis 1 can be verified by checking, for all $\binom{\Theta(n \log n)}{|S|}$ choices of $|S|$ probes in the NTT circuit, whether the left part of Equation 12 holds, a subtask that can be done via Gaussian elimination. When $|S| = \Theta(n)$, the number of subsets to check is superexponential in n but is still tractable via exhaustive search for small values of n.

As an illustration, the circuit C in Figure 4 computes the degree-8 NTT over $\mathbb{F}_{257}$ for input $(x_0, x_1, x_2, x_3, 0, 0, 0, 0)$. Each node i is labelled by a vector $\boldsymbol{u}_i$ such that probing i reveals $\langle \boldsymbol{u}_i, \boldsymbol{x} \rangle$, with $\boldsymbol{x} = (x_0, x_1, x_2, x_3)$. Note that since half the input coefficients are 0, the circuit description is somewhat simpler than a full NTT. By exhaustive search, we can see that C is 3-probing secure for $\omega = 138$, whereas it is only 2-probing secure for $\omega = 209$. Indeed, in the latter case $\boldsymbol{v}_\omega = (1, 209, 248, 175)$, and one can check that:

$$\boldsymbol{v}_\omega = 51 \cdot (1, 64, 241, 4) + 243 \cdot (0, 0, 1, 0) + 207 \cdot (1, 241, 256, 16). \tag{13}$$

This illustrates the importance of the choice of ω in Hypothesis 1.

Distinct finite fields may yield distinct values for t_n^{NTT}. Indeed, applying the same methodology to $\mathbb{F}_{97}$, we found values of ω for which $t_4^{\mathsf{NTT}} = 2$ and $t_8^{\mathsf{NTT}} = 5$, where each time the NTT has degree $2n$.

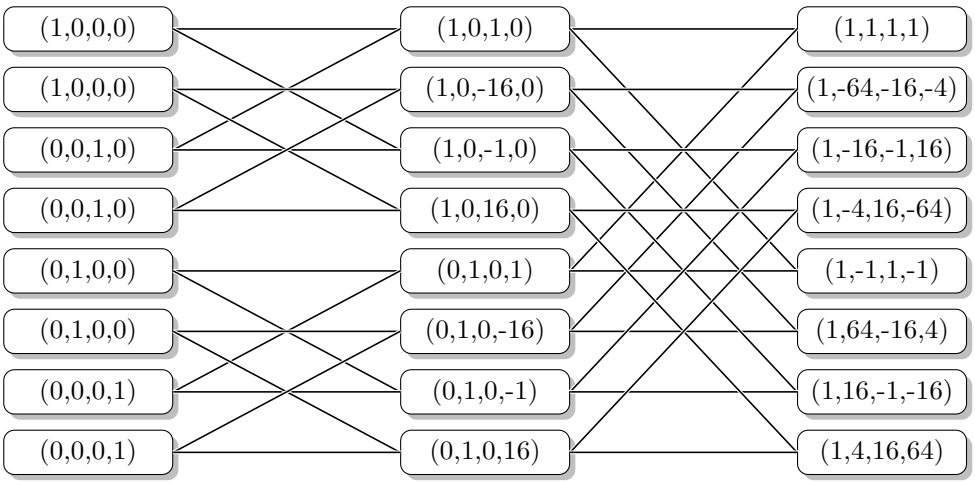

Figure 4: Circuit of the degree-8 NTT applied on an order-4 encoding $\boldsymbol{x} = (x_0, x_1, x_2, x_3)$ for the prime field $\mathbb{F}_{257}$. Probing a node i reveals $\langle \boldsymbol{u}_i, \boldsymbol{x} \rangle$ to an attacker for a given $\boldsymbol{u}_i$. Each butterfly (a subcircuit of the form ⧖) computes $(\boldsymbol{c}, \boldsymbol{d}) \leftarrow (\boldsymbol{a} + \lambda \boldsymbol{b}, \boldsymbol{a} - \lambda \boldsymbol{b})$ given an input $(\boldsymbol{a}, \boldsymbol{b})$ and a fixed λ.

For prime fields and a power-of-two NTT, we were able in practice to determine t_n^{FFT} only for $n \leq 8$, due to combinatorial explosion. This raises the question of proposing algorithms more efficient than exhaustive search for compute t_n^{FFT}. Note that $t_n^{\mathsf{FFT}} + 1$ is the minimum weight w for which we can find a vector $\boldsymbol{a}$ with at most w non-zero coefficients such that $\boldsymbol{a} \cdot \boldsymbol{U} = \boldsymbol{v}_\omega$, where $\boldsymbol{U} = (\boldsymbol{u}_i)_i \in \mathbb{K}^{\Theta(n \log n) \times n}$. This can be cast as a specific instance of the information-set decoding (ISD) problem, which is common in code-based cryptography. However, unlike ISD instances that are usually studied in

code-based cryptanalysis, the matrix $\boldsymbol{U}$ we consider has more lines than columns, is not random but fixed, and the underlying field may be non-binary. We see the question of computing t_n^{FFT} more efficiently as an interesting open problem.

5.4 Security Proof for Large Fields

The security proof given hereafter follows the same lines as the original proof from [29] but it is more general as it applies to any instance of the GJR$^+$ scheme (with any field and FFT algorithm) and it holds in the stronger region probing model. Moreover, our security proof corrects a flaw in the original proof which we exhibit hereafter.

Flaw in the original proof. The original GJR scheme is based on a different refresh gadget for the composition. In a nutshell, their gadget follows the classical approach of adding a random ω-encoding of 0. The latter is generated based on a fixed ω-encoding of 0 denoted $\boldsymbol{e}$ in [29], which is randomly generated at the beginning of the computation and which is considered to be fully leaked to the adversary. When a fresh ω-encoding of 0 must be generated, one draws a random vector $\boldsymbol{u}$ and multiplies it with $\boldsymbol{e}$ through the multiplication gadget, which gives a fresh and uniform ω-encoding of 0. Such a refresh procedure satisfies a slightly weaker version of the IOS property (see Subsection 3.1 for comparison). Specifically, when a sharing $\boldsymbol{x}$ is refreshed into a new sharing $\boldsymbol{x}'$, the leakage from the refresh procedure can be simulated by linear combinations of $\boldsymbol{x}$ and linear combinations of $\boldsymbol{x}'$. These leaking linear combinations can in turn be perfectly simulated with overwhelming probability over the random distribution of ω, provided that ω is defined on a large enough field $\mathbb{K}$ (see Lemmas 1 & 2 of [29] which we recall in Appendix C). The flaw in the proof is that it implicitly assumes that the aforementioned leaking linear combinations have constant coefficients with respect to ω. However, by definition of the refresh procedure, these coefficients depend on $\boldsymbol{e}$, the initial ω-encoding of 0, which cannot be considered as constant with respect to ω. This prevents the application of Lemmas 1 & 2 of [29]. This bug invalidates the composition security proof of the original GJR scheme although it does not lead to an obvious security flaw: it is not clear whether the linear combinations coming from the refresh imply an exploitable information leakage (or equivalently a simulation failure).

New proof. The region-probing security of the GJR$^+$ scheme simply holds from the IOS property of the refresh gadget and assuming that the underlying FFT algorithm is somehow linear. This is captured by the following definition.

Definition 8 (Linear FFT Circuit)**.** An FFT circuit is said *linear* if the circuits processing

$$\mathsf{FFT}_{\boldsymbol{\alpha}} : (\boldsymbol{x} \parallel \boldsymbol{0}) \mapsto \boldsymbol{r} \quad \text{and} \quad \mathsf{FFT}_{\boldsymbol{\alpha}}^{-1} : \boldsymbol{u}' \mapsto \boldsymbol{t}$$

are composed of additions and multiplications by constants (on $\mathbb{K}$).

The above definition implies that the value carried by each wire in the FFT circuit can be expressed as a linear combination of the coordinates of the input sharing. This property is necessary to apply the security argument of the original GJR scheme. Note that this requirement is relatively weak since it is satisfied by classical FFT algorithms such as the NTT (used in [29]) and the Gao-Mateer additive FFT [27].

Corollary 1. *If the FFT circuit is linear and made of* $|\mathsf{FFT}_n| = \alpha\, n \log n$ *wires, the GJR*$^+$ *compiler is* (r_n, ε_n)*-region probing secure with*

$$r_n = \left(\frac{1}{2\alpha + 18}\right) \frac{1}{\log n} \ , \tag{14}$$

and

$$\varepsilon_n = \frac{n}{|\mathbb{K}|} \ . \tag{15}$$

Proof. Using Lemmas 1 & 2 of [29] (which are recalled in Appendix C for the sake of completeness) and thanks to the linearity of the FFT circuit, we have that for any choice of $n-1$ leaking wires from the FFT circuit, the probability that the leaking wires cannot be perfectly simulated is lower than $n/|\mathbb{K}|$. Besides the linearity of the FFT circuit, the only requirement for this upper bound to apply is that the choice of the leaking wires is made independently of ω, which occurs in the region probing model since the placement of the probes by the adversary is done independently of the random generation of ω. We can then directly apply Theorem 3 and obtain the probing rate r_n from Equation 11 with $\gamma = \frac{n-1}{n} \approx 1$ and $\beta = 3$. □

In Appendix B, we further detail the security proof of GJR$^+$ in the random probing model which holds from its security in the region probing model by applying the Chernoff's bound. This further implies the security of GJR$^+$ in the noisy leakage model by the reduction from [25].

6 Application

In this section, we present an application of our extended GJR scheme and compare it with a more standard scheme based on SNI gadgets. We investigate the masked computation of two different ciphers:

- The Advanced Encryption Standard (AES) [1]: a very common application scenario which favors efficient masking schemes on the field $\mathbb{F}_{256}$;
- MiMC [4]: a cipher with efficient arithmetic representation on a large field. MiMC has been designed with the aim to minimize the number of multiplications (which makes it particularly amenable to masked computation). We focus on the prime-field variant of MiMC (the base field is a prime field $\mathbb{F}_p$).

For these two application contexts, we described masked computations based on the two following masking schemes:

- The GJR$^+$ scheme described in Section 5 of this paper, in two modes of application:
  - on a binary field with the Gao-Mateer FFT algorithm (to mask AES),
  - on a prime field with NTT algorithm (to mask MiMC).
- An extended ISW scheme, that we shall refer to as ISW$^+$, and which is based on
  - the ISW multiplication gadget [31] over the base field $\mathbb{K}$ (either $\mathbb{F}_{256}$ for AES or $\mathbb{F}_p$ for MiMC),
  - share-wise linear gadgets (for additions, subtractions, $\mathbb{F}_{256}$-squares, multiplications by constants),
  - the BCPZ quasilinear refresh gadget [10].

We first address implementation aspects of the two above masking schemes, then describes masking of AES and MiMC with these schemes and finally provide comparison of performances in terms of operation counts and randomness consumption.

Cautionary note: To ease the presentation, we consider that each gadget includes a refreshing of its output sharing, except the ISW multiplication gadget which achieves the SNI notion without further refreshing. In a masked computation, a sharing might be input

of several gadgets which would be an issue with respect to region probing security (*e.g.* one could accumulate t probes on this sharing per gadget). We therefore impose that such a sharing is refreshed before each new usage.

6.1 Implementation of GJR$^+$

6.1.1 Multiplication gadget on $\mathbb{F}_p$ based on the NTT

It is well-known that polynomials can be multiplied in quasi-linear time in finite fields using the *Number Theoretic Transform* (NTT), a Fast Fourier Transform (FFT) which requires that the coefficient ring contain certain roots of unity. More precisely, it is possible to multiply two polynomials of degree $\leq N$ in a finite field $\mathbb{F}_q$ in $O(N \log N)$ arithmetic operations in $\mathbb{F}_q$ if $\mathbb{F}_q$ contains a primitive $2N$-th root of unity (which occurs if and only if $2N$ divides $q-1$). The number theoretic transform was introduced by Pollard [37] and we refer the reader to [43, Section 8.2] for an exhaustive description. As stated in [43, Theorem 8.18], if N is a power of S ($N = 2^m$), the multiplication of two polynomials of degree $< N$ in a finite field $\mathbb{F}_q$ which contains a $2N$-th root of unity requires $6N\log(N) + 6N$ additions in $\mathbb{F}_q$, $3N\log(N) + 4N - 2$ multiplications by constants in $\mathbb{F}_q$, $2N$ (bilinear) multiplications in $\mathbb{F}_q$ and $2N$ divisions by $2N$ in $\mathbb{F}_q$.

Our multiplication gadget (for an encoding order n) described in Subsection 5.1 over such a finite field has thus a total complexity of $8n\log(n)+11n$ additions in $\mathbb{F}_q$, $5n\log(n)+7n-2$ multiplications by constants in $\mathbb{F}_q$ and $2n$ (bilinear) multiplications in $\mathbb{F}_q$. It requires $2n\log(n) + 2n$ random elements from $\mathbb{F}_q$.

6.1.2 Multiplication gadget on $\mathbb{F}_{2^k}$ based on the Gao-Mateer FFT

The classical NTT cannot be applied when the underlying field does not have the desired roots of unity. We describe the *additive FFT* algorithm proposed by Gao and Mateer in 2010 [27], which works over fields of characteristic two. The idea of this class of FFTs is to evaluate polynomials of degree m over a linear (additive) subspace of $\mathbb{K}[x]$ rather than a group and it comes in two flavors: generic algorithms for an arbitrary m, or specialized ones for m a power of two. The specialized algorithms are faster than the generic ones, but the condition on m heavily constraints their use. We will use it with a slight generalization of Cantor bases which we call *self-folding bases*, and which allow even more aggressive optimization than what is done in [16, 15, 21] and may be of independent interest.

Let $\mathbb{F} = \mathbb{F}_{2^k}$ be a finite field of characteristic two.We consider the additive FFT of a polynomial $f \in \mathbb{F}[x]$, that is we evaluate f over the $\mathbb{F}_2$-linear span generated by m elements $\beta_0, \ldots, \beta_m \in \mathbb{F}$ linearly independent over $\mathbb{F}_2$. This span contains $N = 2^m$ elements, and is defined if and only if $N \leq 2^k$, or equivalently $m \leq k$.

Taylor Expansion. An important subroutine of the Gao-Mateer additive FFT is the Taylor expansion, a slight variant of the usual notion of Taylor series. It consists of writing any polynomial $f \in \mathbb{F}[x]$ of degree $< N$ as follows:

$$f(x) = \sum_{i=0}^{N/2-1} h_i(x) \cdot (x^2 - 1)^i, \tag{16}$$

where each h_i is a polynomial of $\mathbb{F}[x]$ of degree at most 1. Algorithm 6 (provided in Appendix D) is an algorithm presented in [27] for computing the Taylor expansion of a polynomial in the case where m is generic (and not a power of 2), in which it is shown to require $\frac{1}{2}N\log N - \frac{1}{2}N$ field additions and no multiplication.

Basis folding. We abusively call basis any subset $B = \{\beta_0, \ldots, \beta_{m-1}\} \subset \mathbb{F}^m$ of m elements of $\mathbb{F}$ linearly independent over $\mathbb{F}_2$.[3] For a basis B of length m and an integer $0 \leq i < 2^m$ which can be written as $i = \sum_{j=0}^{m-1} a_j 2^j$ with $a_j \in \{0, 1\}$, we will note:

$$B[i] = \sum_{j=0}^{m-1} a_j \beta_j. \tag{17}$$

We will say that $B[i]$ is the i-th element of $\langle B \rangle$. An important subroutine in [27] consists of what we call *folding* a basis, a process we recall in Algorithm 7.

Additive FFT. Gao and Mateer [27] proposed an algorithm for computing the additive FFT of a polynomial f over the subspace $\langle B \rangle$ generated by a basis B. This algorithm is described in Algorithm 8 (Appendix D), which costs $2N \log N - 2N + 1$ field additions, as well as $\frac{1}{4}N(\log N)^2 + \frac{3}{4}N \log N - \frac{N}{2}$ scalar field multiplications. No formal description of the inverse algorithm is given, but it is observed that an inverse additive FFT can be obtained by performing the inverse of each operation in the reverse order. Since the scalar field multiplications and their inversions can be precomputed, the cost of the inverse additive FFT is the same as for the forward algorithm.

Self-Folding Bases. One could assume that the choice of B is not too important, because the operations involving B can be precomputed anyway. However, we show in this subsection that carefully choosing B can go a long way in making the additive FFT faster, simpler to implement and less costly in memory.

In fact, in the case where m is a power of two, by taking $\beta_{m-1} = 1$ for their Cantor basis, Bernstein *et al.* [16] showed that one could saved up some of the multiplication operations (namely the computation that involved the inverse of β_{m-1}) for the top-level recursion cases. We take this idea further and propose a specific kind of bases such that $\beta_{m-1} = 1$ at every layer of the recursion. In the following, those kind of bases are called self-folding bases.

Definition 9 (Self-folding bases)**.** Let $\mathbb{F}$ be a finite field of characteristic two and let $m \geq 1$. A self-folding basis $B = \{\beta_0, \ldots, \beta_{m-1}\} \subset \mathbb{F}^m$ is a basis which verifies the two following conditions:

1. $\beta_{m-1} = 1$;
2. for $i \in \{0, \ldots, m-2\}$, it holds that $\beta_i^2 - \beta_i = \beta_{i+1}$.

We have the following properties about self-folding bases:

Proposition 1. *Let $\mathbb{F}$ be a finite field of characteristic two and let $m \geq 1$. The following properties hold:*

1. *Let $B = \{\beta_0, \ldots, \beta_{m-1}\}$ be a self-folding basis and $G, D \leftarrow \mathsf{Fold}(B)$. We have $G = \{\beta_0, \beta_1, \ldots, \beta_{m-2}\}$ and $D = \{\beta_1, \beta_2, \ldots, \beta_{m-1}\}$.*
2. *Let $\mathbb{F}'$ be a subfield of $\mathbb{F}$ of cardinality $2^{k'}$. For any $m \leq k'$, there exists a self-folding basis B of m elements such that $\langle B \rangle$ is included in $\mathbb{F}'$.*

Proof. The first item is immediate from the definition. The second item is proven in the special case $\mathbb{F}' = \mathbb{F}$ and $m = k'$ in the appendix of [27], and the proof is constructive. From there, the general case is immediate to obtain by embedding the solutions for $\mathbb{F}'$ in the larger field $\mathbb{F}$ and keeping only the m last elements of B. □

[3] Here our notation differ slightly from [27], where B denotes the linear space spanned by the basis. We find it more natural for our purposes to denote by B the basis.

From Proposition 1, we can see that B "folds onto itself": folding B into G, D yields subsets of B, and this self-folding property transfers to D.

The notion of self-folding basis is close to that of Cantor basis [17, 27]; the only difference is that the elements are taken in reverse order, and that no condition is imposed on the basis elements belonging to a subfield $\mathbb{F}_{2^m}$ (such a subfield does not always exist, thus self-folding bases are slightly more general objects than Cantor bases). The most important difference, however, is how they are used. In [27, 16, 15], Cantor bases are used to speed up the *specialized* additive FFT algorithm of [27], which only works on a restricted set of parameters (namely, when m is a power of two). In comparison, our improvements apply to speed up the *generic* algorithm of [27], which works for any value of m.

Half-FFT. Our improved (half-)FFT works with a self-folding basis and its iterated foldings. It is described in Appendix D. Its main advantage is that the step 3 of Algorithm 8 becomes unnecessary; in total, this saves us $N \log N$ scalar multiplications. Moreover, it divides by two the number of precomputed tables. As another algorithmic optimization, we make full use of the fact that for polynomial multiplications, half the inputs of the FFT's call are zero coefficients which leads to speed up to the computations by a factor two compared to a regular additive FFT. These optimizations applies in a similar way to the inverse additive FFT (with the difference here is that we cannot exploit anymore the fact that half of the polynomial coefficients are zero).

Complexity. In this section, we provide the complexity of our new algorithms (the detailed analysis is provided in Appendix D). Table 1 gives the operation counts for our variant of the Gao-Mateer additive FFT. For N being a power of 2 ($N = 2^m$), the multiplication of two polynomials of degree lower than N in a finite field $\mathbb{F}_{2^k}$ requires $(1/2)N\log^2(N) + 2N\log(N) - 2N$ additions in $\mathbb{F}_{2^k}$, $(3/2)N\log(N) - 2N$ multiplications by constants in $\mathbb{F}_{2^k}$ and $2N$ (bilinear) multiplications in $\mathbb{F}_{2^k}$. Our multiplication gadget (for an encoding order n) described in Subsection 5.1 has thus a total complexity of $(1/2)n\log^2(n) + 4n\log(n) + n$ additions in $\mathbb{F}_{2^k}$, $(7/2)n\log(n)$ multiplications by constants in $\mathbb{F}_{2^k}$, and $2n$ (bilinear) multiplications in $\mathbb{F}_{2^k}$. It further requires $2n\log(n) + 2n$ random elements from $\mathbb{F}_{2^k}$. Table 2 summarizes the operation counts for the different gadgets for GJR$^+$.

Table 1: Complexity of our variant of Gao-Mateer additive FFT.

	Gao-Mateer FFT [27]	HalfFFT	InverseFFT
Add	$\frac{1}{4} \cdot N \cdot (\log_2(N))^2 + \frac{3}{4} \cdot N \cdot \log_2(N) - \frac{1}{2} \cdot N$	$\frac{1}{8} \cdot N \cdot (\log_2(N))^2 + \frac{5}{8} \cdot N \cdot \log_2(N) - N$	$\frac{1}{4} \cdot N \cdot (\log_2(N))^2 + \frac{3}{4} \cdot N \cdot \log_2(N)$
Mult	$2 \cdot N \cdot \log_2(N) - 2N - 1$	$\frac{1}{2} \cdot N \cdot \log_2(N) - \frac{3}{4} \cdot N$	$\frac{1}{2} N \cdot \log_2(N) - \frac{1}{2} \cdot N$

Table 2: Operation counts for GJR$^+$.

	Mult	Add	Random
Mult. gadget ($\mathbb{F}_p$)	$5n\log(n) + 9n - 2$	$8n\log(n) + 11n$	$2n\log(n) + 2n$
Mult. gadget ($\mathbb{F}_{2^m}$)	$7n\log(n)/2 + 2n$	$n\log^2(n)/2 + 4n\log(2n)$	$2n\log(n) + 2n$
Addition gadget	0	n	0
Refresh gadget	$n\log(n)/2$	$n\log(n)$	$n\log(n)/2$

6.2 Implementation of ISW$^+$

Table 3 summarizes the operation counts of the different gadgets for ISW$^+$.

Table 3: Operation counts for ISW$^+$.

	Mult	Add	Random
Mult. gadget (ISW) [31]	n^2	$2n(n-1)$	$n(n-1)/2$
Addition gadget	0	n	0
Refresh gadget (BCPZ) [10]	0	$2n\log(n)-n$	$n\log(n)-n/2$

6.3 Masking of AES

We consider the AES cipher as an arithmetic circuit over $\mathbb{F}_{256}$, composed of additions, multiplications, squares, multiplication by constants, and a special gate computing the affine part of the AES s-box. Besides the multiplication, all these operations give rise to a *linear gadget* in the masked setting, *i.e.* a gadget which applies the operation share-wisely and refreshes the output sharing.

Algorithm 3 Masked AES

Require: n-sharings $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}$ of plaintext bytes $x_1, \ldots, x_{16} \in \mathbb{F}_{256}$, n-sharings $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{16}$ of the round key bytes $k_1^j, \ldots, k_{16}^j \in \mathbb{F}_{256}$, for $0 \leq j \leq r$

Ensure: n-sharing of $\mathsf{AES}(x, k)$

1: $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow (G_n^{\oplus}(\boldsymbol{x}_1, \boldsymbol{k}_1^0), \ldots, G_n^{\oplus}(\boldsymbol{x}_{16}, \boldsymbol{k}_{16}^0)))$
2: **for** $j = 1, \ldots, r-1$ **do**
3: $\quad(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow (G_n^{\mathsf{Aff}}(\mathsf{MaskedExp}(\boldsymbol{x}_1)), \ldots, G_n^{\mathsf{Aff}}(\mathsf{MaskedExp}(\boldsymbol{x}_{16})))$
4: $\quad(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow \mathsf{ShiftRows}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16})$
5: $\quad(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow \mathsf{MaskedMixColumns}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16})$
6: $\quad(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow (G_n^{\oplus}(\boldsymbol{x}_1, \boldsymbol{k}_1^j), \ldots, G_n^{\oplus}(\boldsymbol{x}_{16}, \boldsymbol{k}_{16}^j)))$
7: $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow (G_n^{\mathsf{Aff}}(\mathsf{MaskedExp}(\boldsymbol{x}_1)), \ldots, G_n^{\mathsf{Aff}}(\mathsf{MaskedExp}(\boldsymbol{x}_{16})))$
8: $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow \mathsf{ShiftRows}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16})$
9: $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16}) \leftarrow (G_n^{\oplus}(\boldsymbol{x}_1, \boldsymbol{k}_1^r), \ldots, G_n^{\oplus}(\boldsymbol{x}_{16}, \boldsymbol{k}_{16}^r)))$
10: **return** $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{16})$

The masked description of AES is described in Algorithm 3. The linear gadget for the s-box affine transformation is denoted G_n^{Aff}. The ShiftRows transformation simply permutes the byte indexes and is virtually free (in particular it does not involve any computation on the shares). The MaskedMixColumns transformation simply applies the MixColumns transformation by using gadgets $G_n^{\oplus}$ and G_n^{xtimes} in place of xors and xtimes operations (*i.e.* multiplications by the constant 02 on the field $\mathbb{F}_{256}$). We consider the implementation described in [26] which involves 15 additions and 3 xtimes per column:

$$
\begin{aligned}
acc &\leftarrow x_1 \oplus x_2 \oplus x_3 \oplus x_4 \\
y_1 &\leftarrow \mathsf{xtimes}(x_1 \oplus x_2) \oplus acc \oplus x_1 \\
y_2 &\leftarrow \mathsf{xtimes}(x_2 \oplus x_3) \oplus acc \oplus x_2 \\
y_3 &\leftarrow \mathsf{xtimes}(x_3 \oplus x_4) \oplus acc \oplus x_3 \\
y_4 &\leftarrow y_1 \oplus y_2 \oplus y_3 \oplus acc
\end{aligned}
$$

A masked implementation of this process thus involves 18 linear gadgets as well as 15 refresh gadgets (for the variables used multiple times). The MaskedExp procedure is further depicted in Algorithm 4, which is based on the Rivain-Prouff scheme [41]. As explained above, a sharing in input of several gadgets is refreshed before each new usage.

Algorithm 4 MaskedExp

Require: n-sharing $\boldsymbol{x}$ of $x \in \mathbb{F}_{256}$
Ensure: n-sharing of x^{254}

1: $\boldsymbol{z} \leftarrow G_n^{(\cdot)^2}(\boldsymbol{x})\,; \quad \boldsymbol{x} \leftarrow G_n^{\mathsf{R}}(\boldsymbol{x})$
2: $\boldsymbol{y} \leftarrow G_n^{\otimes}(\boldsymbol{x}, \boldsymbol{z})\,; \quad \boldsymbol{z} \leftarrow G_n^{\mathsf{R}}(\boldsymbol{z})$
3: $\boldsymbol{w} \leftarrow G_n^{(\cdot)^4}(\boldsymbol{y})\,; \quad \boldsymbol{y} \leftarrow G_n^{\mathsf{R}}(\boldsymbol{y})$
4: $\boldsymbol{y} \leftarrow G_n^{\otimes}(\boldsymbol{y}, \boldsymbol{w})\,; \quad \boldsymbol{w} \leftarrow G_n^{\mathsf{R}}(\boldsymbol{w})$
5: $\boldsymbol{y} \leftarrow G_n^{(\cdot)^{16}}(\boldsymbol{y})$
6: $\boldsymbol{y} \leftarrow G_n^{\otimes}(\boldsymbol{y}, \boldsymbol{w})$
7: $\boldsymbol{y} \leftarrow G_n^{\otimes}(\boldsymbol{y}, \boldsymbol{z})$
8: **return** $\boldsymbol{y}$

The gadget count of the masked AES is given in Table 4. According to Algorithm 4, the MaskedExp involves 4 multiplication gadgets, 3 linear gadgets and 4 refresh gadgets. According to the above description, we get a total of 72 linear gadgets and 60 refresh gadgets for the full MaskedMixColumns. One full round is composed of 16 calls to MaskedExp, one call to MaskedMixColumns, plus 32 linear gadgets (16 gadgets $G_n^{\oplus}$ and 16 gadgets G_n^{Aff}). A full AES computation is composed of 9 full rounds, 1 partial round (without the MixColumns) plus one key addition (16 gadgets $G_n^{\oplus}$).

Table 4: Gadget counts for masked AES.

	Mult.	Linear	Refresh
MaskedExp	4	3	4
MaskedMixColumns	0	72	60
One round	64	136	124
Full masked AES	640	1304	1180

6.4 Masking of MiMC

Let x be some plaintext and k be some secret key, both belonging to some large field $\mathbb{K}$. The MiMC cipher is defined as:

$$\mathsf{MiMC}(x, k) = F_{k,c_r} \circ \cdots \circ F_{k,c_1}(x) + k \ ,$$

with $F_{k,c_i}(x) = (x + k + c_i)^3$, with $r = \lceil \log(|\mathbb{K}|)/\log(3) \rceil$.

For our application, we consider the prime field variant of MiMC for which $\mathbb{K}$ can be chosen as any prime field $\mathbb{F}_p$ such that $\gcd(3, p-1) = 1$ (so that x^3 is invertible on $\mathbb{F}_p$). Since we wish to apply the GJR$^+$ scheme based on the NTT (as in the original GJR scheme), the chosen field must further satisfy $p - 1 = \alpha \cdot (2n_{max})$ for some odd integer α and some integer n_{max} which is a power of two. In practice n_{max} is the maximum masking order which can be achieved by the GJR$^+$ scheme. We hence choose a prime $p = \alpha \cdot 2^{\ell+1} + 1$ with $\gcd(\alpha, 3) = 1$ and $\ell = \log_2 n_{max}$. Specifically, for a given target field size $\lambda = \lceil \log_2 p \rceil$, we search for greatest integer ℓ and smallest integer α such that: *(i)* $3 \nmid \alpha$, *(ii)* $\log_2 \alpha + \ell + 1 < \lambda$, and *(iii)* $p = \alpha \cdot 2^{\ell+1} + 1$ is prime. For our application, we thus instantiate MiMC with such 128-bit and 256-bit prime fields:

- for $\lambda = 128$, we get $p = 407 \cdot 2^{119} + 1$, giving $n_{max} = 118$,
- for $\lambda = 256$, we get $p = 467 \cdot 2^{247} + 1$, giving $n_{max} = 246$.

Algorithm 5 gives a masked description of MiMC based on any standard circuit compiler. For the sake of clarity, we omit to apply the refresh gadget G_n^{R} to the output of an arithmetic gadget $G_n^{\oplus}$ or $G_n^{\otimes}$ and consider that the refresh is part of these gadget when necessary. For $\lambda = 128$ (resp. $\lambda = 256$), the number of rounds is $r = 81$ (resp. $r = 162$).

Algorithm 5 Masked MiMC

Require: n-sharing $\boldsymbol{x}$ of plaintext $x \in \mathbb{F}_p$, n-sharing $\boldsymbol{k}$ of secret key $k \in \mathbb{F}_p$
Ensure: n-sharing of $\mathsf{MiMC}(x, k)$

1: **for** $i = 1, \ldots, r$ **do**
2: $\quad \boldsymbol{x} \leftarrow G_n^{\oplus}(\boldsymbol{x}, \boldsymbol{k}, c_i)$
3: $\quad \boldsymbol{k} \leftarrow G_n^{\mathsf{R}}(\boldsymbol{k})$
4: $\quad \boldsymbol{y} \leftarrow G_n^{\otimes}(\boldsymbol{x}, \boldsymbol{x})$
5: $\quad \boldsymbol{x} \leftarrow G_n^{\otimes}(\boldsymbol{x}, \boldsymbol{y})$
6: $\boldsymbol{x} \leftarrow G_n^{\oplus}(\boldsymbol{x}, \boldsymbol{k})$
7: **return** $\boldsymbol{x}$

Table 5: Gadget counts for masked MiMC.

	Mult.	Linear	Refresh
One round	2	2	1
Full MiMC ($\lambda = 128$)	162	163	81
Full MiMC ($\lambda = 256$)	324	325	162

6.5 Performances and Comparison

Table 6 and Table 7 summarize the operation counts for full MiMC (with $\lambda = 128$) and full AES with the two masking schemes ISW$^+$ and with GJR$^+$ implemented over the same finite field. They show that our approach results in a 62% decrease in the randomness complexity and a 51% decrease of the number of multiplication for MiMC masked at order 128 and in a 46% (*resp.* 59%) decrease in the randomness complexity and a 20% (*resp.* 52%) decrease of the number of multiplication for AES masked at order 64 (*resp.* 128).

For AES, the masking scheme ISW$^+$ can always be implemented over the $\mathbb{F}_{256}$ finite field. However, to achieve provable region-probing security without relying on Hypothesis 1, Corollary 1 imposes that the masking scheme GJR$^+$ is implemented over a larger finite field such as $\mathbb{F}_{2^{128}}$ (which has less efficient arithmetic). To compare the different complexities of the schemes GJR$^+$ and ISW$^+$, we can implement $\mathbb{F}_{2^{128}}$ as a degree 16 extension of $\mathbb{F}_{2^8} = \mathbb{F}_{256}$, so that: (1) an addition over $\mathbb{F}_{2^{128}}$ takes 16 additions over $\mathbb{F}_{256}$, (2) a multiplication over $\mathbb{F}_{2^{128}}$ takes 81 multiplications (and a large number of additions) over $\mathbb{F}_{256}$ using Karatsuba's algorithm (since a multiplication of two polynomials of degree at most $16 = 2^4$ over $\mathbb{F}_{256}$ requires $81 = 3^4$ multiplications, see [43, Section 8.1]) and (3) a random element of $\mathbb{F}_{2^{128}}$ requires 16 random elements of $\mathbb{F}_{256}$. The computational efficiency of the masking scheme GJR$^+$ for AES compared to ISW$^+$ is then only better for masking order $n \geq 8192$ and its randomness complexity is better for masking order $n \geq 2048$.

Table 6: Performances comparison for masked MiMC ($\lambda = 128$).

n		Mul	Add.	Random
8	Full MiMC with ISW$^+$	10416.0	45408.0	17544.0
	Full MiMC with GJR$^+$	40512.0	66128.0	20100.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	3.89	1.46	1.15
16	Full MiMC with ISW$^+$	41600.0	153056.0	55856.0
	Full MiMC with GJR$^+$	100796.0	165968.0	51872.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	2.43	1.09	0.93
32	Full MiMC with ISW$^+$	166208.0	513536.0	173984.0
	Full MiMC with GJR$^+$	240812.0	399360.0	127088.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	1.45	0.78	0.74
64	Full MiMC with ISW$^+$	664320.0	1773696.0	555456.0
	Full MiMC with GJR$^+$	559740.0	933568.0	300864.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	0.85	0.53	0.55
128	Full MiMC with ISW$^+$	2656000.0	6367744.0	1857664.0
	Full MiMC with GJR$^+$	1275388.0	2136832.0	695104.0
	Efficiency ratio (GJR$^+$/ISW$^+$)	0.49	0.34	0.38

Table 7: Performances comparison for masked AES.

n		Mul	Add.	Random
8	Full AES with ISW$^+$	64896	297088	123520
	Full AES with GJR$^+$	157056	257408	110080
	Efficiency ratio (GJR$^+$/ISW$^+$)	2.43	0.87	0.9
16	Full AES with ISW$^+$	211712	926976	372480
	Full AES with GJR$^+$	396032	683776	286720
	Efficiency ratio (GJR$^+$/ISW$^+$)	1.88	0.74	0.77
32	Full AES with ISW$^+$	751104	2847232	1077760
	Full AES with GJR$^+$	955904	1725952	706560
	Efficiency ratio (GJR$^+$/ISW$^+$)	1.28	0.61	0.66
64	Full AES with ISW$^+$	2812928	8991744	3148800
	Full AES with GJR$^+$	2239488	4209664	1679360
	Efficiency ratio (GJR$^+$/ISW$^+$)	0.8	0.47	0.54
128	Full AES with ISW$^+$	10868736	29820928	9594880
	Full AES with GJR$^+$	5134336	10016768	3891200
	Efficiency ratio (GJR$^+$/ISW$^+$)	0.48	0.34	0.41

Acknowledgments

This work was partly supported by the French FUI-AAP25 VeriSiCC project and by the Innovate UK Research Grant 104423 (PQ Cybersecurity). We would like to thank Jean-Sébastien Coron as well as the anonymous reviewers for meaningful comments and suggestions. Special thanks to Gaëtan Cassiers for reporting a flaw in the original version of Section 4 (refresh gadget and IOS proof) and for follow-up discussions that helped to patch the flaw.

References

[1] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, Nov. 2001.

[2] M. Ajtai. Secure computation with information leaking to an adversary. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 715–724. ACM Press, June 2011.

[3] M.-L. Akkar and C. Giraud. An implementation of DES and AES, secure against some attacks. In Çetin Kaya. Koç, D. Naccache, and C. Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 309–318. Springer, Heidelberg, May 2001.

[4] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, Dec. 2016.

[5] P. Ananth, Y. Ishai, and A. Sahai. Private circuits: A modular approach. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 427–455. Springer, Heidelberg, Aug. 2018.

[6] M. Andrychowicz, S. Dziembowski, and S. Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 586–615. Springer, Heidelberg, May 2016.

[7] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, F.-X. Standaert, and P.-Y. Strub. Improved parallel mask refreshing algorithms: Generic solutions with parametrized non-interference & automated optimizations. Cryptology ePrint Archive, Report 2018/505, 2018. https://eprint.iacr.org/2018/505.

[8] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini. Strong non-interference and type-directed higher-order masking. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 116–129. ACM Press, Oct. 2016.

[9] A. Battistello, J.-S. Coron, E. Prouff, and R. Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In B. Gierlichs and A. Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 23–39. Springer, Heidelberg, Aug. 2016.

[10] A. Battistello, J.-S. Coron, E. Prouff, and R. 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.

[11] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud. Randomness complexity of private circuits for multiplication. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 616–648. Springer, Heidelberg, May 2016.

[12] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud. Private multiplication over finite fields. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 397–426. Springer, Heidelberg, Aug. 2017.

[13] S. Belaïd, J.-S. Coron, E. Prouff, M. Rivain, and A. R. Taleb. Random probing security: Verification, composition, expansion and new constructions. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 339–368. Springer, Heidelberg, Aug. 2020.

[14] S. Belaïd, D. Goudarzi, and M. Rivain. Tight private circuits: Achieving probing security with the least refreshing. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 343–372. Springer, Heidelberg, Dec. 2018.

[15] D. J. Bernstein and T. Chou. Faster binary-field multiplication and faster binary-field MACs. In A. Joux and A. M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 92–111. Springer, Heidelberg, Aug. 2014.

[16] D. J. Bernstein, T. Chou, and P. Schwabe. McBits: Fast constant-time code-based cryptography. In G. Bertoni and J.-S. Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 250–272. Springer, Heidelberg, Aug. 2013.

[17] D. G. Cantor. On arithmetical algorithms over finite fields. *J. Comb. Theory, Ser. A*, 50(2):285–300, 1989.

[18] C. Carlet, L. Goubin, E. Prouff, M. Quisquater, and M. Rivain. Higher-order masking schemes for S-boxes. In A. Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 366–384. Springer, Heidelberg, Mar. 2012.

[19] G. Cassiers and F. Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[20] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Heidelberg, Aug. 1999.

[21] T. Chou. McBits revisited. In W. Fischer and N. Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 213–231. Springer, Heidelberg, Sept. 2017.

[22] J.-S. Coron. Higher order masking of look-up tables. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 441–458. Springer, Heidelberg, May 2014.

[23] J.-S. Coron, E. Prouff, M. Rivain, and T. Roche. Higher-order side channel security and mask refreshing. In S. Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 410–424. Springer, Heidelberg, Mar. 2014.

[24] J.-S. Coron, A. Roy, and S. Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In L. Batina and M. Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 170–187. Springer, Heidelberg, Sept. 2014.

[25] A. Duc, S. Dziembowski, and S. Faust. Unifying leakage models: From probing attacks to noisy leakage. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, Heidelberg, May 2014.

[26] G. Fumaroli, A. Martinelli, E. Prouff, and M. Rivain. Affine masking against higher-order side channel analysis. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 262–280. Springer, Heidelberg, Aug. 2011.

[27] S. Gao and T. Mateer. Additive fast Fourier transforms over finite fields. *IEEE Transactions on Information Theory*, 56(12):6265–6272, Dec 2010.

[28] L. Goubin and J. Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya. Koç and C. Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, Heidelberg, Aug. 1999.

[29] D. Goudarzi, A. Joux, and M. Rivain. How to securely compute with noisy leakage in quasilinear complexity. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 547–574. Springer, Heidelberg, Dec. 2018.

[30] D. Goudarzi and M. Rivain. How fast can higher-order masking be in software? In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 567–597. Springer, Heidelberg, Apr. / May 2017.

[31] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Heidelberg, Aug. 2003.

[32] A. Journault and F.-X. Standaert. Very high order masking: Efficient implementation and security evaluation. In W. Fischer and N. Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 623–643. Springer, Heidelberg, Sept. 2017.

[33] H. Kim, S. Hong, and J. Lim. A fast and provably secure higher-order masking of AES S-box. In B. Preneel and T. Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 95–107. Springer, Heidelberg, Sept. / Oct. 2011.

[34] A. Mathieu-Mahias. *Securisation of implementations of cryptographic algorithms in the context of embedded systems*. Theses, Université Paris-Saclay, Dec. 2021.

[35] T. S. Messerges. Securing the AES finalists against power analysis attacks. In B. Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 150–164. Springer, Heidelberg, Apr. 2001.

[36] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A side-channel analysis resistant description of the AES S-box. In H. Gilbert and H. Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 413–423. Springer, Heidelberg, Feb. 2005.

[37] J. M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25:365–374, 1971.

[38] T. Prest, D. Goudarzi, A. Martinelli, and A. Passelègue. Unifying leakage models on a Rényi day. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 683–712. Springer, Heidelberg, Aug. 2019.

[39] E. Prouff and M. Rivain. Masking against side-channel attacks: A formal security proof. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, Heidelberg, May 2013.

[40] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede. Consolidating masking schemes. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 764–783. Springer, Heidelberg, Aug. 2015.

[41] M. Rivain and E. Prouff. Provably secure higher-order masking of AES. In S. Mangard and F.-X. Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 413–427. Springer, Heidelberg, Aug. 2010.

[42] A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, Nov. 1979.

[43] J. von zur Gathen and J. Gerhard. *Modern computer algebra (2. ed.)*. Cambridge University Press, 2003.

[44] J. Wang, P. K. Vadnala, J. Großschädl, and Q. Xu. Higher-order masking in practice: A vector implementation of masked AES for ARM NEON. In K. Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 181–198. Springer, Heidelberg, Apr. 2015.

[45] Y. Wang and X. Zhu. A fast algorithm for the Fourier transform over finite fields and its VLSI implementation. *IEEE Journal on Selected Areas in Communications*, 6(3):572–577, April 1988.

A General Definitions

We give hereafter the formal security properties of *uniformity* and *input-output separation* for general gadgets for binary operations which generalize the definitions given for the refresh gadgets in Subsection 3.1.

Definition 10 (Uniformity)**.** Let $\boldsymbol{v} \in \mathbb{K}^n$ and g be any binary operation $g : (x, y) \in \mathbb{K}^2 \mapsto z \in \mathbb{K}$. A $\boldsymbol{v}$-gadget G of g is *uniform* if for every $\boldsymbol{x} \in \mathbb{K}^n$ and $\boldsymbol{y} \in \mathbb{K}^n$, the output $G(\boldsymbol{x}, \boldsymbol{y})$ is a uniform $\boldsymbol{v}$-linear sharing of $g(\langle \boldsymbol{v}, \boldsymbol{x} \rangle, \langle \boldsymbol{v}, \boldsymbol{y} \rangle)$.

Let g be any binary operation $g : (x, y) \in \mathbb{K}^2 \mapsto z \in \mathbb{K}$ and G be a $\boldsymbol{v}$-gadget for g. In the following, we shall say that a triple of vector $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in (\mathbb{K}^n)^3$ is *admissible* for G if there exists a random tape $\boldsymbol{\rho}$ such that $\boldsymbol{z} = G^{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{z})$. For an admissible triple $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and a set $\mathcal{W} \subseteq [|G|]$, the wire distribution of G in $\mathcal{W}$ *induced* by $(\boldsymbol{x}, \boldsymbol{y})$, denoted $G_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, is the random vector $G^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y})$, *i.e.* the tuple of wire values for the wire indexes in $\mathcal{W}$, obtained for a uniform drawing of $\boldsymbol{\rho}$ among the set $\{\boldsymbol{\rho} \in \mathbb{K}^q \,;\, G^{\boldsymbol{\rho}}_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{z}\}$.

Definition 11 (IOS)**.** Let $\boldsymbol{v} \in \mathbb{K}^n$ and g be any binary operation $g : (x, y) \in \mathbb{K}^2 \mapsto z \in \mathbb{K}$. A $\boldsymbol{v}$-gadget G of g is said *t-input-output separative (t-IOS)*, if it is uniform and if for every admissible triple $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and every set of wires $\mathcal{W} \subseteq [s]$ with $|\mathcal{W}| \leq t$, there exists a (two-stage) simulator $\mathcal{S}_{G,\mathcal{W}} = \left(\mathcal{S}^{(1)}_{G,\mathcal{W}}, \mathcal{S}^{(2)}_{G,\mathcal{W}}\right)$ such that

1. $\mathcal{S}^{(1)}_{G,\mathcal{W}}(\perp) = (I_1, I_2, J)$ where $I_1, I_2, J \subseteq [n]$, with $|I_1| + |I_2| \leq |\mathcal{W}|$ and $|J| \leq |\mathcal{W}|$;

2. $\mathcal{S}^{(2)}_{G,\mathcal{W}}(\boldsymbol{x}_{|I_1}, \boldsymbol{y}_{|I_2}, \boldsymbol{z}_{|J}) \overset{\text{id}}{=} G_{\mathcal{W}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.

A $\boldsymbol{v}$-gadget is simply said to be *IOS* if it is *n-IOS*.

B Random Probing and Noisy Leakage Security

Ajtai introduced in [2] the so-called *random probing model* in which an adversary cannot choose a fixed number of arbitrary wires but instead the leaking wires are chosen independently, each with some given probability p. This model is therefore similar to the *binary erasure channel* used in coding theory and information theory.

Definition 12 (Random Probing Model)**.** Let $\varepsilon, p \in [0,1]$. A randomized arithmetic circuit $\widehat{C}$ is (p,ε)-*random probing secure* w.r.t. an encoding algorithm Encode if there exists a simulator $\mathcal{S}_{\widehat{C}}$ such that, sampling a set of wires $\mathcal{W} \subseteq \left[|\widehat{C}|\right]$, where each wire from $\widehat{C}$ belongs to $\mathcal{W}$ independently with probability p, and for every plain input $\boldsymbol{x}$, we have:

$$\mathcal{S}_{\widehat{C},\mathcal{W}}(\perp) \stackrel{\text{id}}{=} \widehat{C}_{\mathcal{W}}(\mathsf{Encode}(\boldsymbol{x})) \ . \tag{18}$$

with probability at least $1-\varepsilon$ over the random sampling of $\mathcal{W}$. A circuit compiler $(\mathsf{Compile}, \mathsf{Encode}, \mathsf{Decode})$ is (p,ε)-*random probing secure* if for every circuit C the compiled circuit $\widehat{C} = \mathsf{Compile}(C)$ is (p,ε)-random probing secure w.r.t. Encode (where p and ε might be a function of the encoding order and the circuit size).

Using the classical Chernoff's inequality, it is easy to prove that security in the region probing model implies security in the random probing model with appropriate parameters.

Proposition 2. *Let $r \in [0,1]$ and $\widehat{C}$ be a randomized circuit r-region probing secure w.r.t. an encoding algorithm* Encode *with a circuit partition $\widehat{C} \equiv (C_1, C_2, \ldots, C_m)$ where $|C_i| \geq t$ for each $i \in \{1,\ldots,m\}$. The circuit $\widehat{C}$ is (p,ε)-random probing secure w.r.t.* Encode *with*

$$p = \frac{r}{2} \text{ and } \varepsilon \leq m \cdot \exp(-p \cdot t/3).$$

Proof. More generally, let $\delta \geq 1$ and let us suppose that $p \leq r/(1+\delta)$. Let $\mathcal{W} \subseteq \left[|\widehat{C}|\right]$ be a wire set where each wire from $\widehat{C}$ belongs to $\mathcal{W}$ independently with probability p. For each subcircuit C_i of the circuit partition $\widehat{C} \equiv (C_1, C_2, \ldots, C_m)$, we denote $\mathcal{W}_i = \mathcal{W} \cap C_i$ for $i \in \{1,\ldots,m\}$. By Chernoff's bound, we have $\Pr[|\mathcal{W}_i| \geq (1+\delta)p|C_i|] \leq \exp(-\delta^2 p|C_i|/(\delta+2))]$ for $i \in \{1,\ldots,m\}$ and therefore

$$\Pr\left(|\mathcal{W}_i| \geq \lceil r|C_i| \rceil\right) \ \leq \ \exp(-\delta p \cdot t/3)$$

for $i \in \{1,\ldots,m\}$. Using the union bound over the m sets $\mathcal{W}_i$ for $i \in \{1,\ldots,m\}$ and setting $\delta = 1$, we get the claimed bounds. □

Noisy Leakage Model. The noisy leakage model was first formalized by Prouff and Rivain [39]. In this model, the adversary may learn information about every single wire; however, instead of learning exactly the value x of a wire (as in the probing model), the adversary learns a randomized function $f(x)$ of x. The generality of the noisy leakage model allows it to encompass several real-life instances of leakages, making it a very realistic leakage model. However, due to its somewhat analytical nature, security proofs are notoriously hard to do in it.

Thankfully, is has been shown that the noisy leakage and random probing models are equivalent: one implication was proven [25] and improved in [38], the other one was proven in [38]. As a workaround to the complexity of the noisy leakage model, one can therefore establish security proofs in the random probing model and subsequently transfer them in the noisy leakage model using the equivalence between the two. Proposition 2 provides an additional tool to this proof strategy and, combined with results of [25, 38], imply that a compiler secure in the region probing model is secure in the noisy leakage model.

Security of the GJR$^+$ scheme. We have the following corollary of Theorem 3.

Corollary 2. *If the FFT circuit is linear and under the t_n^{R}-IOS property of the refresh gadget, the GJR$^+$ compiler is (p_n, ε_n)-random probing secure with*

$$p_n = \frac{1}{2} \max_{t \leq t_n^{\mathsf{R}}} \min\left(\frac{(n-1)-6t}{2 \cdot |\mathsf{FFT}_n|} \ , \ \frac{t}{|G_n^{\mathsf{R}}|} \right) \tag{19}$$

and

$$\varepsilon_n = (2N^{\oplus} + 4N^{\otimes}) \cdot \exp(-n \cdot p_n) + 3N^{\otimes} \cdot \frac{n}{|\mathbb{F}|} \ , \tag{20}$$

where $N^{\oplus}$ (resp. $N^{\otimes}$) is the number of addition gates (resp. multiplication gates) in the original circuit.

Proof. Let C be an arithmetic circuit composed of $N^{\oplus}$ addition (or linear) gates and $N^{\otimes}$ multiplication gates and let $\widehat{C}$ be the corresponding compiled circuit output from the GJR$^+$ compiler. We consider a split of $\widehat{C}$ into different regions following the region probing security reduction of the GJR$^+$ scheme. Specifically, each addition (or sharewise) gadget and each refresh gadget consists in a single region while each multiplication gadget gives rise to three different regions: the first block (*i.e.* the circuit considered in Lemma 1), the internal refresh, and the second block (*i.e.* the circuit considered in Lemma 2). We hence get a total of $N_{reg} = 2N^{\oplus} + 4N^{\otimes}$ regions in $\widehat{C}$ (counting the refresh gadgets). In the random probing model, each wire leaks independently of the other wires with a given probability, denoted p_n here. We show hereafter that with overwhelming probability over the distribution of the indexes of the leaking wires, the full random probing leakage can be perfectly simulated. More specifically, we exhibit two failure events $\mathcal{F}_1$ and $\mathcal{F}_2$ that may prevent such a perfect simulation. Whenever none of these failure events occur, a perfect simulation is achieved in the same way as in the region probing security reduction given above.

The first failure event $\mathcal{F}_1$ occurs when the number of leaking wires in at least one region R exceeds the threshold $2p_n|R|$ where $|R|$ denotes the number of wires in R. Applying the Chernoff bound, this occurs in a given region with probability lower than

$$\exp\Big(-\frac{p_n \cdot |R|}{3}\Big) \leq \exp(-n \cdot p_n) \ , \tag{21}$$

where the inequality holds since the minimal value of $|R|$ is obtained for the addition gadget with $|R| = 3n$. We deduce that the first failure event occurs with probability

$$\Pr(\mathcal{F}_1) \leq N_{reg} \cdot \exp(-n \cdot p_n) \ . \tag{22}$$

Provided that the first failure event does not occur, the random probing simulator needs to simulate less than $2p_n|R|$ wires per region that is $r_n|R|$ wires per region where $r_n = 2p_n$ is as defined in Theorem 3 for $t_n^{\mathsf{FFT}} = (n-1)$. This translates to simulating at most $t_n^{\mathsf{FFT}} = (n-1)$ probes in each FFT circuit through the above security reduction. Our second failure event $\mathcal{F}_2$ occurs whenever the $n-1$ leaking wires within a FFT circuit cannot be perfectly simulated. Using Lemma 2 from [29] (see Appendix C) and thanks to the linearity of the FFT circuit, we have that for any choice of $n-1$ leaking wires from the FFT circuit, the probability that the leaking wires cannot be simulated is lower than $n/|\mathbb{F}|$. Besides the linearity of the FFT circuit, the only requirement for this upper bound to apply is that the choice of the leaking wires is made independently of ω, which occurs in the random probing model since the placement of the probes is randomly draw (with leakage probability p for each wire) independently of the random generation of ω. We deduce that the second failure event $\mathcal{F}_2$ occurs with probability

$$\Pr(\mathcal{F}_2) \leq 3N^{\otimes} \cdot \frac{n}{|\mathbb{F}|} \ . \tag{23}$$

Whenever no failure event occur, the leaking wires within each FFT circuit can be perfectly simulated which –following the above reduction– implies that the overall leaking wires can be perfectly simulated. It results that the GJR$^+$ scheme is (p_n, ε_n)-random probing secure with

$$\varepsilon_n \leq \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) \leq \Pr(\mathcal{F}_1) + \Pr(\mathcal{F}_2) \ , \tag{24}$$

which together with Equation 22 and Equation 23 concludes the proof. □

In the above random probing security proof, the tolerated number of probes in an FFT circuit is $t_n^{\mathsf{FFT}} = (n-1)$ which implies $\gamma \approx 1$ in Equation 11. On the other hand, our refresh gadget is such that $\beta = 3$. Assuming a FFT algorithm satisfying $|\mathsf{FFT}_n| = \alpha \cdot n \log n$, we finally get

$$p_n = \frac{r_n}{2} \approx \left(\frac{1}{4\alpha + 36}\right) \cdot \frac{1}{\log n} \ . \tag{25}$$

For instance, the NTT algorithm used in the original GJR scheme satisfies $\alpha \approx 6$, which gives $p_n \approx \frac{1}{60 \log n}$.

C Lemmas from [29]

We recall hereafter the key lemmas from [29] for the security of the GJR scheme. Let FFT_n be a linear FFT circuit on field $\mathbb{K}$ taking an ω-encoding as input. Every value v taken by a wire of FFT_ncan be expressed as

$$v = \sum_{i=0}^{n-1} \alpha_i a_i \tag{26}$$

where the α_i's are constant coefficients over $\mathbb{K}$. The lemmas use the following notation

$$[v] = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})^{\mathsf{T}} \tag{27}$$

for the column vector of coefficients of such a wire value. Similarly, we shall denote $[a] = (1, \omega, \omega^2, \ldots, \omega^{n-1})^{\mathsf{T}}$ for an ω-encoding $(a_1, \ldots, a_n)$ of a variable a since we have $a = \sum_{i=0}^{n-1} \omega^i a_i$ by definition. Moreover, $[v_0, v_1, \ldots, v_\ell]$ shall denote the matrix with column vectors $[v_0]$, $[v_1]$, $\ldots$, $[v_\ell]$.

Lemma 3 (Lemma 1 of [29])**.** *Let $v_1, v_2, \ldots, v_\ell$ be the values taken by $\ell < n$ wires of FFT_n on input a uniform ω-encoding of a variable a. The distribution of the tuple $(v_1, v_2, \ldots, v_\ell)$ is statistically independent of a iff*

$$[a] \notin \mathrm{span}([v_1, \ldots, v_\ell]) \ , \tag{28}$$

where span$(\cdot)$ *refers to the linear span of the input matrix.*

Lemma 4 (Lemma 2 of [29])**.** *Let ω be a uniform random element in $\mathbb{K}^*$. And let $v_1, v_2, \ldots, v_\ell$ be a set of $\ell < n$ intermediate variables of FFT_n on input an ω-encoding of a variable a. We have:*

$$\Pr\left([a] \in \mathrm{span}([v_1, \ldots, v_\ell])\right) \leq \frac{\ell}{|\mathbb{K}| - 1} < \frac{n}{|\mathbb{K}|} \ , \tag{29}$$

where the above probability is taken over a uniform random choice of ω.

From these two lemmas, the values taken by any set of $\ell < n$ wires of FFT_n can be perfectly simulated without knowledge of a. The simulation simply works by taking a random a, picking a random ω-encoding of a, and evaluating the wires $v_1, \ldots, v_\ell$ accordingly leads to a perfect simulation. According to Lemma 2 of [29] such a simulation fails with probability lower than $n/|\mathbb{K}|$.

D Complements on Gao-Mateer additive FFT

D.1 Algorithms used in the Gao-Mateer additive FFT

This section presents the detailed algorithms used in the Gao-Mateer additive FFT.

Algorithm 6 $\mathsf{TaylorExpansion}(f, N)$

Require: $f \in \mathbb{F}[x]$ of degree $< N$
Ensure: The Taylor expansion $\{h_0, \ldots, h_{N/2-1}\}$ of f w.r.t. $(x^2 - x)$

if $N \leq 2$ **then**
  return $\{f\}$
$k \leftarrow N/4$
$g_0 \leftarrow \sum_{i=0}^{2k} f_i \cdot x^i$ ▷ g_0 is the low-order coefficients of f
$g_1 \leftarrow \sum_{i=0}^{2k} f_{i+2k} \cdot x^i$ ▷ g_1 is the high-order coefficients of f
for $i = 0, \ldots, k-1$ **do**
  $g_1(x) \leftarrow g_1(x) \oplus g_{1,i+k} \cdot x^i$
  $g_0(x) \leftarrow g_0(x) \oplus g_{1,i} \cdot x^{i+k}$
$V_0 \leftarrow \mathsf{TaylorExpansion}(g_0, N/2)$ ▷ Recursive call
$V_1 \leftarrow \mathsf{TaylorExpansion}(g_1, N/2)$ ▷ Recursive call
return $V_0 \| V_1$

Algorithm 7 $\mathsf{Fold}(B)$

Require: A basis $B = \{\beta_0, \ldots, \beta_{m-1}\} \subset \mathbb{F}^m$
Ensure: Two new basis $G, D \subset \mathbb{F}^{m-1}$

for $i = m-2, \ldots, 0$ **do**
  $\gamma_i \leftarrow \beta_i / \beta_{m-1}$
  $\delta_i \leftarrow \gamma_i^2 - \gamma_i$
$G \leftarrow \{\gamma_0, \ldots, \gamma_{m-2}\}$
$D \leftarrow \{\delta_0, \ldots, \delta_{m-2}\}$
return G, D

Algorithm 8 $\mathsf{GaoMateerFFT}(f, m, B)$

Require: A polynomial $f \in \mathbb{F}[x]$ of degree $< 2^m$, a basis $B = \{\beta_0, \ldots, \beta_{m-1}\} \subset \mathbb{F}^m$
Ensure: The additive FFT of f over the span $\langle B \rangle$

if $m = 1$ **then**
  return $\{f(0), f(\beta_0)\}$.
Compute $g(x) = f(\beta_{m-1}x)$.
Compute the Taylor expansion of g, i.e. $g(x) = \sum_i (g_{i,0} + g_{i,1}x) \cdot (x^2 - x)^i$.
Let $g_0 \leftarrow \sum_i g_{i,0} x^i$ and $g_1 \leftarrow \sum_i g_{i,1} x^i$.
Let $G, D \leftarrow \mathsf{Fold}(B)$, and $k = 2^{m-1}$.
$\{u_0, u_1, \ldots, u_{\frac{N}{2}-1}\} \leftarrow \mathsf{GaoMateerFFT}(g_0, m-1, D)$
$\{v_0, v_1, \ldots, v_{\frac{N}{2}-1}\} \leftarrow \mathsf{GaoMateerFFT}(g_1, m-1, D)$
for $i = 0, \ldots, \frac{N}{2} - 1$ **do**
  $w_i \leftarrow u_i \oplus G[i] \cdot v_i$
  $w_{\frac{N}{2}+i} \leftarrow w_i \oplus v_i$
return $\{w_0, w_1, \ldots, w_{n-1}\}$

D.2 Half-FFT

Our improved (half-)FFT works with a self-folding basis and its iterated foldings. We clarify the notation: let $B = \{\beta_0, \ldots, \beta_{m-1}\}$ be a self-folding basis, and $B_k = \{\beta_{m-k}, \ldots, \beta_{m-1}\}$ for $k \in \{1, \ldots, m\}$. We also denote $G_k = \{\beta_{m-k-1}, \ldots, \beta_{m-2}\}$, so that $|B_k| = |G_k| = k$, and $(G_{k-1}, B_{k-1}) = \mathsf{Fold}(B_k)$.

For our purposes, it is adequate to precompute $G_j[i]$ for $1 \leq j < m$ and $0 \leq i < 2^j$. In this section, we describe how various algorithmic tricks can help to improve the additive FFT of [27] in our setting. First, we observe that from Proposition 1, it holds that for each B_k, the $(k-1)$-th element of B_k is 1, hence the step 3 of Algorithm 8 becomes unnecessary; in total, this saves us $N \log N$ scalar multiplications. Second, we note that since each G_k is a subset of G_{m-1}, storing precomputed multiplication tables for the step 10 does not require to store $\frac{N}{2} + \frac{N}{4} + \cdots + 1 = N - 1$ precomputed tables anymore, but rather $\frac{N}{2} - 1$ (not having to store the multiplication by zero saves an additional element). Added to the removal of step 3, this more than divides by two the number of precomputed tables. As a final algorithmic optimization, we make full use of the fact that for polynomial multiplications, half the inputs of the FFT's call are zero coefficients which leads to speed up to the computations by a factor two compared to a regular additive FFT. This optimized FFT is described in Algorithm 9.

Algorithm 9 $\mathsf{HalfFFT}(f, B)$

Require: $f \in \mathbb{F}[x]$ of degree $< N = 2^{m-1}$, a self-folding basis $B = \{\beta_0, \ldots, \beta_{m-1}\}$ and its iterated foldings $G_{m-1}, \ldots, G_1$.
Ensure: The evaluation of f over the $\mathbb{F}_2$-linear space generated by B_m

1: $W \leftarrow 0^N$
2: **if** $m = 2$ **then** ▷ Bottom of the recursion
3: $\quad W_0 \leftarrow f_0$ ▷ Since $\forall j, G_j[0] = 0$, it simplifies the computation
4: $\quad W_1 \leftarrow f_0 \oplus G_1[1] \cdot f_1$
5: $\quad W_2 \leftarrow f_0 \oplus f_1$
6: $\quad W_3 \leftarrow W_1 \oplus f_1$
7: $\quad$ **return** W
8: $T \leftarrow \mathsf{TaylorExpansion}(f, N/2)$ ▷ T is a list of $N/4$ polynomials $T_i = T_{i,0} \oplus T_{i,1}x$
9: $g_0 \leftarrow \sum_{i=0}^{N/4-1} T_{i,0} \cdot x^i$
10: $g_1 \leftarrow \sum_{i=0}^{N/4-1} T_{i,1} \cdot x^i$
11: $U \leftarrow \mathsf{HalfFFT}(g_0, \{\beta_1, \ldots, \beta_{m-1}\})$ ▷ Recursive call
12: $V \leftarrow \mathsf{HalfFFT}(g_1, \{\beta_1, \ldots, \beta_{m-1}\})$ ▷ Recursive call
13: **for** $i = 0, \ldots, N/2$ **do**
14: $\quad W_i \leftarrow U_i \oplus G_{m-1}[i] \cdot V_i$
15: $\quad W_{i+N/2} \leftarrow W_i \oplus V_i$
16: **return** W

The optimizations we just described apply in a similar way to the inverse additive FFT. The only difference here is that we cannot exploit anymore the fact that half of the polynomial coefficients are zero: indeed, since we use the additive FFT to multiply two polynomials of degree at most $n - 1 = N/2 - 1$, we expect the degree of the result to be at most $N - 2$. Hence, simply applying the inverse of each operations of Algorithm 9 in reverse order would yield an incorrect result. Our optimized inverse additive FFT is described in Algorithm 10.

Algorithm 10 $\mathsf{InverseFFT}(W, B)$

Require: A self-folding basis $B = \{\beta_0, \ldots, \beta_{m-1}\}$ and its foldings $G_{m-1}, \ldots, G_1$, the evaluation W of f over the $\mathbb{F}_2$-linear space generated by B
Ensure: A polynomial $f \in \mathbb{F}[x]$ of degree $< N = 2^m$

if $m = 2$ **then** ▷ Bottom of the recursion
  $u \leftarrow W_1 \oplus W_2$
  $v \leftarrow W_1 \oplus W_3$
  $f_0 \leftarrow W_0 \oplus (G_1[0] \cdot u)$
  $f_1 \leftarrow (G_1[1] \cdot (v)) \oplus f_0 \oplus W_0 \oplus u$
  $f_2 \leftarrow f_1 \oplus v$
  $f_3 \leftarrow f_1 \oplus f_2 \oplus W_0 \oplus W_2$
  return $f_0 \oplus f_1 \cdot x \oplus f_2 \cdot x^2 \oplus f_3 \cdot x^3$
for $i = 0, \ldots, N/2$ **do**
  $V_i \leftarrow W_i \oplus W_{i+N/2}$
  $U_i \leftarrow W_i \oplus G_{m-1}[i] \cdot V_i$
$U \leftarrow (U_i)_i, V \leftarrow (V_i)_i$
$g_0 \leftarrow \mathsf{InverseFFT}(U, \{\beta_1, \ldots, \beta_{m-1}\})$ ▷ Recursive call
$g_1 \leftarrow \mathsf{InverseFFT}(V, \{\beta_1, \ldots, \beta_{m-1}\})$ ▷ Recursive call
$T \leftarrow (g_{0,i} + g_{1,i} \cdot x)_{0 \le i \le N/2}$
$f \leftarrow \mathsf{inverseTaylorExpansion}(T, N)$
return f

D.3 Complexity Analysis

In this section, we provide the complexity of our new algorithms. We do not change the algorithm for the Taylor expansion, which cost remains the same: $N(\log N - 1)/2$ field additions and no multiplication.

Let $A(N)$ and $M(N)$ denote the number of field additions and multiplications entailed by Algorithm 9:

- We note that $M(4) = 1$ and that $M(N) = 2 \cdot M(N/2) + N/2$. From these two facts, one can show by induction that $M(N) = \frac{1}{2} \cdot N \cdot \log_2(N) - \frac{3}{4} \cdot N$.
- Similarly, we have $A(4) = 3$ and $A(N) = \frac{1}{4} N \log \frac{N}{2} - \frac{1}{4} N + 2A(N/2) + N$. It follows by induction that $A(N) = \frac{1}{8} \cdot N \cdot (\log_2(N))^2 + \frac{5}{8} \cdot N \cdot \log_2(N) - N$.

Using the same techniques, one can show that Algorithm 10 performs $\frac{1}{4} \cdot N \cdot (\log_2(N))^2 + \frac{3}{4} \cdot \log_2(N)$ additions and $\frac{1}{2} \cdot \log_2(N) - \frac{1}{2} \cdot N$ multiplications.

We note that thanks to our use of self-folding bases, we divide by about 4 the number of multiplications compared to [27]. Similarly, exploiting the degree of the input polynomials divides the number of additions in Algorithm 8 by two. For all practical purposes, our algorithms even perform better than the specialized additive FFT described in [27, Section IV]. This FFT requires slightly more multiplications ($\frac{1}{2} N \log_2(N)$) than ours, and the number of additions ($N \log_2(N) + \frac{1}{2} N \log_2(N) \log_2 \log_2 N$), while asymptotically better than ours, remains larger than the number of additions in Algorithm 10 for any $N \le 256$. Since the specialized FFT works only for m a power of two, it will only start to outperform Algorithm 10 for $N \ge 2^{16}$, corresponding to a masking order $n = 32768$. It also remains to be studied whether this algorithm can exploit the degree of the input polynomials to reduce the number of additions, as done by Algorithm 8. We note that just like in [27], all the multiplications in our algorithms are field multiplications by a constant, which can be stored in precomputed tables.

Appendix F

Random Probing Security: Verification, Composition, Expansion & New Constructions

Hereafter is appended the full version of our paper [BCP+20], joint work with Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff and Abdul Rahman Taleb, published at **CRYPTO 2020**.

Random Probing Security: Verification, Composition, Expansion and New Constructions

Sonia Belaïd[1], Jean-Sébastien Coron[2], Emmanuel Prouff[3,4], Matthieu Rivain[1], and Abdul Rahman Taleb[1]

[1] CryptoExperts, France
[2] University of Luxembourg
[3] ANSSI, France
[4] Sorbonne Universités, UPMC Univ Paris 06, POLSYS, UMR 7606, LIP6, F-75005, Paris, France

{sonia.belaid,matthieu.rivain,abdul.taleb}@cryptoexperts.com
[5] jean-sebastien.coron@uni.lu
[6] emmanuel.prouff@ssi.gouv.fr

Abstract. The masking countermeasure is among the most powerful countermeasures to counteract side-channel attacks. Leakage models have been exhibited to theoretically reason on the security of such masked implementations. So far, the most widely used leakage model is the *probing model* defined by Ishai, Sahai, and Wagner at (CRYPTO 2003). While it is advantageously convenient for security proofs, it does not capture an adversary exploiting full leakage traces as, *e.g.*, in horizontal attacks. Those attacks target the multiple manipulations of the same share to reduce noise and recover the corresponding value. To capture a wider class of attacks another model was introduced and is referred to as the *random probing model*. From a leakage parameter p, each wire of the circuit leaks its value with probability p. While this model much better reflects the physical reality of side channels, it requires more complex security proofs and does not yet come with practical constructions.
In this paper, we define the first framework dedicated to the random probing model. We provide an automatic tool, called VRAPS, to quantify the random probing security of a circuit from its leakage probability. We also formalize a composition property for secure random probing gadgets and exhibit its relation to the *strong non-interference* (SNI) notion used in the context of probing security. We then revisit the expansion idea proposed by Ananth, Ishai, and Sahai (CRYPTO 2018) and introduce a compiler that builds a random probing secure circuit from small base gadgets achieving a *random probing expandability* property. We instantiate this compiler with small gadgets for which we verify the expected properties directly from our automatic tool. Our construction can tolerate a leakage probability up to 2^{-8}, against 2^{-25} for the previous construction, with a better asymptotic complexity.

Keywords: Compiler, Masking, Automated verification, Random probing model

1 Introduction

Most cryptographic algorithms are assumed to be secure against *black-box* attacks where the adversary is limited to the knowledge of some inputs and outputs to recover the manipulated secrets. However, as revealed in the late nineties [21], when implemented on physical devices, they become vulnerable to the more powerful *side-channel attacks* which additionally exploit the physical emanations such as temperature, time, power consumption, electromagnetic radiations.

As such attacks may only require cheap equipment and can be easily mounted in a short time interval, the community had to adapt quickly by looking for efficient countermeasures. The most widely deployed approach to counteract side-channel attacks was simultaneously introduced in 1999 by Chari et al. [12] and by Goubin and Patarin [18] and is now called *masking*. Basically, the idea is to split each sensitive variable x of the implementation into n shares such that $n-1$ of them

are generated uniformly at random and the last one is computed as the combination of x and all the previous shares according to some group law $*$. When $*$ is the (bitwise) addition, we talk about *linear sharing* (aka Boolean masking). The adversary thus needs to get information on all the shares of x to recover information on the sensitive value. This countermeasure is really simple to implement for linear operations which are simply applied on each share separately. However, things are getting trickier for non-linear operations where it is impossible to compute the result without combining shares.

To reason about the security of masked implementations, the community introduced leakage models. One of the most broadly used is the *probing model*, introduced by Ishai, Sahai, and Wagner [20]. In a nutshell, a circuit is claimed to be t-probing secure if the exact values of any set of t intermediate variables do not reveal any information on the secrets. As leakage traces are assumed to reveal noisy functions of the manipulated data, this model is motivated by the difficulty to recover information from the combination of t variables from their noisy functions in masking schemes (as t grows). Nevertheless, the probing model fails to capture the huge amount of information resulting from the leakage of all manipulated data, and in particular from the repeated manipulation of identical values (see horizontal attacks in [7]). Therefore, after a long sequence of works building and analyzing masking schemes with respect to their security in the probing model [25, 15, 9], the community is now looking for security in more practical models.

The *noisy leakage model* was originally considered by Chari et al. in [12] and was later formalized by Prouff and Rivain in [24] as a specialization of the *only computation leaks* model [23] in order to better capture the reality of the physical leakage. Informally, a circuit is secure in the noisy leakage model if the adversary cannot recover the secrets from a noisy function of each intermediate variable of the implementation. While realistic, this model is not convenient for security proofs, and therefore masking schemes continued to be verified in the probing model relying on the *not tight* reduction that was formally established by Duc, Dziembowski, and Faust [17].

The latter reduction actually came with an intermediate leakage model, called *random probing model*, to which the security in the noisy leakage model reduces to. In the random probing model, each intermediate variable leaks with some constant leakage probability p. A circuit is secure in this model if there is a negligible probability that these leaking wires actually reveal information on the secrets. It is worth noting that this notion advantageously captures the 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 side-channel noise might not be customizable by the implementer.

Only a few constructions [1, 3, 2] tolerate a constant leakage probability. These three constructions are conceptually involved and their practical instantiation is not straightforward. The first one from Ajtai et al. and its extension [3] are based on expander graphs. The tolerated probability is not made explicit. The third work [2] is based on multi-party computation protocols and an expansion strategy; the tolerated probability is around 2^{-26} and for a circuit with $|C|$ gates, the complexity is $\mathcal{O}(|C| \cdot \mathsf{poly}(\kappa))$ for some parameter κ but the polynomial is not made explicit.

Following the long sequence of works relying on the probing security, formal tools have recently been built to supervise the development of masking implementations proven secure in the probing model. Namely, verification tools are now able to produce a security proof or identify potential attacks from the description of a masked implementation at up to some masking orders (i.e., < 5) [4, 14, 11]. In the same vein, compilers have been built to automatically generate masked

implementations at any order given the high level description of a primitive [5, 11, 10]. Nevertheless, no equivalent framework has yet been proposed to verify the security of implementations in the random probing model.

Our contributions. In this paper, we aim to fill this huge gap by providing a framework to verify, compose, and build random probing secure circuits from simple gadgets. Our contributions are three-fold.

Automatic verification tool. As a first contribution, we define a verification method that we instantiate in a tool to automatically exhibit the random probing security parameters of any small circuit defined with addition and multiplication gates whose wires leak with some probability p. In a nutshell, a circuit is (p, f)-random probing secure if it leaks information on the secret with probability $f(p)$, where $f(p)$ is the *failure probability function*. From these notations, our tool named VRAPS (for Verifier of Random Probing Security), based on top of a set of rules that were previously defined to verify the probing security of implementations [4], takes as input the description of a circuit and outputs an upper bound on the failure probability function. While it is limited to small circuits by complexity, the state-of-the-art shows that verifying those circuits can be particularly useful in practice (see e.g. the `maskVerif` tool [4]), for instance to verify gadgets and then deduce global security through composition properties and/or low-order masked implementations. The source code of VRAPS is publicly available.[7]

Composition and expanding compiler. We introduce a composition security property to make gadgets composable in a global random probing secure circuit. We exhibit the relation between this new *random probing composability* (RPC) notion and the *strong non-interference* (SNI) notion which is widely used in the context of probing security [5]. Then, we revisit the modular approach of Ananth, Ishai, and Sahai [2] which uses an expansion strategy to get random probing security from a multi-party computation protocol. We introduce the *expanding compiler* that builds random probing secure circuits from small base gadgets. We formalize the notion of *random probing expandability* (RPE) and show that a base gadget satisfying this notion can be securely used in the expanding compiler to achieve arbitrary/composable random probing security. As a complementary contribution, our verification tool, VRAPS, is extended to verify the newly introduced RPC and RPE properties.

Instantiation. We instantiate the expanding compiler with new constructions of simple base gadgets that fulfill the desired RPE property, which is verified by VRAPS. For a security level κ, our instantiation achieves a complexity of $\mathcal{O}(\kappa^{7.5})$ and tolerates a constant leakage probability $p \approx 0.0045 > 2^{-8}$. In comparison, and as a side contribution, we provide a precise analysis of the construction from [2] and show that it achieves an $\mathcal{O}(\kappa^{8.2})$ complexity for a much lower tolerated leakage probability ($p \approx 2^{-26}$). Finally, we note that our framework probably enables more efficient constructions based on different base gadgets; we leave such optimizations open for future works.

2 Preliminaries

Along the paper, $\mathbb{K}$ 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

[7] See `https://github.com/CryptoExperts/VRAPS`

$\boldsymbol{x}|_I = (x_i)_{i\in I}$. 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 Circuit Compilers

In this paper, an *arithmetic circuit* over a field $\mathbb{K}$ is a labeled directed acyclic graph whose edges are *wires* and vertices are *arithmetic gates* processing operations over $\mathbb{K}$. We consider three types of arithmetic gate:

- an addition gate, of fan-in 2 and fan-out 1, computes an addition over $\mathbb{K}$,
- a multiplication gate, of fan-in 2 and fan-out 1, computes a multiplication over $\mathbb{K}$,
- a copy gate, of fan-in 1 and fan-out 2, outputs two copies of its input.

A *randomized arithmetic circuit* is equipped with an additional type of gate:

- a random gate, of fan-in 0 and fan-out 1, outputs a fresh uniform random value of $\mathbb{K}$.

A (randomized) arithmetic circuit is further formally composed of input gates of fan-in 0 and fan-out 1 and output gates of fan-in 1 and fan-out 0. Evaluating an ℓ-input m-output circuit C consists in writing an input $\boldsymbol{x} \in \mathbb{K}^\ell$ in the input gates, processing the gates from input gates to output gates, then reading the output $\boldsymbol{y} \in \mathbb{K}^m$ from the output gates. This is denoted by $\boldsymbol{y} = C(\boldsymbol{x})$. During the evaluation process, each wire in the circuit is assigned with a value on $\mathbb{K}$. We call the tuple of all these wire values a *wire assignment* of C (on input $\boldsymbol{x}$).

Definition 1 (Circuit Compiler). *A* circuit compiler *is a triplet of algorithms* $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows:*

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

These three algorithms satisfy the following properties:

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

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

- ***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 Linear Sharing and Gadgets

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

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

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

Definition 2 (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 $i \in [\ell]$.*

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.

Definition 3 (Standard Circuit Compiler). *Let $\lambda \in \mathbb{N}$ be some security parameter and let $n = \mathrm{poly}(\lambda)$. Let G_{add}, G_{mult} and G_{copy} be n-share gadgets respectively for the addition, multiplication and copy over $\mathbb{K}$. The* standard circuit compiler *with* sharing order n *and* base gadgets *G_{add}, G_{mult}, G_{copy} is the circuit compiler $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ satisfying the following:*

1. *The input encoding Enc is an n-linear encoding.*
2. *The output decoding Dec is the n-linear decoding mapping LinDec.*
3. *The circuit compilation CC consists in replacing each gate in the original circuit by an n-share gadget with corresponding functionality (either G_{add}, G_{mult} or G_{copy}), and each wire by a set of n wires carrying a n-linear sharing of the original wire. If the input circuit is a randomized arithmetic circuit, each of its random gates is replaced by n random gates, which duly produce a n-linear sharing of a random value.*

For such a circuit compiler, the correctness and efficiency directly holds from the correctness and efficiency of the gadgets G_{add}, G_{mult} and G_{copy}.

2.3 Random Probing Leakage

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

In order to formally define the random-probing leakage of a circuit, we shall consider two 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 $\boldsymbol{x}$.

We can now formally define the random probing leakage of a circuit.

Definition 4 (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*

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

Remark 1. By convention the output wires of C (*i.e.* the wires incoming output gates) are excluded by the $\mathsf{LeakingWires}$ sampler whereas the input wires of C (*i.e.* the wires connecting input gates to subsequent gates) are included. Namely the output set $\mathcal{W}$ of $\mathsf{LeakingWires}(C, p)$ does not include any output wire label of C. This is because when composing several circuits (or gadgets), the output wires of a circuit are the input wires in a next circuit. This also relates to the widely admitted *only computation leaks* assumption [23]: the processing of a gate leaks information on its input values (and information on the output can be captured through information on the input).

Definition 5 (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* $\boldsymbol{x} \in \mathbb{K}^{\ell}$*:*

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

A circuit compiler $(\mathsf{CC}, \mathsf{Enc}, \mathsf{Dec})$ *is* (p, ε)-random probing secure *if for every (randomized) arithmetic circuit* C *the compiled circuit* $\widehat{C} = \mathsf{CC}(C)$ *is* $(p, |C| \cdot \varepsilon)$*-random probing secure where* $|C|$ *is the size of original circuit.*

As in [2] we shall consider a *simulation with abort*. In this approach, the simulator first calls the leaking-wires sampler to get a set $\mathcal{W}$ and then either aborts (or fails) with probability ε or outputs the exact distribution of the wire assignment corresponding to $\mathcal{W}$. Formally, for any leakage probability $p \in [0,1]$, the simulator Sim is defined as

$$\mathsf{Sim}(\widehat{C}) = \mathsf{SimAW}(\widehat{C}, \mathsf{LeakingWires}(\widehat{C}, p)) \tag{2}$$

where SimAW, the *wire assignment simulator*, either returns $\perp$ (simulation failure) or a perfect simulation of the requested wires. Formally, the experiment

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

leads to

$$\Pr[out = \perp] = \varepsilon \quad \text{and} \quad \big(out \mid out \neq \perp\big) \stackrel{\text{id}}{=} \big(\mathsf{AssignWires}(\widehat{C}, \mathcal{W}, \mathsf{Enc}(\boldsymbol{x})) \mid out \neq \perp\big) \ . \tag{3}$$

It is not hard to see that if we can construct such a simulator SimAW for a compiled circuit $\widehat{C}$, then this circuit is (p, ε)-random probing secure.

3 Formal Verification

In this section we show how to compute the simulation failure probability $f(p)$ as a function of the leakage probability p for the base gadgets. Since even for simple gadgets this tasks would be difficult to perform by hand, we use a formal verification tool to compute $f(p)$.

3.1 Simulation Failure probability

We first derive an upper bound on the simulation failure probability as a function of the leakage probability p. We consider a compiled circuit $\widehat{C}$ composed of s wires labeled from 1 to s and a simulator SimAW as defined in previous section. For any sub-set $\mathcal{W} \subseteq [s]$ we denote by $\delta_{\mathcal{W}}$ the value defined as follows:

$$\delta_{\mathcal{W}} = \begin{cases} 1 & \text{if } \mathsf{SimAW}(\widehat{C}, \mathcal{W}) = \perp, \\ 0 & \text{otherwise.} \end{cases}$$

The simulation failure probability ε in (3) can then be explicitly expressed as a function of p. Namely, we have $\varepsilon = f(p)$ with f defined for every $p \in [0, 1]$ by:

$$f(p) = \sum_{\mathcal{W} \subseteq [s]} \delta_{\mathcal{W}} \cdot p^{|\mathcal{W}|} \cdot (1-p)^{s-|\mathcal{W}|} \ . \tag{4}$$

Letting c_i be the number of sub-sets $\mathcal{W} \subseteq [s]$ of cardinality i for which $\delta_{\mathcal{W}} = 1$, namely for which the simulation fails, we have $c_i = \sum_{|\mathcal{W}|=i} \delta_{\mathcal{W}}$ and hence (4) simplifies to

$$f(p) = \sum_{i=1}^{s} c_i \cdot p^i \cdot (1-p)^{s-i} \ . \tag{5}$$

For any circuit $\widehat{C}$ achieving t-probing security, the values $\delta_{\mathcal{W}}$ with $|\mathcal{W}| \leq t$ are equal to zero, and therefore the corresponding c_i's are zero, which implies the following simplification:

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

Moreover, by definition, the coefficients c_i satisfy:

$$c_i \leqslant \binom{s}{i} \tag{6}$$

which leads to the following upper-bound for $f(p)$:

$$f(p) \leqslant \sum_{i=t+1}^{s} \binom{s}{i} \cdot p^i \cdot (1-p)^{s-i} \ .$$

Example: evaluating $f(p)$ for the 2-share ISW multiplication gadget (ISW-2). This gadget takes at input the 2-sharings (x_0, x_1) and (y_0, y_1) of x and y respectively, and outputs the 2-sharing

$$(z_0, z_1) = (x_0 \cdot y_0 + r_0, x_1 \cdot y_1 + r_0 + x_0 \cdot y_1 + x_1 \cdot y_0)$$

where r_0 is a random value. The processing is composed of the following intermediate results, where each variable is assigned a wire:

$$\begin{array}{cccc} c_0 = x_0 * y_0 & z_0 = c_0 + r_0 & c_1 = x_1 * y_1 & c_2 = c_1 + r_0 \\ c_3 = x_0 * y_1 & c_4 = c_2 + c_3 & c_5 = x_1 * y_0 & z_1 = c_4 + c_5 \end{array}$$

When the same variable is involved as input of several operations, a copy gadget (with 1 input wire and 2 output wires) is applied to duplicate it. Consequently, each new use of the same variable as input of an operation adds 2 wires to the final count of overall wires. It may be checked that the circuit corresponding to ISW-2 is composed of 21 wires, excluding the 2 output wires. Since it is 1-SNI but not 2-SNI, every set with a single wire can be simulated, which is not the case for all pairs of wires. Actually, 51 among the latter pairs cannot be simulated. If we continue the test for the sets of cardinality from 3 to 21, we get the following list of coefficients c_i, $1 \leq i \leq 21$, computed with the verification tool described in the next section: 0, 51, 754, 4827, 18875, 52994, 115520, 203176, 293844, 352702, 352715, 293930, 203490, 116280, 54264, 20349, 5985, 1330, 210, 21, 1. Directly injecting these coefficients in (5) gives the expression of $f(p)$ for ISW-2.

3.2 Verification method

For any compiled circuit $\widehat{C}$ and any simulator defined as in Section 2.3, the computation of the function $f(p)$ for any probability p essentially amounts to computing the values of the coefficients c_i's appearing in (5). If no assumption is made on the circuit, this task seems difficult to carry out by hand. Actually, it may be checked that an exhaustive testing of all the possible tuples of wires for a gadget with s wires has complexity lower bounded by 2^s, which gives 2^{21} for a simple gadget like the ISW multiplication gadget with two shares per input. Here, we introduce a verification tool, that we call VRAPS, enabling to automatically test the perfect simulation for any set of wires of size lower than or equal to some threshold β. The role of the latter threshold is simply to control the verification duration (which can be long if the circuit to test is complex). Our tool implicitly defines a simulator that may fail with a probability $\varepsilon = f(p)$ satisfying (5).

The verification tool takes as input the representation of a compiled circuit $\widehat{C}$ and a test parameter β, and outputs the list of coefficients c_1, ..., c_β. It is assumed that $\widehat{C}$ takes as input the n-linear encoding $\mathsf{Enc}(\boldsymbol{x})$ of vector $\boldsymbol{x} = (x_1, \ldots, x_\ell)$ defined in $\mathbb{K}^\ell$. It is moreover assumed that $\widehat{C}$ is composed of s wires respectively denoted by w_1, ..., w_s. In the following, we consider s-tuples in the form of $u = (u_1, \ldots, u_s) \in \{0,1\}^s$ together with the common rule $u' \subset u$ iff for every $i \in [s]$, $u'_i = 1 \Rightarrow u_i = 1$ (in this case u' will be said to be included in u). An s-tuple u for which there exists an assignment of the wires in $\mathcal{W} = \{w_i;\, i \in [s], u_i = 1\}$ such that the simulation fails shall be called a *failure tuple*. Such a tuple shall be said to be *incompressible* if no tuple $t' \subset t$ is a failure tuple. The main idea of the proposed verification tool is to test the simulation failure only on incompressible failure tuples whose Hamming weight ranges from 1 to β. The steps are described in Algorithm 1.

Algorithm 1 Verification tool

Input: a compiled circuit $\widehat{C}$ with s wires and a threshold $\beta \leqslant s$
Output: a list of β coefficients c_1, ..., c_β

1: $\ell_p \leftarrow []$ ▷ will be used to store a list of failure tuples
2: $\boldsymbol{c} \leftarrow (0, \ldots, 0)$ ▷ will be used to store the output coefficients
3: **for** $h = 1$ to β **do**
4: $\quad \ell_h \leftarrow \mathsf{listTuples}(s,h)$ ▷ list of s-tuples of Hamming weight h
5: $\quad (\ell_h^p, \ell_h^{f_1}) \leftarrow \mathsf{eliminateFromSmaller}(\ell_h, \ell_p)$ ▷ select tuples including an incompressible failure tuple
6: $\quad \ell_h^{f_2} \leftarrow \mathsf{failureTest}(\widehat{C}, \ell_h^p)$ ▷ identify failure tuples in ℓ_h^p
7: $\quad \ell_p \leftarrow \ell_p \cup \ell_h^{f_2}$ ▷ update list of incompressible failure tuples
8: $\quad \boldsymbol{c} \leftarrow \mathsf{updateCoeffs}(\boldsymbol{c}, \ell_h^{f_1} \cup \ell_h^{f_2})$ ▷ update coefficients
9: **end for**
10: **return** $\boldsymbol{c}$

The function $\mathsf{listTuples}$ outputs the list of all s-tuples with Hamming weight h with $h \in [s]$. The function $\mathsf{eliminateFromSmaller}$ takes as input the list ℓ_h of current tuples of Hamming weight h and the list of incompressible failure tuples ℓ_p. It returns two lists:

- $\ell_h^{f_1}$: the elements of ℓ_h which are not incompressible (*i.e.* which include at least one element from ℓ_p)
- ℓ_h^p: the elements of ℓ_h which are incompressible (*i.e.* $\ell_h \backslash \ell_h^{f_1}$)

The function $\mathsf{failureTest}$ takes as input the second list ℓ_h^p and checks if a perfect simulation can be achieved for each wire family $\mathcal{W}$ corresponding to a tuple in ℓ_h^p. Basically, for each wire family, a sequence of rules taken from $\mathsf{maskVerif}$ [4] is tested to determine whether $\mathcal{W}$ can be perfectly simulated. It outputs $\ell_h^{f_2}$, the list of incompressible failure s-tuples of Hamming weight h. In a nutshell, each wire w_i in $\mathcal{W}$ is considered together with the algebraic expression $\varphi_i(\cdot)$ describing its assignment by $\widehat{C}$ as a function of the circuit inputs and the random values returned by the random gates, then the three following rules are successively and repeatedly applied on all the wires families $\mathcal{W}$ (see [4] for further details):

rule 1: check whether all the expressions $\varphi_i(\cdot)$ corresponding to wires w_i in $\mathcal{W}$ contain all the shares of at least one of the coordinates of $\boldsymbol{x}$;
rule 2: for every $\varphi_i(\cdot)$, check whether a random r (*i.e.* an output of a random gate) additively masks a sub-expression e (which does not involve r) and appears nowhere else in the other $\varphi_j(\cdot)$

with $j \neq i$; in this case replace the sum of the so-called sub-expression and r by r, namely $e + r \leftarrow r$;

rule 3: apply mathematical simplifications on the tuple.

Function updateCoeffs takes as input the current array of β coefficients c_i for $1 \leqslant i \leqslant \beta$ and the concatenation of both lists of potential failure tuples $\ell_h^{f_1}$ and $\ell_h^{f_2}$. For each failure tuple, these coefficients are updated.

Link with the tool maskVerif . This tool was introduced in [4] to automatically and formally verify higher-order masking implementations, and has further been improved to verify the t-NI and t-SNI security properties. Essentially, this tool verifies each property by analyzing the dependency of sets of fixed number of wires (say t) with a specific number of input shares. In our case, the size of the wires' sets which must be tested (to decide whether the corresponding coefficient c_i must be incremented or not) is *a priori* not bounded, or (for efficiency reasons) is bounded by a threshold β that is not a security parameter but an efficiency one. Moreover, our testing must take intermediate failures into account. Although maskVerif does not directly allows to answer our specific needs, we could have exploited its rules directly in our tool with dedicated add-ons. However we wanted to provide an easy-to-understand global tool and we therefore re-implemented the common parts (essentially those enabling to decided whether a given set of wires can be simulated or not).

Optimization 1 (grouping the wires). In most of the compiled circuits that we usually considered, several wires are always assigned the same value. Grouping those wires altogether allows us to significantly reduce the number of wires to be considered by the verification tool. Let us denote by $s^\star$ the number of groups, by α_i the size of the i-th group and by w_i a representative of the i-th group. Then, Algorithm 1 can be almost directly applied to the shortened list of $s^\star$ wires (instead of s). The single main difference is that the updateCoeffs procedure also takes into account the weights α_i when updating the coefficients c_i. For instance, considering $h = 3$, and the tuple $(1, 1, 1, 0, ..., 0)$ with respective weights $\alpha_1 = 2$ (for w_1), $\alpha_2 = 1$ (for w_2) and $\alpha_3 = 3$ (for w_3), the function would increase c_3 with 6, c_4 with 6, c_5 with 4 and c_6 with 1. The latter evaluation is performed using a recursive function which evaluates the number of partitions of an integer j into h parts with the constraints that each part should be at least one. When this optimization is applied, it may be observed that the updateCoeffs procedure also starts to update some coefficients c_i for $i > \beta$. These updated coefficients can be used as lower bounds of the final c_i values. They will be called c_i^{inf} in the rest of this paper. c_i^{sup} will be used to denote the maximal possible value for c_i, namely the binomial coefficient $\binom{s}{i}$.

Optimization 2 (using the 'longest failure tuple'). To build all the potential failure tuples, a strategy consists in exhaustively testing all the s-tuples with Hamming weight below the Hamming weight of the longest *incompressible attack tuple*. Once this set, let say $\mathcal{U}_{\text{inc}}$, has been built, the set of all potential failure tuples can be deduced by executing the following procedure:

- for one $u_{\text{inc}} \in \mathcal{U}_{\text{inc}}$ define $\mathcal{U}_{\text{failure}} = \{u \in \{0,1\}^s;\ u_{\text{inc}} \subset u\}$.
- for every new $u_{\text{inc}} \in \mathcal{U}_{\text{inc}}$, update $\mathcal{U}_{\text{failure}} = \mathcal{U}_{\text{failure}} \cup \{u \in \{0,1\}^s;\ u_{\text{inc}} \subset u\}$

Implementation. An implementation of Algorithm 1 has been developed in Python. This tool, named VRAPS, has been open sourced at:

`https://github.com/CryptoExperts/VRAPS`

Small examples. In order to illustrate our automatic verification of gadgets in the random probing model, we give the list of coefficients and the subsequent failure functions obtained for three known gadgets from the literature, namely the 2-share and 3-share multiplication gadgets introduced by Ishai, Sahai, and Wagner in [20] and a 3-share multiplication gadget from [8] with an optimal number of random variables to achieve security in the 2-probing model. Descriptions for these three gadgets are given below together with an approximation of the corresponding failure function f produced by our tool. Operations are performed according to the standard priority rules. Sharings $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the inputs, sharings $\boldsymbol{z}$ denote the outputs, and r_i are random variables. Copy gates are implicit when variables are used more than once. Hereafter $\mathcal{O}(p^5)$ is to be interpreted as p tends to 0.

2-*share ISW multiplication gadget (ISW-2):*

$$\begin{cases} z_0 = x_0 \cdot y_0 + r_0 \\ z_1 = x_1 \cdot y_1 + r_0 + x_0 \cdot y_1 + x_1 \cdot y_0 \end{cases} \Rightarrow \; f(p) = 51p^2 + 754p^3 + 4827p^4 + \mathcal{O}(p^5)$$

3-*share multiplication gadget from [8] (EC16-3):*

$$\begin{cases} z_0 = x_0 \cdot y_0 + r_0 + x_0 \cdot y_2 + x_2 \cdot y_0 \\ z_1 = x_1 \cdot y_1 + r_1 + x_0 \cdot y_1 + x_1 \cdot y_0 \\ z_2 = x_2 \cdot y_2 + r_0 + r_1 + x_1 \cdot y_2 + x_2 \cdot y_1 \end{cases} \Rightarrow \; f(p) = 1116p^3 + 44909p^4 + \mathcal{O}(p^5)$$

3-*share ISW multiplication gadget (ISW-3):*

$$\begin{cases} z_0 = x_0 \cdot y_0 + r_0 + r_1 \\ z_1 = x_1 \cdot y_0 + (x_0 \cdot y_1 + r_0) + x_1 \cdot y_1 + r_2 \\ z_2 = x_2 \cdot y_0 + (x_0 \cdot y_2 + r_1) + \\ \quad (x_2 \cdot y_1 + (x_1 \cdot y_2 + r_2)) + x_2 \cdot y_2 \end{cases} \Rightarrow \; f(p) = 1219p^3 + 55756p^4 + \mathcal{O}(p^5)$$

For our three examples, our verification tool (Algorithm 1) has been launched respectively with $\beta = s = 21$ for ISW-2, with $\beta = 9 < s = 57$ for ISW-3 and with $\beta = 13 < s = 52$ for EC16-3. In the two later cases, the missing coefficients c_i with $i > \beta$ have been either set to 0 or to $\binom{s}{i}$. This allowed us to define a lower bound f_{inf} and an upper bound f_{sup} for the functions f corresponding to ISW-3 and EC16-3. The behavior of these functions is plotted in Figures 1 to 3.

4 Composition

This section aims to provide composition properties for random-probing secure gadgets. In a nutshell, we aim to show how to build random probing secure larger circuits from specific random probing secure building blocks.

4.1 Random Probing Composability

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

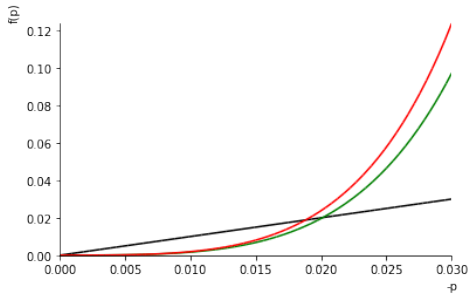

Fig. 1: Values taken by $f_{\inf}(p)$ for ISW-3 (red) and EC16-3 (green) compared to the function $p \mapsto p$ (black).

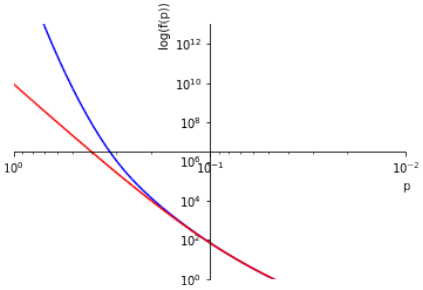

Fig. 2: Values of $\log(f_{\inf}(p))$ (red) and $\log(f_{\sup}(p))$ (blue) for ISW-3.

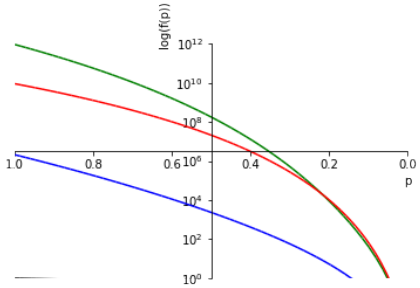

Fig. 3: For values p ranging from 1 to 0, values of $\log(f_{\inf}(p))$ for ISW-3 (red) and EC16-3 (green) together with the values of $\log(f(p))$ for ISW-2 (blue).

Definition 6 (Random Probing Composability). *Let $n, \ell, m \in \mathbb{N}$. An n-share gadget $G : (\mathbb{K}^n)^\ell \to (\mathbb{K}^n)^m$ is (t, p, ε)-*random probing composable (RPC) *for some $t \in \mathbb{N}$ and $p, \varepsilon \in [0, 1]$ if there exists a deterministic algorithm Sim_1^G and a probabilistic algorithm Sim_2^G such that for every input $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$ and for every set collection $J_1 \subseteq [n]$, ..., $J_m \subseteq [n]$ of cardinals $|J_1| \leq t$, ..., $|J_m| \leq t$, the random experiment*

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

yields

$$\Pr\left((|I_1| > t) \vee \ldots \vee (|I_\ell| > t)\right) \leq \varepsilon \tag{7}$$

and

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

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

In the above definition, the first-pass simulator Sim_1^G determines the necessary input shares (through the returned collection of sets $\boldsymbol{I}$) for the second-pass simulator Sim_2^G to produce a perfect simulation of the leaking wires defined by the set $\mathcal{W}$ together with the output shares defined by the

collection of sets $\boldsymbol{J}$. Note that there always exists such a collection of sets $\boldsymbol{I}$ since $\boldsymbol{I} = ([n], \ldots, [n])$ trivially allows a perfect simulation whatever $\mathcal{W}$ and $\boldsymbol{J}$. However, the goal of Sim_1^G is to return a collection of sets $\boldsymbol{I}$ with cardinals at most t. The idea behind this constraint is to keep the following composition invariant: for each gadget we can achieve a perfect simulation of the leaking wires plus t shares of each output sharing from t shares of each input sharing. We shall call *failure event* the event that at least one of the sets $I_1, \ldots, I_\ell$ output of Sim_1^G has cardinality greater than t. When (t, p, ε)-RPC is achieved, the failure event probability is upper bounded by ε according to (7). A failure event occurs whenever Sim_2^G requires more than t shares of one input sharing to be able to produce a perfect simulation of the leaking wires (*i.e.* the wires with label in $\mathcal{W}$) together with the output shares in $\widehat{\boldsymbol{y}}|_{\boldsymbol{J}}$. Whenever such a failure occurs, the composition invariant is broken. In the absence of failure event, the RPC notion implies that a perfect simulation can be achieved for the full circuit composed of RPC gadgets. This is formally stated in the next theorem.

4.2 Composition Security

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

Proof. Let $\mathcal{W}$ denote the leaking wires of the randomized circuit $\mathsf{CC}(C)$ with probability p for each wire. We now build a simulator Sim taking as inputs $\mathsf{CC}(C)$ and $\mathcal{W}$ and that perfectly simulates $\mathcal{W}$ with probability at least $(1 - |C| \cdot \varepsilon)$ from the simulators of the (t, p, ε)-RPC base gadgets.

We start with splitting set $\mathcal{W}$ into $|C|$ distinct subsets $\mathcal{W}_i$ for $i \in \{1, \ldots, |C|\}$ such that each $\mathcal{W}_i$ stands for the output of $\mathsf{LeakingWires}$ when applied to the i'th gadget G_i of $\mathsf{CC}(C)$ with probability p. Then, we start from end gadgets whose outputs coincide with the circuit's outputs. We execute their $\mathsf{Sim}_1^{G_i}$ with $\mathcal{W}_i$ and $J = \emptyset$, to get the sets I of required inputs. Then, we target their parents, that are gadgets whose outputs are inputs of end gadgets. For each such gadget G_i, we execute $\mathsf{Sim}_1^{G_i}$ with $\mathcal{W}_i$ and J as defined by children sets I, to get the new sets I of required inputs. The simulation goes through the circuit from bottom to top by applying the Sim_1^G simulators to get the $\mathcal{W}_i$ and I/J sets. The simulation fails if at least one set I is of cardinal greater than t. For $|C|$ gadgets, this happens with probability $1 - (1 - \varepsilon)^{|C|} \leq |C| \cdot \varepsilon$. Otherwise, the simulation runs the Sim_2^G simulators from top to bottom by randomly picking the initial $(x_i)_I$, which completes the construction of our global simulator Sim. □

4.3 Relation with Standard Probing Composition Notions

We first reformulate the Strong Non-Interference notion introduced in [5] with the formalism used for our definition of the Random Probing Composability.

Definition 7 (Strong Non-Interference (SNI)). *Let n, ℓ and t be positive integers. An n-share gadget $G : (\mathbb{K}^n)^\ell \to \mathbb{K}^n$ is t-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 $|\mathcal{W}| + |J| \leqslant t$, the following random experiment with any $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$*

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

yields

$$|I_1| \leqslant |\mathcal{W}|, \ldots, |I_\ell| \leqslant |\mathcal{W}| \tag{8}$$

and

$$out \overset{id}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}, \widehat{\boldsymbol{x}}) \ , \ \widehat{\boldsymbol{y}}|_J\right) \tag{9}$$

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

Then, we demonstrate that gadgets satisfying the t-SNI notion are also random probing composable for specific values that we explicit in the following proposition, whose proof is available in Appendix A.

Proposition 1. *Let n, ℓ and t be positive integers and let G be a gadget from $(\mathbb{K}^n)^\ell$ to $\mathbb{K}^n$. If G is t-SNI, then it is also $(t/2, p, \varepsilon)$-RPC for any probability p and ε satisfying:*

$$\varepsilon = \sum_{i=\lfloor \frac{t}{2}+1 \rfloor}^{s} \binom{s}{i} p^i (1-p)^{s-i} \ , \tag{10}$$

where s is the number of wires in G.

4.4 Verification of Gadget Composability

Our random probing verification tool (Algorithm 1) can be easily extended to define a simulator for the (t, p, ε)-random probing composability of a gadget for some t and some p. This essentially amounts to extend Algorithm 1 inputs with a multi-set $\mathcal{O}$ and to modify the $\mathsf{failureTest}$ procedure in order to test the simulation for each tuple in the input list ℓ_n^p augmented with the outputs coordinates with indices in $\mathcal{O}$. Then, our extended algorithm is called for every set $\mathcal{O}$ composed of at most t indices in each of the sets $J_1, \ldots, J_m$. The output for the call with input set $\mathcal{O}$ is denoted by $\boldsymbol{c}_\mathcal{O} = (c_1^\mathcal{O}, \ldots, c_\beta^\mathcal{O})$. For our simulator construction, the probability ε satisfies

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

where s denotes the number of wires in the tested gadget. Moreover, the c_i's satisfy $c_i = \max_\mathcal{O} c_i^\mathcal{O}$.

Example. As an illustration of the proposition, let us consider the well deployed 3-share ISW multiplication gadget $G_{\text{ISW-3}} : (\mathbb{K}^3)^2 \to (\mathbb{K}^3)$ displayed in Section 3 and satisfying 2-SNI from [5]. Considering implicit copy gadgets that are mandatory in the circuit definition when a variable is reused, the corresponding circuit contains $s = 57$ wires. From Proposition 1, this gadget is also $(1, p, \varepsilon_{\text{ISW}})$-RPC for any probability p and ε_{ISW} such that

$$\varepsilon_{\text{ISW}} = \sum_{i=2}^{57} \binom{s}{i} p^i (1-p)^{57-i}.$$

Figure 4 displays for $p \in [0, 1]$ the values taken by ε_{ISW} (in red). It also displays (in green) the values $\varepsilon'_{\text{ISW}}$ obtained by calling our verification tool on the same gadget $G_{\text{ISW-3}}$ with $\beta = 5$ (see Algorithm 1) and by replacing the missing coefficients c_i with $i > \beta$ by their upper bound $\binom{s}{i}$ (see (6)). It may be checked for small values of p the failure probability $\varepsilon'_{\text{ISW}}$ is smaller than ε_{ISW} which directly implies that the simulation induced by our verification tool is tighter than that deduced from Proposition 1.

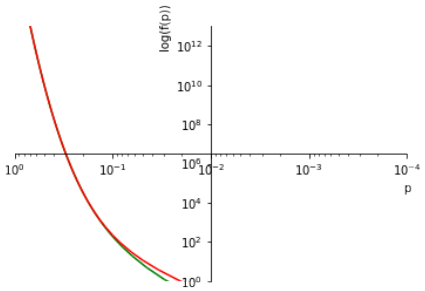

Fig. 4: Values taken by $\varepsilon_{\mathrm{ISW}}$ and $\varepsilon'_{\mathrm{ISW}}$ as a function of $p \in [0, 1]$.

5 Expansion

Constructing random-probing-secure circuit compilers with a gadget expansion strategy has been proposed by Ananth, Ishai and Sahai in [2]. Such strategy was previously used in the field of multi-party computation (MPC) with different but close security goals [13, 19]. Note that such approach is called *composition* in [2] since it roughly consists in composing a base circuit compiler several times. We prefer the terminology of *expansion* here to avoid any confusion with the notion of composition for gadgets as considered in Section 4 and usual in the literature – see for instance [5, 9, 11].

We recall hereafter the general principle of the gadget expansion strategy and provide an asymptotic analysis of the so-called *expanding circuit compiler*. Then we propose an implementation of this strategy which relies on the new notion of *gadget expandability*. In contrast, the construction of [2] relies on a t-out-n secure MPC protocol in the passive security model. The advantage of our notion is that it can be achieved and/or verified by simple atomic gadgets leading to simple and efficient constructions. After introducing the gadget expandability notion, we show that it allows to achieve random-probing security with the expansion strategy. We finally explain how to adapt the verification tool described in Section 3 to this expandability notion.

5.1 Expansion Strategy

The basic principle of the gadget expansion strategy is as follows. Assume we have three n-share gadgets G_{add}, G_{mult}, G_{copy} and denote CC the standard circuit compiler for these base gadgets. We can derive three new n^2-share gadgets by simply applying CC to each gadget: $G^{(2)}_{\mathrm{add}} = \mathsf{CC}(G_{\mathrm{add}})$, $G^{(2)}_{\mathrm{mult}} = \mathsf{CC}(G_{\mathrm{mult}})$ and $G^{(2)}_{\mathrm{copy}} = \mathsf{CC}(G_{\mathrm{copy}})$. Let us recall that this process 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. 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 $G^{(2)}_{\mathrm{add}}$, $G^{(2)}_{\mathrm{mult}}$ or $G^{(2)}_{\mathrm{copy}}$ 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 as been replaced by n^k wires carrying an (n^k)-linear sharing of the original wire. The underlying compiler is called *expanding circuit compiler* which is formally defined hereafter.

Definition 8 (Expanding Circuit Compiler). *Let* CC *be the standard circuit compiler with sharing order* n *and base gadgets* G_{add}, G_{mult}, G_{copy}. *The* expanding circuit compiler *with* expansion level k *and base compiler* CC *is the circuit compiler* $(\mathsf{CC}^{(k)}, \mathsf{Enc}^{(k)}, \mathsf{Dec}^{(k)})$ *satisfying the following:*

1. *The input encoding* $\mathsf{Enc}^{(k)}$ *is an* (n^k)*-linear encoding.*
2. *The output decoding* Dec *is the* (n^k)*-linear decoding mapping.*
3. *The circuit compilation is defined as*

$$\mathsf{CC}^{(k)}(\cdot) = \underbrace{\mathsf{CC} \circ \mathsf{CC} \circ \cdots \circ \mathsf{CC}}_{k \text{ times}}(\cdot)$$

The goal of the expansion strategy in the context of random probing security is to replace the leakage probability p of a wire in the original circuit by the failure event probability ε in the subsequent gadget simulation. If this simulation fails then one needs the full input sharing for the gadget simulation, which corresponds to leaking the corresponding wire value in the base case. The security is thus amplified by replacing the probability p in the base case by the probability ε (assuming that we have $\varepsilon < p$). If the failure event probability ε can be upper bounded by some function of the leakage probability: $\varepsilon < f(p)$ for every leakage probability $p \in [0, p_{\max}]$ for some $p_{\max} < 1$, then the expanding circuit compiler with expansion level k shall result in a security amplification as

$$p = \varepsilon_0 \xrightarrow{f} \varepsilon_1 \xrightarrow{f} \cdots \xrightarrow{f} \varepsilon_k = f^{(k)}(p) \ ,$$

which for an adequate function f (*e.g.* $f : p \mapsto p^2$) provides exponential security. In order to get such a security expansion, the gadgets must satisfy a stronger notion than the composability notion introduced in Section 4 which we call *random probing expandability*; see Section 5.3 below.

5.2 Asymptotic Analysis of the Expanding Compiler

In this section we show that the asymptotic complexity of a compiled circuit $\widehat{C} = \mathsf{CC}^{(k)}(C)$ is $|\widehat{C}| = \mathcal{O}\big(|C| \cdot \kappa^e\big)$ for security parameter κ, for some constant e that we make explicit.

Let us denote by $\boldsymbol{N} = (N_a, N_c, N_m, N_r)^\mathsf{T}$ the column vector of gate counts for some base gadget G, where N_a, N_c, N_m, N_r stands for the number of addition gates, copy gates, multiplication gates and random gates respectively. We have three different such vectors

$$\begin{aligned}
\boldsymbol{N}_{\text{add}} &\doteq (N_{\text{add},a}, N_{\text{add},c}, N_{\text{add},m}, N_{\text{add},r})^\mathsf{T} \\
\boldsymbol{N}_{\text{mult}} &\doteq (N_{\text{mult},a}, N_{\text{mult},c}, N_{\text{mult},m}, N_{\text{mult},r})^\mathsf{T} \\
\boldsymbol{N}_{\text{copy}} &\doteq (N_{\text{copy},a}, N_{\text{copy},c}, N_{\text{copy},m}, N_{\text{copy},r})^\mathsf{T}
\end{aligned}$$

for the gate counts respectively in the base addition gadget G_{add}, in the base multiplication gadget G_{mult} and in the base copy gadgets G_{copy}. Let us define the 4×4 square matrix $\boldsymbol{M}$ as

$$\boldsymbol{M} = \big(\boldsymbol{N}_{\text{add}} \mid \boldsymbol{N}_{\text{copy}} \mid \boldsymbol{N}_{\text{mult}} \mid \boldsymbol{N}_{\text{rand}}\big) \quad \text{with} \quad \boldsymbol{N}_{\text{rand}} = (0,0,0,n)^\mathsf{T} \ ,$$

where the definition $\boldsymbol{N}_{\text{rand}}$ holds from the fact that the standard circuit compiler replaces each random gate by n random gates.

It can be checked that applying the standard circuit compiler with base gadgets G_{add}, G_{mult} and G_{copy} to some circuit C with gate-count vector $\boldsymbol{N}_C$ gives a circuit $\widehat{C}$ with gate-count vector $\boldsymbol{N}_{\widehat{C}} = \boldsymbol{M} \cdot \boldsymbol{N}_C$. It follows that the kth power of the matrix $\boldsymbol{M}$ gives the gate counts for the level-k gadgets as:

$$\boldsymbol{M}^k = \underbrace{\boldsymbol{M} \cdot \boldsymbol{M} \cdots \boldsymbol{M}}_{k \text{ times}} = \left(\boldsymbol{N}_{\text{add}}^{(k)} \mid \boldsymbol{N}_{\text{copy}}^{(k)} \mid \boldsymbol{N}_{\text{mult}}^{(k)} \mid \boldsymbol{N}_{\text{rand}}^{(k)}\right) \quad \text{with} \quad \boldsymbol{N}_{\text{rand}}^{(k)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ n^k \end{pmatrix}$$

where $\boldsymbol{N}_{\text{add}}^{(k)}$, $\boldsymbol{N}_{\text{mult}}^{(k)}$ and $\boldsymbol{N}_{\text{copy}}^{(k)}$ are the gate-count vectors for the level-k gadgets $G_{\text{add}}^{(k)}$, $G_{\text{mult}}^{(k)}$ and $G_{\text{copy}}^{(k)}$ respectively. Let us denote the eigen decomposition of $\boldsymbol{M}$ as $\boldsymbol{M} = \boldsymbol{Q} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{Q}^{-1}$, we get

$$\boldsymbol{M}^k = \boldsymbol{Q} \cdot \boldsymbol{\Lambda}^k \cdot \boldsymbol{Q}^{-1} \quad \text{with} \quad \boldsymbol{\Lambda}^k = \begin{pmatrix} \lambda_1^k & & & \\ & \lambda_2^k & & \\ & & \lambda_3^k & \\ & & & \lambda_4^k \end{pmatrix}$$

where λ_1, λ_2, λ_3, λ_4 are the eigenvalues of $\boldsymbol{M}$. We then obtain an asymptotic complexity of

$$|\widehat{C}| = \mathcal{O}\big(|C| \cdot (\lambda_1^k + \lambda_2^k + \lambda_3^k + \lambda_4^k)\big) = \mathcal{O}\big(|C| \cdot \max(\lambda_1, \lambda_2, \lambda_3, \lambda_4)^k\big)$$

for a compiled circuit $\widehat{C} = \mathsf{CC}^{(k)}(C)$ (where the constant in the $\mathcal{O}(\cdot)$ depends on $\boldsymbol{Q}$ and shall be fairly small).

Interestingly, if multiplication gates are solely used in the multiplication gadget (*i.e.* $N_{\text{add},m} = N_{\text{copy},m} = 0$) which is the case in the constructions we consider in this paper, it can be checked that (up to some permutation) the eigenvalues satisfy

$$(\lambda_1, \lambda_2) = \mathsf{eigenvalues}(\boldsymbol{M}_{ac}) \ , \quad \lambda_3 = N_{\text{mult},m}^k \quad \text{and} \quad \lambda_4 = n^k$$

where $\boldsymbol{M}_{ac}$ is the top left 2×2 block matrix of $\boldsymbol{M}$ *i.e.*

$$\boldsymbol{M}_{ac} = \begin{pmatrix} N_{\text{add},a} & N_{\text{copy},a} \\ N_{\text{add},c} & N_{\text{copy},c} \end{pmatrix} \ .$$

We finally get

$$|\widehat{C}| = \mathcal{O}\big(|C| \cdot N_{\max}^k\big) \quad \text{with} \quad N_{\max} = \max(\mathsf{eigenvalues}(\boldsymbol{M}_{ac}), N_{\text{mult},m}) \ . \tag{11}$$

In order to reach some security level $\varepsilon = 2^{-\kappa}$ for some target security parameter κ and assuming that we have a security expansion $p \to f^{(k)}(p)$, the expansion level k must be chosen so that $f^{(k)}(p) \leq 2^{-\kappa}$. In practice, the function f is of the form

$$f : p \mapsto \sum_{i \geq d} c_i \, p^i \leq (c_d + \mathcal{O}(p)) \, p^d \ .$$

where $\mathcal{O}(p)$ is to be interpredted as p tends to 0. In the rest of this paper, we shall say that such a function has *amplification order* d.

The upper bound $f(p) \leq c'_d\, p^d$ with $c'_d = c_d + \mathcal{O}(p)$ implies $f^{(k)}(p) < (c'_d\, p)^{d^k}$. Hence, to satisfy the required security $f^{(k)}(p) \leq 2^{-\kappa}$ while assuming $c'_d\, p < 1$, the number k of expansions must satisfy:

$$k \geqslant \log_d(\kappa) - \log_d(-\log_2(c'_d\, p)) \ .$$

We can then rewrite (11) as

$$|\widehat{C}| = \mathcal{O}\big(|C| \cdot \kappa^e\big) \quad \text{with} \quad e = \frac{\log N_{\max}}{\log d} \ . \tag{12}$$

5.3 Random Probing Expandability

In the evaluation of random probing composability, let us recall that the failure event in the simulation of a gadget means that more that t shares from one of its inputs are necessary to complete a perfect simulation. For a gadget to be expandable we need slightly stronger notions than random probing composability. As first requirement, a two-input gadget should have a failure probability which is independent for each input. This is because in the base case, each wire as input of a gate leaks independently. On the other hand, in case of failure event in the child gadget, the overall simulator should be able to produce a perfect simulation of the full output (that is the full input for which the failure occurs). To do so, the overall simulator is given the clear output (which is obtained from the simulation of the base case) plus any set of $n-1$ output shares. This means that whenever the set J is of cardinal greater than t, the gadget simulator can replace it by any set J' of cardinal $n-1$.

Definition 9 (Random Probing Expandability). *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*

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{13}$$

   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*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_{J'}\big) \tag{14}$$

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 lower than ε (and the Sim_2^G simulator is then simply given $\widehat{x}|_I$ where $\widehat{x}$ is the single input sharing). For the sake of completeness, and since we only focus in $2 \to 1$ and $1 \to 2$ gadgets in this paper, the RPE definition for the $1 \to 2$ case is given in Appendix B.

It is not hard to check that the above expandability notion is stronger that the composability notion introduced in Section 4. Formally, we have the following reduction:

Proposition 2. *Let $f = \mathbb{R} \to \mathbb{R}$ and $n \in \mathbb{N}$. Let G be an n-share gadget. If G is (t, f)-RPE then G is (t, f')-RPC, with $f'(\cdot) = 2 \cdot f(\cdot)$.*

Proof. We consider a (t, f)-RPE n-share gadget $G : \mathbb{K}^n \times \mathbb{K}^n \to \mathbb{K}^n$. The $(t, 2 \cdot f)$-random composability property is directly implied by the (t, f)-random probing expandability by making use of the exact same simulators and observing that

$$\Pr\big((|I_1| > t) \vee (|I_2| > t)\big) \leq \Pr(|I_1| > t) + \Pr(|I_2| > t) = 2 \cdot \varepsilon.$$

The case of $1 \to 2$ gadgets is even more direct. □

5.4 Expansion Security

Definition 9 of random probing expandability is valid for base gadgets. For level-k gadgets $G^{(k)} = \mathsf{CC}^{(k-1)}(G)$ where $G \in \{G_{\text{add}}, G_{\text{mult}}, G_{\text{copy}}\}$ is a base gadget, we provide a generalized definition of random probing expandability.

Adequate subsets of $[n^k]$. We first define the notion of "adequate" subsets of $[n^k]$, instead of only bounded subsets. For this we define recursively a family $S_k \in \mathcal{P}([n^k])$, where $\mathcal{P}([n^k])$ denotes the set of all subsets of $[n^k]$, as follows:

$$S_1 = \{I \in [n],\ |I| \leq t\}$$
$$S_k = \{(I_1, \ldots, I_n) \in (S_{k-1} \cup [n^{k-1}])^n, I_j \in S_{k-1}\ \forall\ j \in [1, n] \text{ except at most } t\}$$

In other words, a subset I belongs to S_k if among the n subset parts of I, at most t of them are full, while the other ones recursively belong to S_{k-1}; see Figure 9 in Appendix C.1 for an illustration with $n = 3$ and $t = 1$.

Generalized definition of Random Probing Expandability. We generalize Definition 9 as follows. At level k the input sets I_1 and I_2 must belong to S_k, otherwise we have a failure event. As in Definition 9, the simulation is performed for an output subset J' with $J' = J$ if $J \in S_k$, otherwise $J' = [n^k] \setminus \{j^\star\}$ for some $j^\star \in [n^k]$.

Definition 10 (Random Probing Expandability with $\{S_k\}_{k\in\mathbb{N}}$). *Let $f : \mathbb{R} \to \mathbb{R}$ and $k \in \mathbb{N}$. An n^k-share gadget $G : \mathbb{K}^{n^k} \times \mathbb{K}^{n^k} \to \mathbb{K}^{n^k}$ is* (S_k, 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 $(\widehat{x}, \widehat{y}) \in \mathbb{K}^{n^k} \times \mathbb{K}^{n^k}$, for every set $J \in S_k \cup [n^k]$ 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 $\mathcal{F}_1 \equiv \big(I_1 \notin S_k\big)$ and $\mathcal{F}_2 \equiv \big(I_2 \notin S_k\big)$ verify*
$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \text{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{15}$$
with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),
2. *the set J' is such that $J' = J$ if $J \in S_k$, and $J' = [n^k] \setminus \{j^\star\}$ for some $j^\star \in [n^k]$ otherwise,*
3. *the output distribution satisfies*
$$out \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_{J'}\big) \tag{16}$$

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

The notion of random probing expandability from Definition 10 naturally leads to the statement of our main theorem; the proof is given in Appendix C.1.

Theorem 2. *Let $n \in \mathbb{N}$ and $f : \mathbb{R} \to \mathbb{R}$. Let G_{add}, G_{mult}, G_{copy} be n-share gadgets for the addition, multiplication and copy on $\mathbb{K}$. Let CC be the standard circuit compiler with sharing order n and base gadgets G_{add}, G_{mult}, G_{copy}. Let $\mathsf{CC}^{(k)}$ be the expanding circuit compiler with base compiler CC. If the base gadgets G_{add}, G_{mult} and G_{copy} are (t, f)-RPE then, $G_{add}^{(k)} = \mathsf{CC}^{(k-1)}(G_{add})$, $G_{mult}^{(k)} = \mathsf{CC}^{(k-1)}(G_{mult})$, $G_{copy}^{(k)} = \mathsf{CC}^{(k-1)}(G_{copy})$ are $(S_k, f^{(k)})$-RPE, n^k-share gadgets for the addition, multiplication and copy on $\mathbb{K}$.*

The random probing security of the expanding circuit compiler can then be deduced as a corollary of the above theorem together with Proposition 2 (RPE $\Rightarrow$ RPC reduction) and Theorem 1 (composition theorem).

Corollary 1. *Let $n \in \mathbb{N}$ and $f : \mathbb{R} \to \mathbb{R}$. Let G_{add}, G_{mult}, G_{copy} be n-share gadgets for the addition, multiplication and copy on $\mathbb{K}$. Let CC be the standard circuit compiler with sharing order n and base gadgets G_{add}, G_{mult}, G_{copy}. Let $\mathsf{CC}^{(k)}$ be the expanding circuit compiler with base compiler CC. If the base gadgets G_{add}, G_{mult} and G_{copy} are (t, f)-RPE then $\mathsf{CC}^{(k)}$ is $(p, 2 \cdot f^{(k)}(p))$-random probing secure.*

5.5 Relaxing the Expandability Notion

The requirement of the RPE property that the failure events $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent might seem too strong. In practice it might be easier to show or verify that some gadgets satisfy a weaker notion. We say that a gadget is (t, f)-*weak random probing expandable* (wRPE) if the failure events verify $\Pr(\mathcal{F}_1) \leq \varepsilon$, $\Pr(\mathcal{F}_2) \leq \varepsilon$ and $\Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) \leq \varepsilon^2$ instead of (22) in Definition 9. Although being easier to achieve and to verify this notion is actually not much weaker as the original RPE. We have the following reduction of RPE to wRPE; see Appendix C.3 for the proof.

Proposition 3. *Let $f = \mathbb{R} \to [0, 0.14]$. Let $G : \mathbb{K}^n \times \mathbb{K}^n \to \mathbb{K}^n$ be an n-share gadget. If G is (t, f)-wRPE then G is (t, f')-RPE with $f'(\cdot) = f(\cdot) + \frac{3}{2} f(\cdot)^2$.*

Assume that we can show or verify that a gadget is wRPE with the following failure event probabilities

$$\Pr(\mathcal{F}_1) = f_1(p) \ , \quad \Pr(\mathcal{F}_2) = f_2(p) \quad \text{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = f_{12}(p) \ ,$$

for every $p \in [0, 1]$. Then the above proposition implies that the gadget is (p, f)-RPE with

$$f : p \mapsto f_{\max}(p) + \frac{3}{2} f_{\max}(p)^2 \quad \text{with} \quad f_{\max} = \max(f_1, f_2, \sqrt{f_{12}}) \ .$$

We shall base our verification of the RPE property on the above equation as we describe hereafter.

5.6 Verification of Gadget Expandability

We can easily adapt our automatic tool to verify the weak random probing expandability for base gadgets (Definition 9). Basically, the verification is split into two steps that we first describe for the case of addition and multiplication gadgets with fan-in 2 and fan-out 1.

In a first step, our tool computes the function f to check the (t, f)-wRPE property for output sets of shares of cardinal at most t. For 2-input gadgets, this step leads to the computation of coefficients c_i corresponding to three failure events $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_1 \wedge \mathcal{F}_2$ as defined above but restricted to output sets of shares of cardinal less than t. The process is very similar to the verification of random probing composability but requires to separate the failure events counter into failure events for the first input ($|\mathcal{I}_1| > t$), for the second input ($|\mathcal{I}_2| > t$) or for both ($(|\mathcal{I}_1| > t) \wedge (|\mathcal{I}_2| > t)$). In the following, we denote the three functions formed from the corresponding coefficients as $f_1^{(1)}$, $f_2^{(1)}$, and $f_{12}^{(1)}$.

Then, in a second step, our tool verifies that there exists at least one set of $n-1$ shares for each output, such that the simulation failure is limited by $f(p)$ for some probability $p \in [0, 1]$. In that case, it still loops on the possible output sets of shares (of cardinal $n-1$) but instead of computing the maximum coefficients, it determines whether the simulation succeeds for at least one of such sets. A failure event is recorded for a given tuple if no output sets of cardinal $n-1$ can be simulated together with this tuple from at most t shares of each input. As for the first verification step, we record the resulting coefficients for the three failure events to obtain functions $f_1^{(2)}$, $f_2^{(2)}$, and $f_{12}^{(2)}$.

From these two steps, we can deduce f such that the gadget is (t, f)-wRPE:

$$\forall p \in [0, 1], f(p) = \max(f_1(p), f_2(p), \sqrt{f_{12}(p)})$$

with

$$f_\alpha(p) = \max(f_\alpha^{(1)}(p), f_\alpha^{(2)}(p)) \qquad \text{for} \quad \alpha \in \{1, 2, 12\}$$

The computation of f for a gadget to satisfy (t, f)-weak random probing expandability is a bit trickier for copy gadgets which produce two outputs. Instead of two verification steps considering both possible ranges of cardinals for the output set of shares J, we need to consider four scenarios for the two possible features for output sets of shares J_1 and J_2. In a nutshell, the idea is to follow the first verification step described above when both J_1 and J_2 have cardinal equal or less than t and to follow the second verification step described above when both J_1 and J_2 have greater cardinals. This leads to functions $f^{(1)}$ and $f^{(2)}$. Then, two extra cases are to be considered, namely

when ($|J_1| \leq t$) and ($|J_2| > t$) and the reverse when ($|J_1| > t$) and ($|J_2| \leq t$). To handle these scenarios, our tool loops over the output sets of shares of cardinal equal or less than t for the first output, and it determines whether there exists a set of $n-1$ shares of the second output that a simulator can perfectly simulate with the leaking wires and the former set. This leads to function $f^{(12)}$ and reversely to function $f^{(21)}$. From these four verification steps, we can deduce f such that the copy gadget is (t, f)-wRPE:

$$\forall p \in [0,1], f(p) = \max(f^{(1)}(p), f^{(2)}(p), f^{(12)}(p), f^{(21)}(p)).$$

Once gadgets have been proven (t, f)-weak RPE, they are also proven to be (t, f')-RPE from Proposition 3 with $f' : p \mapsto f(p) + \frac{3}{2}f(p)^2$. Examples of such computations for 3-share gadgets are provided in Section 6.

6 New Constructions

In this section, we exhibit and analyze $(1, f)$-wRPE gadgets for the addition, multiplication, and copy (on any base field $\mathbb{K}$) to instantiate the expanding circuit compiler. These gadgets are sound in the sense that their function f has amplification order strictly greater than one. As explained in previous sections, an amplification order strictly greater than one guarantees that there exists a probability $p_{max} \in [0,1]$ such that $\forall p \leq p_{max}, f(p) \leq p$, which is necessary to benefit from the expansion. For 2-input gadgets, f is defined as the maximum between f_1, f_2, and $\sqrt{f_{12}}$. Therefore, the constraint on the amplification order also applies to the functions f_1, f_2, and $\sqrt{f_{12}}$. For the function f_{12}, this means that the amplification order should be strictly greater than two.

We start hereafter with an impossibility result, namely there are no (2-share, 2-to-1) $(1, f)$-RPE gadgets such that f has an amplification order greater than one. Then, we provide concrete instantiations of addition, multiplication, and copy gadgets based on 3 shares which successfully achieve $(1, f)$-RPE for amplification order greater than one and can be used in the expansion compiler.

6.1 About 2-Share Gadgets

Consider a gadget G with a 2-share single output $\boldsymbol{z} = (z_0, z_1)$ and two 2-share inputs $\boldsymbol{x} = (x_0, x_1)$ and $\boldsymbol{y} = (y_0, y_1)$. We reasonably assume that the latter are the outputs of gates with fan-in at most two (and not direct input shares). For G to be $(1, f)$-RPE with f of amplification order strictly greater than one, then f_{12} must be of amplification strictly greater than two. In other words, we should be able to exhibit a simulator such that one share of each input is enough to simulate anyone of the output shares and an arbitrary couple of leaking wires. But the output wire z_0 and both input gates of the second output share z_1 represent the full output and require the knowledge of both inputs to be simulated. Therefore, f_{12} has a non-zero coefficient in p and is thus not of amplification order strictly greater than two. We thus restrict our investigation to n-share gadgets, with $n \geq 3$ to instantiate our compiler.

In the upcoming gadget descriptions, notice that variables r_i are fresh random values, operations are processed with the usual priority rules, and the number of implicit copy gates can be deduced from the occurrences of each intermediate variable such that n occurrences require $n-1$ implicit copy gates. Also, the function expression below each gadget corresponds to the function obtained from our verification tool when verifying weak random probing expandability. It implies that the gadget is (t, f)-wRPE for t usually equal to one except when defined otherwise. A more complete description of each function (with more coefficients) is available in Appendix D.1.

6.2 Addition Gadgets

The most classical masked addition schemes are sharewise additions which satisfy the simpler probing security property. Basically, given two input n-sharings $\boldsymbol{x}$ and $\boldsymbol{y}$, such an addition computes the output n-sharing $\boldsymbol{z}$ as $z_1 \leftarrow x_1 + y_1$, $z_2 \leftarrow x_2 + y_2$, $\ldots$, $z_n \leftarrow x_n + y_n$. Unfortunately, such elementary gadgets do not work in our setting. Namely consider an output set of shares J of cardinality t. Then, for any n, there exists sets $\mathcal{W}$ of leaking wires of cardinality one such that no set I of cardinality $\leq t$ can point to input shares that are enough to simulate both the leaking wire and the output shares of indexes in J. For instance, given a set $J = \{1, \ldots, t\}$, if $\mathcal{W}$ contains x_{t+1}, then no set I of cardinal $\leq t$ can define a set of input shares from which we can simulate both the leaking wire and $z_1, \ldots, z_t$. Indeed, each z_i for $1 \leq i \leq t$ requires both input shares x_i and y_i for its simulation. Thus, a simulation set I would contain at least $\{1, \ldots, t\}$ and $t+1$ for the simulation of the leaking wire. I would thus be of cardinal $t+1$ which represents a failure event in the random probing expandability definition. As a consequence, such a n-share addition gadget could only be (t, f)-RPE with f with a first coefficient c_1 as defined in Section 3 strictly positive. In other words, f would be of amplification order one such that $\forall p \in [0, 1]$, $f(p) \geq p$.

In the following, we introduce two 3-share addition gadgets. From our automatic tool, both are $(1, f)$-wRPE with f of amplification order strictly greater than one. Basically, in our first addition gadget G^1_{add}, both inputs are first refreshed with a circular refreshing gadget as originally introduced in [6]:

$$\begin{aligned} G^1_{\text{add}} : z_0 &\leftarrow x_0 + r_0 + r_1 + y_0 + r_3 + r_4 \\ z_1 &\leftarrow x_1 + r_1 + r_2 + y_1 + r_4 + r_5 \qquad\qquad f_{max}(p) = \sqrt{10}p^{3/2} + \mathcal{O}(p^2) \\ z_2 &\leftarrow x_2 + r_2 + r_0 + y_2 + r_5 + r_3 \end{aligned}$$

The second addition gadget G^2_{add} simply rearranges the order of the refreshing variables:

$$\begin{aligned} G^2_{\text{add}} : z_0 &\leftarrow x_0 + r_0 + r_4 + y_0 + r_1 + r_3 \\ z_1 &\leftarrow x_1 + r_1 + r_5 + y_1 + r_2 + r_4 \qquad\qquad f_{max}(p) = \sqrt{69}p^2 + \mathcal{O}(p^3) \\ z_2 &\leftarrow x_2 + r_2 + r_3 + y_2 + r_0 + r_5 \end{aligned}$$

In each gadget, $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input sharings and $\boldsymbol{z}$ the output sharing; f_{max} additionally reports the maximum of the first non zero coefficient (as defined in Section 3) of the three functions f_1, f_2, and f_{12}, as defined in the previous section, obtained for the random probing expandability automatic verifications. A further definition of these functions can be found in Appendix D.1. Note that both gadgets G^1_{add} and G^2_{add} are built with 15 addition gates and 6 implicit copy gates.

6.3 Multiplication Gadget

We start by proving an impossibility result: no 3-share multiplication gadget composed of direct products between input shares satisfies $(1, f)$-RPE with amplification order strictly greater than one. Consider such a gadget G with two 3-input sharings $\boldsymbol{x}$ and $\boldsymbol{y}$ whose shares are directly multiplied together. Let $(x_i \cdot y_j)$ and $(x_k \cdot y_\ell)$ be two such products such that $i, j, k, \ell \in [3]$ and $i \neq k$ and $j \neq \ell$. If both results are leaking, then the leakage can only be simulated using the four input shares. Namely, $\{i, k\} \subseteq I_1$ and $\{j, \ell\} \subseteq I_2$. This scenario represents a failure since cardinals of I_1 and I_2 are both strictly greater than one. As a consequence, function f_{12} which records the

failures for both inputs is defined with a coefficient c_2 at least equal to one. Hence f_{12} is not of amplification greater than two and f cannot be of amplification order greater than one. Regular 3-share multiplication gadgets consequently cannot be used as base gadgets of our compiler.

To circumvent this issue, we build a 3-share multiplication gadget G^1_{mult} whose both inputs are first refreshed, before any multiplication is performed:

$$\begin{aligned} u_0 &\leftarrow x_0 + r_5 + r_6; & u_1 &\leftarrow x_1 + r_6 + r_7; & u_2 &\leftarrow x_2 + r_7 + r_5 \\ v_0 &\leftarrow y_0 + r_8 + r_9; & v_1 &\leftarrow y_1 + r_9 + r_{10}; & v_2 &\leftarrow y_2 + r_{10} + r_8 \end{aligned}$$

$$\begin{aligned} z_0 &\leftarrow \big(u_0 \cdot v_0 + r_0\big) + \big(u_0 \cdot v_1 + r_1\big) + \big(u_0 \cdot v_2 + r_2\big) \\ z_1 &\leftarrow \big(u_1 \cdot v_0 + r_1\big) + \big(u_1 \cdot v_1 + r_4\big) + \big(u_1 \cdot v_2 + r_3\big) \\ z_2 &\leftarrow \big(u_2 \cdot v_0 + r_2\big) + \big(u_2 \cdot v_1 + r_3\big) + \big(u_2 \cdot v_2 + r_0\big) + r_4 \end{aligned}$$

$$f_{max}(p) = \sqrt{83}p^{3/2} + \mathcal{O}(p^2)$$

6.4 Copy Gadget

We exhibit a 3-share $(1, f)$-wRPE copy gadget G^1_{copy} with f of amplification order strictly greater than one:

$$\begin{aligned} v_0 &\leftarrow u_0 + r_0 + r_1;\ w_0 \leftarrow u_0 + r_3 + r_4 \\ v_1 &\leftarrow u_1 + r_1 + r_2;\ w_1 \leftarrow u_1 + r_4 + r_5 \\ v_2 &\leftarrow u_2 + r_2 + r_0;\ w_2 \leftarrow u_2 + r_5 + r_3 \end{aligned} \qquad f_{max}(p) = 33p^2 + \mathcal{O}(p^3)$$

It simply relies on two calls of the circular refreshing from [6] on the input. This last gadget is made of 6 addition gates and 9 implicit copy gates.

6.5 Complexity and Tolerated Probability

Following the asymptotic analysis of Section 5.2, our construction yields the following instantiation of the matrix $\boldsymbol{M}$

$$\boldsymbol{M} = \begin{pmatrix} 15 & 12 & 28 & 0 \\ 6 & 9 & 23 & 0 \\ 0 & 0 & 9 & 0 \\ 6 & 6 & 11 & 3 \end{pmatrix} \tag{17}$$

with

$$\boldsymbol{M}_{ac} = \begin{pmatrix} 15 & 12 \\ 6 & 9 \end{pmatrix} \quad \text{and} \quad N_{\text{mult},m} = 9\ .$$

The eigenvalues of $\boldsymbol{M}_{ac}$ are 3 and 21, which gives $N_{\max} = 21$. We also have a random probing expandability with function f of amplification order $d = \frac{3}{2}$. Hence we get

$$e = \frac{\log N_{\max}}{\log d} = \frac{\log 21}{\log 1.5} \approx 7.5$$

which gives a complexity of $|\widehat{C}| = \mathcal{O}\big(|C| \cdot \kappa^{7.5}\big)$. Finally, it can be checked from the coefficients of the RPE functions given in Appendix D that our construction tolerates a leakage probability up to

$$p_{\max} \approx 0.0045 > 2^{-8} \ .$$

This corresponds to the maximum value p for which we have $f(p) < p$ which is a necessary and sufficient condition for the expansion strategy to apply with (t, f)-RPE gadgets.

As explained in Sec. 5.2, we can compute the new gate count vectors for each of the compiled gadgets $G^{2(k)}_{\text{add}}$, $G^{1(k)}_{\text{copy}}$, $G^{1(k)}_{\text{mult}}$ by computing the matrix $\boldsymbol{M}^k$. In Fig. 5, we plot the total number of gates $(N_{\text{a}} + N_{\text{c}} + N_{\text{m}} + N_{\text{r}})$ in each of the compiled gadgets as a function of the level k. For instance, for level $k = 9$ the number of gates in the compiled gadgets is around 10^{12}. For the latter level and assuming a leakage probability of $p = 0.0045$ (which is the maximum we can tolerate), we achieve a security of $\varepsilon \approx 2^{-76}$. On its right side, Fig. 6 plots the values taken by the function f such that the gadgets G^1_{add}, G^2_{add}, G^1_{mult} and G^1_{copy} are (t, f)-RPE.

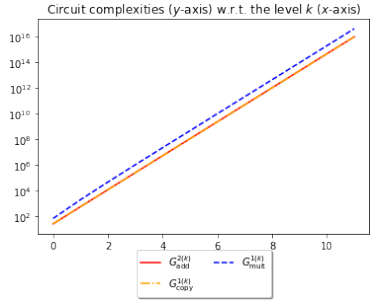

Fig. 5: Number of gates for $G^{2(k)}_{\text{add}}$, $G^{1(k)}_{\text{copy}}$, $G^{1(k)}_{\text{mult}}$ circuits with respect to the level k.

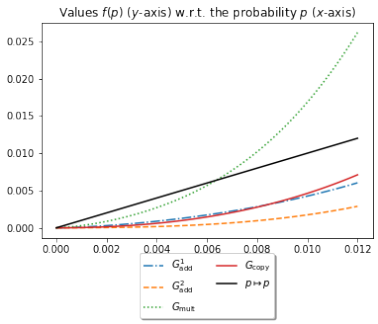

Fig. 6: Values taken by the function f for (t, f)-RPE

7 Comparison with Previous Constructions

In this section, we compare our scheme to previous constructions. Specifically, we first compare it to the well-known Ishai-Sahai-Wagner (ISW) construction and discuss the instantiation of our scheme from the ISW multiplication gadget. Then we exhibit the asymptotic complexity (and tolerated leakage probability) of the Ananth-Ishai-Sahai compiler and compare their results to our instantiation.

7.1 Comparison with ISW

The classical ISW construction [20] is secure in the t-probing model when the adversary can learn any set of t intermediate variables in the circuit, for $n = 2t + 1$ shares. This can be extended to t probes per gadget, where each gadget corresponds to a AND or XOR gate in the original circuit. Using Chernoff bound, security in the t-probing model per gadget implies security in the p-random probing model, where each wire leaks with probability p, with $p = \mathcal{O}(t/|G|)$, where $|G|$ is the gadget size. Since in ISW each gadget has complexity $\mathcal{O}(t^2)$, this gives $p = \mathcal{O}(1/t)$. Therefore, in

the p-random probing model, the ISW construction is only secure against a leakage probability $p = \mathcal{O}(1/n)$, where the number of shares n must grow linearly with the security parameter κ in order to achieve security $2^{-\kappa}$. This means that ISW does not achieve security under a constant leakage probability p; this explains why ISW is actually vulnerable to horizontal attacks [7], in which the adversary can combine information from a constant fraction of the wires.

ISW-based instantiation of the expanding compiler. In our instantiation, we choose to construct a new 3-share multiplication gadget instead of using the ISW multiplication gadget from [20]. In fact, ISW first performs a direct product of the secret shares before adding some randomness, while we proved in Section 6 that no such 3-share multiplication gadget made of direct products could satisfy $(1, f)$-RPE with amplification order strictly greater than one. Therefore the ISW gadget is not adapted for our construction with 3 shares.

Table 1 displays the output of our tool when run on the ISW gadget for up to 7 shares with different values for t. It can be seen that an amplification order strictly greater than one is only achieved for $t > 1$, with 4 or more shares. And an order of 3/2 is only achieved with a minimum of 4 shares for $t = 2$, whereas we already reached this order with our 3-share construction for $t = 1$. If we use the 4-share ISW gadget with appropriate 4-share addition and copy gadgets instead of our instantiation, the overall complexity of the compiler would be greater, while the amplification order would remain the same, and the tolerated leakage probability would be worse (recall that our instantiation tolerates a maximum leakage probability $p \approx 2^{-8}$, while 4-share ISW tolerates $p \approx 2^{-9.83}$). Clearly, the complexity of the 4-share ISW gadget $(N_a, N_c, N_m, N_r) = (24, 30, 16, 6)$ is higher than that of our 3-share multiplication gadget $(N_a, N_c, N_m, N_r) = (28, 23, 9, 11)$. In addition, using 3-share addition and copy gadgets (as in our case) provides better complexity than 4-share gadgets. Hence to reach an amplification order of 3/2, a 4-share construction with the ISW gadget would be more complex and would offer a lower tolerated leakage probability.

For higher amplification orders, the ISW gadgets with more than 4 shares or other gadgets can be studied. This is a open construction problem as many gadgets can achieve different amplification orders and be globally compared.

7.2 Complexity of the Ananth-Ishai-Sahai Compiler

The work from [2] provides a construction of circuit compiler (the AIS compiler) based on the expansion strategy described in Section 5 with a (p, ε)-composable security property, analogous to our (t, f)-RPE property. To this purpose, the authors use an (m, c)-multi-party computation (MPC) protocol Π. Such a protocol allows to securely compute a functionality shared among m parties and tolerating at most c corruptions. In a nutshell, their composable circuit compiler consists of multiple layers: the bottom layer replaces each gate in the circuit by a circuit computing the (m, c)-MPC protocol for the corresponding functionality (either Boolean addition, Boolean multiplication, or copy). The next $k - 1$ above layers apply the same strategy recursively to each of the resulting gates. As this application can eventually have exponential complexity if applied to a whole circuit C directly, the top layer of compilation actually applies the k bottom layers to each of the gates of C independently and then stitches the inputs and outputs using the correctness of the XOR-encoding property. Hence the complexity is in

$$\mathcal{O}(|C| \cdot N_{\mathrm{g}}^{k}) \ , \tag{18}$$

where $|C|$ is the number of gates in the original circuit and N_{g} is the number of gates in the circuit computing Π. The authors of [2] prove that such compiler satisfies (p, ε)-composition security

Table 1: Complexity, amplification order and maximum tolerated leakage probability of the ISW multiplication gadgets. Some leakage probabilities were not computed accurately by VRAPS for performances reasons. An interval on these probabilities is instead given by evaluating lower and upper bound functions f_{inf} and f_{sup} of $f(p)$.

# shares	Complexity $(N_{\text{a}}, N_{\text{c}}, N_{\text{m}}, N_{\text{r}})$	t	Amplification order	$\log_2$ of maximum tolerated leakage probability
3	(12, 15, 9, 3)	1	1	–
4	(24, 30, 16, 6)	1	1	–
		2	3/2	-9.83
5	(40, 50, 25, 10)	1	1	–
		2	3/2	-11.00
		3	2	-8.05
6	(60, 75, 36, 15)	1	1	–
		2	3/2	-13.00
		3	2	$[-9.83, -7.87]$
		4	2	$[-9.83, -5.92]$
7	(84, 105, 49, 21)	1	1	–
		2	3/2	$[-16.00, -14.00]$
		3	2	$[-12.00, -7.87]$
		4	5/2	$[-12.00, -2.27]$
		5	2	$[-12.00, -3.12]$

property, where p is the tolerated leakage probability and ε is the simulation failure probability. Precisely:

$$\varepsilon = N_{\text{g}}^{c+1} \cdot p^{c+1} \tag{19}$$

Equations (18) and (19) can be directly plugged into our asymptotic analysis of Sec. 5.2, with N_g replacing our $N_{\max}$ and where $c+1$ stands for our amplification order d. The obtained asymptotic complexity for the AIS compiler is

$$\mathcal{O}\big(|C| \cdot \kappa^e\big) \quad \text{with} \quad e = \frac{\log N_{\text{g}}}{\log c + 1} \ . \tag{20}$$

This is to be compared to $e = \frac{\log N_{\max}}{\log d}$ in our scheme. Moreover, this compiler can tolerate a leakage probability

$$p = \frac{1}{N_{\text{g}}^2} \ .$$

The authors provide an instantiation of their construction using an existing MPC protocol due to Maurer [22]. From their analysis, this protocol can be implemented with a circuit of $N_{\text{g}} = (4m-c) \cdot \left(\binom{m-1}{c}^2 + 2m\binom{m}{c}\right)$ gates. They instantiate their compiler with this protocol for parameters $m = 5$ parties and $c = 2$ corruptions, from which they get $N_{\text{g}} = 5712$. From this number of gates, they claim to tolerate a leakage probability $p = \frac{1}{5712^2} \approx 2^{-25}$ and our asymptotic analysis gives a complexity of $\mathcal{O}\big(|C| \cdot \kappa^e\big)$ with $e \approx 7.87$ according to (20). In Appendix E, we give a detailed analysis of the Maurer protocol [22] in the context of the AIS compiler instantiation. From our analysis, we get the following number of gates for the associated circuit:

$$N_{\text{g}} = (6m-5) \cdot \left(\binom{m-1}{c}^2 + m(2k-2) + 2k^2\right) \quad \text{where} \quad k = \binom{m}{c} \ .$$

Using the parameters $m = 5$ and $c = 2$ from the AIS compiler instantiation [2], we get $N_g = 8150$. This yields a tolerated leakage probability of $p \approx 2^{-26}$ and an exponent $e = \log 8150/\log 3 \approx 8.19$ in the asymptotic complexity $\mathcal{O}(|C| \cdot \kappa^e)$ of the AIS compiler.

These results are to be compared to the $p \approx 2^{-8}$ and $e \approx 7.5$ achieved by our construction. In either case ($N_g = 5712$as claimed in [2] or $N_g = 8150$ according to our analysis), our construction achieves a slightly better complexity while tolerating a much higher leakage probability. We stress that further instantiations of the AIS scheme (based on different MPC protocols) or of our scheme (based on different gadgets) could lead to better asymptotic complexities and/or tolerated leakage probabilities. This is an interesting direction for further research.

8 Implementation Results

In this section, we describe and report the performances of a proof-of-concept implementation of the expanding compiler with our base gadgets as well as a protected AES implementation. The source code of these implementations are publicly available at:

`https://github.com/CryptoExperts/poc-expanding-compiler`

All implementations were run on a laptop computer (Intel(R) Core(TM) i7-8550U CPU, 1.80GHz with 4 cores) using Ubuntu operating system and various C, python and sage libraries.

8.1 Circuit Compiler

First, we developed an implementation in python of a compiler CC, that given three n-share gadgets G_{add}, G_{mult}, G_{copy} and an expansion level k, outputs the compiled gadgets $G^{(k)}_{\text{add}}$, $G^{(k)}_{\text{copy}}$,$G^{(k)}_{\text{mult}}$, each as a C function. The variables' type is given as a command line argument. Table 2 shows the complexity of the compiled gadgets from Section 6 using the compiler with several expansion levels k, as well as their execution time in milliseconds when run in C on randomly generated 8-bit integers. For the generation of random variables, we consider that an efficient external random number generator is available in practice, and so we simply use the values of an incremented counter variable to simulate random gates.

Table 2: Complexity and execution time (in ms, on an Intel i7-8550U CPU) for compiled gadgets $G^{2(k)}_{\text{add}}$, $G^{1(k)}_{\text{copy}}$, $G^{1(k)}_{\text{mult}}$ from Section 6 implemented in C.

k	# shares	Gadget	Complexity (N_a, N_c, N_m, N_r)	Execution time
1	3	$G^{2(1)}_{\text{add}}$	(15, 6, 0, 6)	$1,69.10^{-4}$
		$G^{1(1)}_{\text{copy}}$	(12, 9, 0, 6)	$1,67.10^{-4}$
		$G^{1(1)}_{\text{mult}}$	(28, 23, 9, 11)	$5,67.10^{-4}$
2	9	$G^{2(2)}_{\text{add}}$	(297, 144, 0, 144)	$2,21.10^{-3}$
		$G^{1(2)}_{\text{copy}}$	(288, 153, 0, 144)	$2,07.10^{-3}$
		$G^{1(2)}_{\text{mult}}$	(948, 582, 81, 438)	$9,91.10^{-3}$
3	27	$G^{2(3)}_{\text{add}}$	(6183, 3078, 0, 3078)	$9,29.10^{-2}$
		$G^{1(3)}_{\text{copy}}$	(6156, 3105, 0, 3078)	$9,84.10^{-2}$
		$G^{1(3)}_{\text{mult}}$	(23472, 12789, 729, 11385)	$3,67.10^{-1}$

It can be observed that both the complexity and running time grow by almost the same factor with the expansion level, with multiplication gadgets being the slowest as expected. Base gadgets with $k = 1$ roughly take 10^{-4} ms, while these gadgets expanded 2 times ($k = 3$) take between 10^{-2} and 10^{-1} ms. The difference between the linear cost of addition and copy gadgets, and the quadratic cost of multiplication gadgets can also be observed through the gadgets' complexities.

8.2 AES Implementation

We describe hereafter a proof-of-concept AES implementation protected with our instantiation of the expanding compiler. We start by describing the underlying AES circuit (over $\mathbb{K} = \mathrm{GF}(256)$), followed by an analysis of the implementation in C of the complete algorithm.

AES circuit. We first describe the non-linear part of the AES, namely the sbox computation. For the field exponentiation ($x \mapsto x^{254}$ over GF(256)), we use the circuit representation of the processing proposed in [16] and presented in Fig. 7. It corresponds to the *addition chain* $(1, 2, 4, 8, 9, 18, 19, 36, 55, 72, 127, 254)$ and it has been chosen due to its optimality regarding the number of multiplications (11 in total). Each time an intermediate result had to be reused, a copy gate (marked with $\|$) has been inserted.

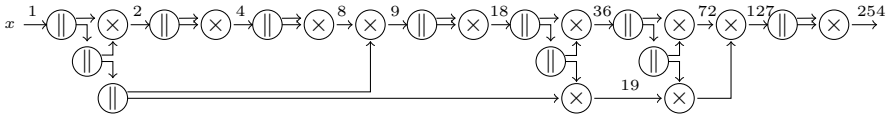

Fig. 7: Circuit for the exponentiation $x \mapsto x^{254}$.

For the second part of the sbox, the affine function is implemented according to the following equation:

$$\mathrm{Affine}(x) = (((((((207x)^2 + 22x)^2 + 1x)^2 + 73x)^2 + 204x)^2 + 168x)^2 + 238x)^2 + 5x + 99$$

with the necessary copy gates. Similarly, the inverse of the affine function is implemented for the sbox inversion as follows:

$$\mathrm{Affine}^{-1}(x) = (((((((147x)^2 + 146x)^2 + 190x)^2 + 41x)^2 + 73x)^2 + 139x)^2 + 79x)^2 + 5x + 5$$

The rest of the operations (MixColumns, ShiftRows, AddRoundKey) are considered as in a standard AES, while adding the necessary copy gates.

Gate count: Table 3 displays the gate count vectors for AES-128 encryption/decryption procedures as well as for their building blocks. The sbox (resp. sbox inversion) gate count vector was computed as the sum of the gate count vectors of both the exponentiation and affine (resp. affine inversion) functions. We recall that N_{a}, N_{c}, N_{m}, N_{r} stand for the number of addition gates, copy gates, multiplication gates, and random gates, respectively.

Using the gadgets G^2_{add}, G^1_{mult} and G^1_{copy} proposed in Sec. 6 for the compilation of the AES algorithm, we obtain the instantiation given in Equation (17) of the matrix $\boldsymbol{M}$ introduced in Sec.

Table 3: AES operations complexity.

AES Operation	**Complexity** $(N_{\mathbf{a}}, N_{\mathbf{c}}, N_{\mathbf{m}}, N_{\mathbf{r}})$
AddRoundKey (for 1 byte)	$(1, 0, 0, 0)$
SubBytes (for 1 byte)	$(8, 25, 26, 0)$
MixColumns (for all columns)	$(60, 60, 16, 0)$
ShiftRows (for all rows)	$(0, 0, 0, 0)$
AES-128 encryption	$(1996, 4540, 4304, 0)$
SubBytes Inversion (for 1 byte)	$(8, 25, 26, 0)$
MixColumns Inversion (for all columns)	$(104, 104, 36, 0)$
ShiftRows Inversion (for all rows)	$(0, 0, 0, 0)$
AES-128 decryption	$(2392, 4936, 4484, 0)$

5.2. Applying the same complexity analysis done previously on the gate count vectors, we display in Fig. 8 the total number of gates in the AES-128 encryption/decryption procedures as functions of the level k. For instance, for the same security level of 2^{-76} exhibited in Sec. 6.5 for the gadgets of Fig. 5, the AES-128 would have to be compiled at a level $k = 9$, and would count around 10^{16} gates.

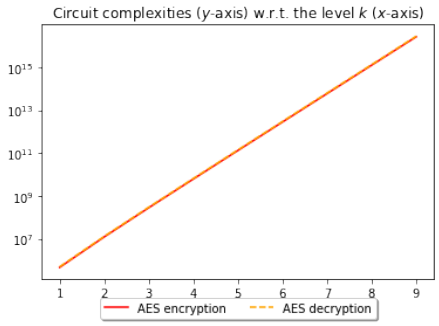

Fig. 8: Number of gates after compilation of AES-128 encryption/decryption circuits with respect to the level k.

Implementation in C: An n-share AES-128 implementation was developed in C from the above description. Compiled gadgets from Section 8.1 were used for basic operations (addition, multiplication, copy), as generated using our circuit compiler described in Sec. 8.1. We chose the C 8-bit unsigned integer type, and considered operations in $\mathsf{GF}(256)$. For the generation of random values, we assume the availability of an efficient (pseudo)random number generator, and so we simply considered the values of an incremented counter variable to simulate the cost.

Table 4 shows the AES-128 execution time on a 16-byte message with 10 pre-computed sub-keys, using compiled gadgets $G_{\text{add}}^{2(k)}$, $G_{\text{copy}}^{1(k)}$, $G_{\text{mult}}^{1(k)}$, with respect to the expansion level k and sharing order $n = 3^k$. It can be seen that the execution time increases with the expansion level with a similar growth as in Table 2. This is because the complexity of the AES circuit strongly depends on the gadgets that are used to replace each gate in the original arithmetic circuit. For example,

with our 3-share gadgets that tolerate a leakage probability of $p \approx 2^{-8}$, a 27-share ($k = 3$) AES-128 takes almost 200 milliseconds to encrypt or decrypt a message.

Table 4: Standard and n-share AES-128 execution time (in ms, on an Intel i7-8550U CPU) using compiled gadgets $G_{\text{add}}^{2(k)}$, $G_{\text{copy}}^{1(k)}$, $G_{\text{mult}}^{1(k)}$.

AES Version	**Execution Time (in ms)**	
	Encryption	**Decryption**
Standard (no sharing)	0.06	0.05
3-share ($k = 1$)	1.08	1.07
9-share ($k = 2$)	11.71	10.26
27-share ($k = 3$)	200.29	197.70

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 Annual ACM Symposium on Theory of Computing*, pages 715–724, San Jose, CA, USA, June 6–8, 2011. ACM Press.
2. Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.
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, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
4. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 457–485, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
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: 23rd Conference on Computer and Communications Security*, pages 116–129, Vienna, Austria, October 24–28, 2016. ACM Press.
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, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.
7. 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, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39, Santa Barbara, CA, USA, August 17–19, 2016. Springer, Heidelberg, Germany.
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, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

9. 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, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 397–426, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.
10. Sonia Belaïd, Pierre-Évariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff. Tornado: Automatic generation of probing-secure masked bitsliced implementations. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 311–341, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.
11. Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. Tight private circuits: Achieving probing security with the least refreshing. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 343–372, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.
12. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.
13. Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 185–202, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
14. Jean-Sébastien Coron. Formal verification of side-channel countermeasures via elementary circuit transformations. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*, volume 10892 of *Lecture Notes in Computer Science*, pages 65–82, Leuven, Belgium, July 2–4, 2018. Springer, Heidelberg, Germany.
15. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany.
16. Ivan Damgård and Marcel Keller. Secure multiparty AES. In Radu Sion, editor, *FC 2010: 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 367–374, Tenerife, Canary Islands, Spain, January 25–28, 2010. Springer, Heidelberg, Germany.
17. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.
18. Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Heidelberg, Germany.
19. Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, January 2000.
20. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.
21. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.
22. Ueli M. Maurer. Secure multi-party computation made simple (invited talk). In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 14–28, Amalfi, Italy, September 12–13, 2003. Springer, Heidelberg, Germany.
23. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.
24. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

25. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427, Santa Barbara, CA, USA, August 17–20, 2010. Springer, Heidelberg, Germany.

A Proof of Proposition 1

Proof. Since G is t-SNI there exist two simulators Sim_1^G and Sim_2^G satisfying (8) and (9) for any $J \subseteq [n]$ and any $\mathcal{W}$ satisfying $|\mathcal{W}| + |J| \leqslant t$. This is in particular true for every $\mathcal{W}$ and every J both of cardinality lower than or equal to $\frac{t}{2}$. For every set $\mathcal{W}$ such that $|\mathcal{W}| > \frac{t}{2}$ and every set J such that $|J| \leqslant \frac{t}{2}$, we modify simulator Sim_1^G to return the full set of input indices (*i.e.* $\boldsymbol{I} = [n]^\ell$). Then, the second simulator Sim_2^G is simply augmented to perfectly simulate $(\mathsf{AssignWires}(G, \mathcal{W}, \widehat{\boldsymbol{x}})\ ,\ \widehat{\boldsymbol{y}}|_J)$ from the full knowledge of the gadget inputs (which is trivially possible). By construction, for any J with $|J| \leqslant \frac{t}{2}$, the output $\boldsymbol{I} = (I_1, \ldots, I_\ell)$ of $\mathsf{Sim}_1^G(\mathcal{W}, J)$ contains at least one I_j with cardinality greater than $\frac{t}{2}$ only when $\mathcal{W}$ has cardinality strictly greater than $\frac{t}{2}$ (and in this case all the I_j's have full cardinality $[n]$). Hence, the probability $\Pr\left((|I_1| > \frac{t}{2}) \vee \ldots \vee (|I_\ell| > \frac{t}{2})\right)$ when J is a given set with $|J| \leqslant \frac{t}{2}$ and $\mathcal{W}$ is the output of $\mathsf{LeakingWires}(G, p)$ satisfies (10), which concludes the proof. □

B Random Probing Expandability for 1-to-2 Gadgets

We provide hereafter the formal definition of the RPE notion for gadgets with 1 input sharing and 2 output sharings.

Definition 11 (Random Probing Expandability). *Let $f = \mathbb{R} \to \mathbb{R}$. An n-share gadget $G : \mathbb{K}^n \to \mathbb{K}^n \times \mathbb{K}^n$ 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 $\widehat{x} \in \mathbb{K}^n$, for every pair of sets $J_1 \subseteq [n]$ and $J_2 \subseteq [n]$, and for every $p \in [0, 1]$, the random experiment*

$$\begin{aligned} &\mathcal{W} \leftarrow \mathsf{LeakingWires}(G, p) \\ &(I, J_1', J_2') \leftarrow \mathsf{Sim}_1^G(\mathcal{W}, J_1, J_2) \\ &\mathit{out} \leftarrow \mathsf{Sim}_2^G(\mathcal{W}, J_1', J_2', \widehat{x}|_I) \end{aligned}$$

ensures that

1. *the failure event probability satisfies $\Pr\left(|I| > t\right) \leq \varepsilon$ with $\varepsilon = f(p)$,*
2. *the set J_1' is such that $J_1' = J_1$ if $|J_1| \leq t$ and $J_1' \subseteq [n]$ with $|J_1'| = n - 1$ otherwise,*
3. *the set J_2' is such that $J_2' = J_2$ if $|J_2| \leq t$ and $J_2' \subseteq [n]$ with $|J_2'| = n - 1$ otherwise,*
4. *the output distribution satisfies*

$$\mathit{out} \overset{id}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}, \widehat{x})\ ,\ \widehat{y}|_{J_1'}\ ,\ \widehat{z}|_{J_2'}\right) \tag{21}$$

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

C Expandability

C.1 Illustration of the subsets S_k

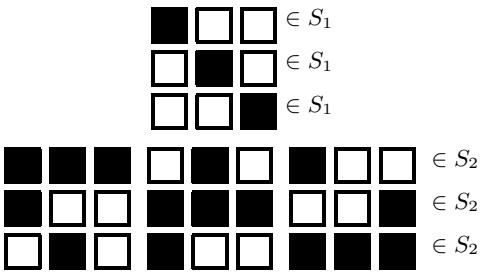

Fig. 9: Illustration of all elements of S_1 and some elements of S_2, for $n = 3$ and $t = 1$.

C.2 Proof of Theorem 2

The proof of the theorem relies on what we shall call the *assignment expansion* property. Throughout the proof we shall denote $\varepsilon_k = f^{(k)}(p)$. We call *level-k gadget* a gadget that has been expanded $k-1$ times $G^{(k)} = \mathsf{CC}^{(k-1)}(G)$ where G is a base gadget (or a level-1 gadget) among G_{add}, G_{mult}, G_{copy}.

We proceed by induction to show that the level-k gadgets are $(S_k, f^{(k)})$-RPE. The base case is one of the theorem hypotheses, namely the base gadgets G_{add}, G_{mult} and G_{copy} (*i.e.* the level-1 gadgets) are (t, f)-RPE, which is equivalent to $(\mathcal{S}_1, f)$-RPE. We must then show the induction step: assuming that the level-k gadgets are $(S_k, f^{(k)})$-RPE, show that the level-$(k+1)$ gadgets are $(S_{k+1}, f^{(k+1)})$-RPE. For the sake of simplicity, we depict our proof by assuming that all the gadgets are 2-to-1 gadget (which is actually not the case for copy gadgets). The proof mechanism for the general case (with 2-to-1 and 1-to-2 gadgets) is strictly similar but heavier on the form.

In order to show that $G^{(k+1)}$ is $(S_{k+1}, f^{(k+1)})$-RPE we must construct two simulators $\mathsf{Sim}_1^{G^{(k+1)}}$ and $\mathsf{Sim}_2^{G^{(k+1)}}$ that satisfy the conditions of Definition 10 for the set of subsets S_{k+1}. More precisely, we must construct two simulators $\mathsf{Sim}_1^{G^{(k+1)}}$ and $\mathsf{Sim}_2^{G^{(k+1)}}$ such that for every $(\widehat{x}^*, \widehat{y}^*) \in \mathbb{K}^{n^{k+1}} \times \mathbb{K}^{n^{k+1}}$, and for every set $J^* \in S_{k+1} \cup [n^{k+1}]$, the random experiment

$$\begin{aligned}
&\mathcal{W}^* \leftarrow \mathsf{LeakingWires}(G^{(k+1)}, p)\\
&(I_1^*, I_2^*, J^{*\prime}) \leftarrow \mathsf{Sim}_1^{G}(\mathcal{W}^*, J^*)\\
&\mathit{out} \leftarrow \mathsf{Sim}_2^{G}(\mathcal{W}^*, J^*, \widehat{x}^*|_{I_1^*}, \widehat{y}^*|_{I_2^*})
\end{aligned}$$

ensures that

1. the failure events $\mathcal{F}_1^* \equiv \big(I_1^* \notin S_{k+1}\big)$ and $\mathcal{F}_2^* \equiv \big(I_2^* \notin S_{k+1}\big)$ verify

$$\Pr(\mathcal{F}_1^*) = \Pr(\mathcal{F}_2^*) = \varepsilon_{k+1} \quad \text{and} \quad \Pr(\mathcal{F}_1^* \wedge \mathcal{F}_2^*) = \varepsilon_{k+1}^2 \tag{22}$$

2. the set $J^{*\prime}$ is such that $J^{*\prime} = J^*$ if $J^* \in S_{k+1}$ and $J^{*\prime} = [n^{k+1}] \setminus \{j^\star\}$ otherwise,

3. the output distribution satisfies

$$out \overset{\text{id}}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_{J^{*\prime}}\right) \tag{23}$$

where $\widehat{z} = G^{(k+1)}(\widehat{x}, \widehat{y})$.

We distinguish two cases: either $J^* \in S_{k+1}$ (normal case), or $J^* = [n^{k+1}]$ (saturated case).

Normal case: $J^* \in S_{k+1}$. By definition of the expanding compiler, we have that a level-$(k+1)$ gadget $G^{(k+1)}$ is obtained by replacing each gate of the base gadget by the corresponding level-k gadget and by replacing each wire of the base gadget by n^k wires carrying a (n^k)-linear sharing of the original wire. In particular $G^{(k+1)}$ has n^{k+1} output wires which can be split in n groups of n^k wires, each group being the output of a different $G^{(k)}$ gadget. We split the set J^* accordingly so that $J^* = J_1^* \cup \cdots \cup J_n^*$, where each set J_i^* pertains to the ith group of output wires. By definition of S_k, since $J^* \in S_{k+1}$, we must have $J_i^* \in S_k$ for all $1 \leq i \leq n$, except at most t of them for which $J_i^* = [n^k]$. We define J_{base} as the set of indexes i such that $J_i^* \notin S_k$. Therefore we must have $|J_{\text{base}}| \leq t$.

We first describe the simulator $\mathsf{Sim}_1^{G^{(k+1)}}$ that takes the leaking wires $\mathcal{W}^*$ and the output wires $J^* \in S_{k+1}$ to be simulated and produce the sets $I_1^* \subseteq [n^{k+1}]$ and $I_2^* \subseteq [n^{k+1}]$ of required inputs. The simulator $\mathsf{Sim}_1^{G^{(k+1)}}$ starts by defining a set $\mathcal{W}_{\text{base}}$ which is initialized to $\emptyset$; this will correspond to the set of leaking wires for the base gadget. Then the simulation goes through all the level-k gadgets composing $G^{(k+1)}$ from bottom to top *i.e.* starting with the level-k gadgets producing the output sharing up to the level-k gadgets processing the input sharings. Let us denote by $\{G_j^{(k)}\}_j$ these level-k gadgets. For each $G_j^{(k)}$, one runs the simulator Sim_1 from the $(S_k, f^{(k)})$-RPE property on input $\mathcal{W}_j$ and J_j defined as follows. The set of leaking wires $\mathcal{W}_j$ is defined as the subset of $\mathcal{W}^*$ corresponding to the wires of $G_j^{(k)}$. For the gadgets $G_j^{(k)}$ on the bottom layer, the set J_j is set to one of the J_i^* (with indices scaled to range in $[n^k]$). For all the other gadgets $G_j^{(k)}$ (which are not on the bottom layer), the set J is defined as the set I_1 or I_2 output from Sim_1 for the child gadget $G^{(k)}{}_{j'}$ (for which Sim_1 has already been run).

Whenever a failure event occurs for a $G_j^{(k)}$ gadget, namely when the set I (either I_1 or I_2) output from Sim_1 is such that $I \notin S_k$, we add the index of the wire corresponding to this input in the base gadget G to the set $\mathcal{W}_{\text{base}}$. Once the Sim_1 simulations have been run for all the $G_j^{(k)}$ gadgets, ending with the top layers, we get the final sets I corresponding to the input shares. Each of these sets corresponds to an n^k-sharing as input of a $G_j^{(k)}$ gadget, which corresponds to a wire as input of the base gadget among the $2 \cdot n$ wires carrying the two input n-sharings of the base gadget. We denote by $I_{1,1}^*, \ldots, I_{1,n}^*$ and $I_{2,1}^*, \ldots, I_{2,n}^*$ the corresponding sets so that defining

$$I_1^* = I_{1,1}^* \cup \ldots \cup I_{1,n}^* \quad \text{and} \quad I_2^* = I_{2,1}^* \cup \ldots \cup I_{2,n}^*\ , \tag{24}$$

the tuple $\widehat{x}^*|_{I_1^*}$ and $\widehat{y}^*|_{I_2^*}$ contains the shares designated by the final I sets.

At the end of the $\mathsf{Sim}_1^{G^{(k+1)}}$ simulation, the set $\mathcal{W}_{\text{base}}$ contains all the labels of wires in the base gadget G for which a failure event has occurred in the simulation of the corresponding $G_j^{(k)}$ gadget. Thanks to the $(S_k, f^{(k)})$-RPE property of these gadgets, the failure events happen (mutually independently) with probability ε_k which implies

$$\mathcal{W}_{\text{base}} \overset{\text{id}}{=} \mathsf{LeakingWires}(G, \varepsilon_k) \tag{25}$$

Recall that $|J_{\text{base}}| \leq t$. We can then run Sim_1^G to obtain:

$$(I_{1,\text{base}}, I_{2,\text{base}}) = \mathsf{Sim}_1^G(\mathcal{W}_{\text{base}}, J_{\text{base}}) \ . \tag{26}$$

For all $1 \leq i \leq n$, if $i \in I_{1,\text{base}}$, we force $I^*_{1,i} \leftarrow [n^k]$, so that the corresponding i-th input wire of the base gadget can be computed from the corresponding input wires in $I^*_{1,i}$. The simulator $\mathsf{Sim}_1^{G^{(k+1)}}$ then returns (I_1^*, I_2^*) as output.

The (t, f)-RPE property of the base gadget G implies that the *base failure events* $|I_{1,\text{base}}| = n$ and $|I_{2,\text{base}}| = n$ are ε_{k+1}-mutually unlikely, where $\varepsilon_{k+1} = f(\varepsilon_k)$. We argue that for all $1 \leq i \leq n$, $I^*_{1,i} \notin S_k \iff i \in I_{1,\text{base}}$. Namely if a failure event has occurred for a set $I^*_{1,i}$ (*i.e.* $I^*_{1,i} \notin S_k$) then we must have $i \in I_{1,\text{base}}$. Indeed, if a failure event has occurred for a set $I^*_{1,i}$ then the label of the ith input wire (for the first sharing) of the base gadget G has been added to $\mathcal{W}_{\text{base}}$ and Sim_1^G has no choice but to include this index to the set $I_{1,\text{base}}$ so that Sim_2^G can achieve a perfect simulation of the wire assignment (as required by the RPE property of G). Moreover if $i \in I_{1,\text{base}}$ then by construction we have set $I^*_{1,i} = [n^k]$ and therefore $I^*_{1,i} \notin S_k$. This implies that if $|I_{1,\text{base}}| \leq t$ then $I_1^* \in S_{k+1}$ (and the same happens for I_2^* w.r.t. $I_{2,\text{base}}$). We deduce that the failure events $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ are also ε_{k+1}-mutually unlikely, as required by the $(S_{k+1}, f^{(k+1)})$-RPE property of $G^{(k+1)}$.

We now describe the simulator $\mathsf{Sim}_2^{G^{(k+1)}}$ that takes as input $\widehat{x}^*|_{I_1^*}$ and $\widehat{y}^*|_{I_2^*}$ and produces a perfect simulation of $\left(\mathsf{AssignWires}(G^{(k+1)}, \mathcal{W}^*, (\widehat{x}^*, \widehat{y}^*)), \widehat{z}|_{J^*}\right)$ where $\widehat{z} = G^{(k+1)}(\widehat{x}, \widehat{y})$. Let $\widehat{x}^b$ and $\widehat{y}^b$ denote the n-linear sharings obtained by applying the linear decoding to each group of n^k shares in $\widehat{x}^*$ and $\widehat{y}^*$, so that the elements of $\widehat{x}^b$ and $\widehat{y}^b$ correspond to the input wires in the base gadget G. The assignment expansion property implies that a perfect assignment of the wires of $G^{(k+1)}$ on input $\widehat{x}^*$ and $\widehat{y}^*$ can be derived from an assignement of the wires of the base gadget G on input $\widehat{x}^b$ and $\widehat{y}^b$. The simulator makes use of this property by first running

$$out_{\text{base}} \leftarrow \mathsf{Sim}_2^G(\mathcal{W}_{\text{base}}, J_{\text{base}}, \widehat{x}^b|_{I_{1,\text{base}}}, \widehat{y}^b|_{I_{2,\text{base}}}) \ , \tag{27}$$

Note that the input values $\widehat{x}^b|_{I_{1,\text{base}}}$ and $\widehat{y}^b|_{I_{2,\text{base}}}$ can be obtained from the corresponding shares in I_1^* and I_2^*. Thanks to the (t, f)-RPE property of G and by construction of $I_{1,\text{base}}$ and $I_{2,\text{base}}$, this outputs a distribution satisfying

$$out_{\text{base}} \overset{\text{id}}{=} \left(\mathsf{AssignWires}(G, \mathcal{W}_{\text{base}}, (\widehat{x}^b, \widehat{y}^b)),\ \widehat{z}^b|_{J_{\text{base}}}\right) \tag{28}$$

The simulator then goes through all the $G_j^{(k)}$ gadgets from input to output and for each of them runs the simulator Sim_2 of the RPE property on inputs $\mathcal{W}_j$, J_j, $\widehat{x}|_{I_1}$ and $\widehat{y}|_{I_2}$ where $\mathcal{W}_j$ and J_j are the sets from the first phase of the simulation for the gadget $G_j^{(k)}$, I_1 and I_2 are the corresponding sets produced by the Sim_1 simulator for $G_j^{(k)}$, and $\widehat{x}$ and $\widehat{y}$ are the inputs of $G_j^{(k)}$ in the evaluation of $G^{(k+1)}(\widehat{x}^*, \widehat{y}^*)$. Provided that the partial inputs $\widehat{x}|_{I_1}$ and $\widehat{y}|_{I_2}$ are perfectly simulated, this call to Sim_2 produces a perfect simulation of $\left(\mathsf{AssignWires}(G_j^{(k)}, \mathcal{W}_j, (\widehat{x}, \widehat{y}), \widehat{z}|_{J_j}\right)$ where $\widehat{z} = G_j^{(k)}(\widehat{x}, \widehat{y})$. In order to get perfect simulations of the partial inputs $\widehat{x}|_{I_1}$ and $\widehat{y}|_{I_2}$, the simulator proceeds as follows. For the top layer of $G^{(k)}$ gadgets (the ones processing the input shares) the shares $\widehat{x}|_{I_1}$ and $\widehat{y}|_{I_2}$ can directly be taken from the inputs $\widehat{x}^*|_{I_1^*}$ and $\widehat{y}^*|_{I_2^*}$. For the next gadgets the shares $\widehat{x}|_{I_1}$ and $\widehat{y}|_{I_2}$ match the shares $\widehat{z}|_J$ output from the call to Sim_2 for a parent gadget. The only exception occurs in case of a failure event.

In that case the simulation needs the full input $\widehat{x} = (x_1, \ldots, x_{n^k})$ (and/or $\widehat{y} = (y_1, \ldots, y_{n^k})$), while we have set $|I_1| = n^k - 1$ (and/or $|I_2| = n^k - 1$) to satisfy the RPE requirements of the parent gadget in the first simulation phase. Nevertheless, for such cases a perfect simulation of the plain value $x = \mathsf{LinDec}(\widehat{x})$ (and/or $y = \mathsf{LinDec}(\widehat{y})$) is included to out_{base} by construction of $\mathcal{W}_{\text{base}}$. We can therefore perfectly simulate the missing share from the $n^k - 1$ other shares and the plain value x (or y). We thus get a perfect simulation of $\big(\mathsf{AssignWires}(G_j^{(k)}, \mathcal{W}_j, (\widehat{x}, \widehat{y}), \widehat{z}|_{J_j}\big)$ for all the level-k gadgets $G_j^{(k)}$ which gives us a perfect simulation of $\big(\mathsf{AssignWires}(G^{(k+1)}, \mathcal{W}^*, (\widehat{x}^*, \widehat{y}^*)), \widehat{z}|_{J^*}\big)$.

Saturated case: $J^* = [n^{k+1}]$. The saturated case proceeds similarly. The difference is that we must simulate all n^{k+1} output shares of the level-$(k+1)$ gadget, except for one share index j^* that can be chosen by the simulator.

The simulator $\mathsf{Sim}_1^{G^{(k+1)}}$ is defined as previously. Since $J^* = [n^{k+1}]$, we must define $J_{\text{base}} = [1, n]$. Moreover we have $J_i^* = [n^k]$ for all $1 \leq i \leq n$. This implies that for the gadgets $G_j^{(k)}$ on the output layer, the sets J_j are all equal to $[n^k]$ as well. The set $\mathcal{W}_{base}$ is defined as previously, and the simulator $\mathsf{Sim}_1^{G^{(k+1)}}$ returns (I_1^*, I_2^*) as previously. The failure events $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ are still ε_{k+1}-mutually unlikely, as required by the $(S_{k+1}, f^{(k+1)})$-RPE property of $G^{(k+1)}$.

The simulator $\mathsf{Sim}_2^{G^{(k+1)}}$ is defined as previously. In particular, from the running of the base gadget simulator Sim_2^{G}, we obtain a perfect simulation of the output wires $\widehat{z}^b|_{J'_{\text{base}}}$ for some J'_{base} with $|J'_{\text{base}}| = n - 1$. Combined with the perfect simulation of the output wires corresponding to the output sets J'_j from the gadgets $G_j^{(k)}$ on the output layer, with $|J'_j| = n^k - 1$, we obtain a subset J' of output wires for our level-$(k+1)$ gadget with $|J'| = n^{k+1} - 1$ as required. Eventually this gives us a perfect simulation of $\big(\mathsf{AssignWires}(G^{(k+1)}, \mathcal{W}^*, (\widehat{x}^*, \widehat{y}^*)), \widehat{z}|_{J'}\big)$. This terminates the proof of Theorem 2.

C.3 Proposition 3

We give here the proof of Proposition 3.

Proof. Let Sim_1^G be the simulator from the (t, f)-wRPE property. This simulator outputs I_1 and I_2 such that

$$\Pr(\mathcal{F}_1) = \varepsilon_1 \leq \varepsilon \ , \quad \Pr(\mathcal{F}_2) = \varepsilon_2 \leq \varepsilon \ \text{ and } \ \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon_{12} \leq \varepsilon^2 \ , \tag{29}$$

where $\mathcal{F}_1 \equiv (|I_1| > t)$ and $\mathcal{F}_2 \equiv (|I_2| > t)$. We show how to construct $\mathsf{Sim}_1^{G'}$ which outputs I'_1 and I'_2 such that

$$\Pr(\mathcal{F}'_1) = \Pr(\mathcal{F}'_2) = \varepsilon' \ \text{ and } \ \Pr(\mathcal{F}'_1 \wedge \mathcal{F}'_2) = (\varepsilon')^2 \ \text{ with } \ \varepsilon' = \varepsilon + \frac{3}{2}\varepsilon^2 \tag{30}$$

where $\mathcal{F}'_1 \equiv (|I'_1| > t)$ and $\mathcal{F}'_2 \equiv (|I'_2| > t)$ and such that $I_1 \subseteq I'_1$ and $I_2 \subseteq I'_2$. In particular, the latter implies that we can keep the same Sim_2^G simulator since it is always given the same input shares plus additional input shares to achieve the same simulation as before.

The simulator $\mathsf{Sim}_1^{G'}$ first calls the simulator Sim_1^G to get I_1 and I_2. Whenever $|I_1|$ and $|I_2|$ are both lower than t, *i.e.* no failure event occurs, which happens with probability $p_{\text{succ}} = 1 - (\varepsilon_1 +$

$\varepsilon_2 - \varepsilon_{12}$), $\mathsf{Sim}_1^{G'}$ outputs

$$(I_1', I_2') = \begin{cases} ([n], I_2) & \text{with probability } p_1 = \delta_1/p_{\text{succ}} \\ (I_1, [n]) & \text{with probability } p_2 = \delta_2/p_{\text{succ}} \\ ([n], [n]) & \text{with probability } p_{12} = \delta_{12}/p_{\text{succ}} \\ (I_1, I_2) & \text{with probability } 1 - (p_1 + p_2 + p_{12}) \end{cases}$$

for some $\delta_1, \delta_2, \delta_{12} \geq 0$ such that $\delta_1 + \delta_2 + \delta_{12} \leq p_{\text{succ}}$. We hence get

$$\begin{aligned} \Pr(\mathcal{F}_1') &= \varepsilon_1 + \delta_1 + \delta_{12} \\ \Pr(\mathcal{F}_2') &= \varepsilon_2 + \delta_2 + \delta_{12} \\ \Pr(\mathcal{F}_1' \wedge \mathcal{F}_2') &= \varepsilon_{12} + \delta_{12} \end{aligned}$$

We must now fix $\delta_1, \delta_2, \delta_{12} \geq 0$ to satisfy (30), with $\varepsilon' := \varepsilon + 3\varepsilon^2/2$ and $\delta_1 + \delta_2 + \delta_{12} \leq p_{\text{succ}} = 1 - (\varepsilon_1 + \varepsilon_2 - \varepsilon_{12})$. We fix $\delta_{12} = (\varepsilon')^2 - \varepsilon_{12}$; this gives $\Pr(\mathcal{F}_1' \wedge \mathcal{F}_2') = (\varepsilon')^2$, and from (29) we obtain $\delta_{12} \geq 0$ as required. We let:

$$\delta_1 := \varepsilon' - \varepsilon_1 - \delta_{12}$$

which gives $\Pr(\mathcal{F}_1') = \varepsilon'$ as required. Moreover we obtain using (29):

$$\begin{aligned} \delta_1 &= \varepsilon + \frac{3}{2}\varepsilon^2 - \varepsilon_1 - \left((\varepsilon + \frac{3}{2}\varepsilon^2)^2 - \varepsilon_{12}\right) \geq \frac{3}{2}\varepsilon^2 - \left(\varepsilon^2 + 3\varepsilon^3 + \frac{9}{4}\varepsilon^4\right) \\ &\geq \varepsilon^2 \cdot \left(\frac{1}{2} - 3\varepsilon - \frac{9}{4}\varepsilon^2\right) \geq 0 \quad \text{for } \varepsilon \leq 0.14. \end{aligned}$$

We obtain similar conditions for $\delta_2 := \varepsilon' - \varepsilon_2 - \delta_{12}$. Finally, we have

$$\begin{aligned} \delta_1 + \delta_2 + \delta_{12} &= \varepsilon' - \varepsilon_1 - \delta_{12} + \varepsilon' - \varepsilon_2 - \delta_{12} + \delta_{12} \\ &= 2\varepsilon' - \varepsilon_1 - \varepsilon_2 - (\varepsilon')^2 + \varepsilon_{12} = p_{\text{succ}} + 2\varepsilon' - (\varepsilon')^2 - 1 \\ &\leq p_{\text{succ}} + 2\varepsilon' - 1 \leq p_{\text{succ}} \quad \text{for } \varepsilon < 0.14. \end{aligned}$$

as required. □

D Instantiation

D.1 Verification functions

In this section, we give the whole set of coefficients obtained for gadgets in Section 6. When sets of coefficients are completed with ..., then a bound of the subsequent function can be obtained from the binomial coefficients as explained in Section 3. In such cases, the number of coefficients (as the number of wires in the circuit) is given in a last column.

Verification timings are also given in the tables by running the tool a laptop computer (Intel(R) Core(TM) i7-8550U CPU, 1.80GHz with 4 cores) using Ubuntu operating system.

D.2 Addition Gadgets

Hereafter are the coefficients c_i as defined in Section 3 for addition gadgets G^1_{add} and G^2_{add} defined in Section 6.

gadget	function	coefficient computed by our automatic tool ($\beta = 5$)	# wires	Verification Time (in s)
G^1_{add}	$f_1^{(1)}$	$\{0, 3, 150, 3649, 53830, \dots\}$	36	148
	$f_2^{(1)}$	$\{0, 3, 116, 2429, 34469, \dots\}$		
	$f_3^{(1)}$	$\{0, 0, 10, 495, 10959, \dots\}$		
	$f_1^{(2)}$	$\{0, 3, 144, 3342, 45611, \dots\}$		
	$f_2^{(2)}$	$\{0, 3, 110, 2208, 27580, \dots\}$		
	$f_3^{(2)}$	$\{0, 0, 4, 228, 4933, \dots\}$		
G^2_{add}	$f_1^{(1)}$	$\{0, 3, 118, 2457, 34998, \dots\}$	36	176
	$f_2^{(1)}$	$\{0, 3, 106, 2035, 27812, \dots\}$		
	$f_3^{(1)}$	$\{0, 0, 0, 69, 3034, \dots\}$		
	$f_1^{(2)}$	$\{0, 3, 118, 2403, 29859, \dots\}$		
	$f_2^{(2)}$	$\{0, 3, 106, 2007, 22079, \dots\}$		
	$f_3^{(2)}$	$\{0, 0, 0, 9, 600, \dots\}$		

D.3 Multiplication Gadgets

Hereafter are the coefficients c_i as defined in Section 3 for multiplication gadget G_{mult} defined in Section 6.

gadget	function	coefficient computed by our automatic tool ($\beta = 5$)	# wires	verification time (in s)
G_{mult}	$f_1^{(1)}$	$\{0, 3, 1232, 60940, 1653719, \dots\}$	97	5228
	$f_2^{(1)}$	$\{0, 7, 1688, 74662, 2152987, \dots\}$		
	$f_3^{(1)}$	$\{0, 0, 62, 5300, 291603, \dots\}$		
	$f_1^{(2)}$	$\{0, 3, 1254, 42135, 1428624, \dots\}$		
	$f_2^{(2)}$	$\{0, 11, 2135, 47322, 1437774, \dots\}$		
	$f_3^{(2)}$	$\{0, 0, 83, 4248, 255461, \dots\}$		

D.4 Copy Gadgets

Hereafter are the coefficients c_i as defined in Section 3 for the copy gadget G_{copy} defined in Section 6.

gadget	function	coefficient computed by our automatic tool ($\beta = s = 33$)	# wires	verification time (in s)
G_{copy}	$f_{1,1}$	$\{0, 33, 1137, 16812, 145288, 852472, 3750849, 13073855,$ $37574146, 91573962, 192726070, 354263297, 572852089,$ $818662608, 1037103082, 1166786707, 1166799413,$ $1037157725, 818809139, 573166437, 354817320, 193536720,$ $92561040, 38567100, 13884156, 4272048, 1107568,$ $237336, 40920, 5456, 528, 33, 1\}$	33	49
	$f_{1,2}$	$\{0, 30, 1285, 19887, 166695, 951201, 4021599, 13567630,$ $38231896, 92255103, 193295461, 354654683, 573074084,$ $818765733, 1037141693, 1166798076, 1166801950,$ $1037158129, 818809180, 573166439, 354817320, 193536720,$ $92561040, 38567100, 13884156, 4272048, 1107568,$ $237336, 40920, 5456, 528, 33, 1\}$		
	$f_{2,1}$	same coefficients than $f_{1,2}$		
	$f_{2,2}$	$\{0, 27, 1433, 23538, 188460, 1016149, 4150387, 13760724,$ $38465921, 92491608, 193496624, 354798258, 573159259,$ $818807160, 1037157912, 1166803059, 1166803107,$ $1037158320, 818809200, 573166440, 354817320, 193536720,$ $92561040, 38567100, 13884156, 4272048, 1107568, 237336,$ $40920, 5456, 528, 33, 1\}$		

E Complexity of the MPC Protocol in the AIS Compiler

In the following we compute the complexity and the value of N_{g} in the instantiation of the AIS compiler [2]. First, using this compiler, given a circuit C to compile, each gate G is implemented using a functionality F associated to the MPC protocol. Such a functionality F receives m shares of each input and then reconstructs them to obtain original values. This reconstruction can be done with $2(m-1)$ addition gates. Then after computing the gate G, m additive shares of the output are computed twice. This step consists of one gate for G, and $2(m-1)$ gates for the additive sharing along with $2(m-1)$ random gates.[8] So each gate G to compile is replaced by $6m-5$ gates, each computed jointly by the m parties in the MPC protocol. Next, we state the complexity of the protocol from [22]. Each gate in a functionality F is jointly computed by all m parties. In the beginning, each party holds one share of each input.

The first step consists in a k-secret sharing of each input share where $k = \binom{m}{c}$. For an input of m shares, each party will hold a total of $m\binom{m-1}{c}$ shares. For two inputs, this step has a complexity of $m(2k-2)$.

The second step is either performing an addition or a multiplication, depending on the gate G associated to the functionality. An addition simply means each party locally adding all its shares, holding a complexity of $m\binom{m-1}{c}$. In case of a multiplication gate, each party will locally compute the sum of the product of the shares of both inputs, and then share its local result using a secret sharing scheme as in the first step. This procedure holds a complexity of $\binom{m-1}{c}^2$ for computing the result, $m(2k-2)$ for the secret sharing, and $2k^2$ copy gates. Clearly, the cost of the second step is more important for the multiplication and can be upper bounded by[9]

$$\binom{m-1}{c}^2 + m \cdot (2k-2) + 2k^2.$$

In the final step, every party broadcasts its shares to all other parties, and then adds all the shares it receives. The complexity of this step is $\binom{m}{c}$.

Considering the cost of replacing each gate G in the circuit to compile by $6m-5$ gates, and the cost to compute each of these gates using the protocol Π, the total number of gates N_{g} is upper bounded by

$$(6m-5) \cdot \left(\binom{m-1}{c}^2 + m(2k-2) + 2k^2 \right).$$

[8] In [2], the authors only consider $2(m-1)$ for the cost of this step, not counting the number of random gates necessary to compute the additive sharing of the output.

[9] The authors claim in their paper a complexity of $\binom{m-1}{c}^2 + 2mk$, since they do not take into account the copy gates needed to compute the product of input shares.

Appendix G

On the Power of Expansion: More Efficient Constructions in the RP Model

Hereafter is appended the full version of our paper [BRT21], joint work with Sonia Belaïd and Abdul Rahman Taleb, published at **EUROCRYPT 2021**.

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 multi-party 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*

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

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

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

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*

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

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*

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

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 ($|I_1| > t$), ..., ($|I_\ell| > t$) 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. □

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 Appendix A.

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 Appendix B 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 $\widehat{\boldsymbol{x}} \in (\mathbb{K}^n)^\ell$*

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

yields

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

and

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

where $\boldsymbol{I} = (I_1, \ldots, I_\ell)$ 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 $I_1, \ldots, I_\ell$ which can perfectly simulate any leaking wires set $\mathcal{W}$ such that $|\mathcal{W}| \leq t$ and any set of output shares J such that $|J| \leq t$, satisfying $|I_1|, \ldots, |I_\ell| \leq |\mathcal{W}|$. Consequently, there exist input sets $I_1, \ldots, I_\ell$ which can perfectly simulate any leaking wires set $\mathcal{W}$ such that $|\mathcal{W}| = t_i \leq t$ and any set of output shares J such that $|\mathcal{W}| + |J| \leq t$ with $|I_1|, \ldots, |I_\ell| \leq t_i$. 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*

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

Proof. Since G is τ-SNI, then for any set of leaking wires $\mathcal{W}$ and output shares J such that $|\mathcal{W}| + |J| \leq \tau$, the wires indexed by $\mathcal{W}$ and the output shares indexed by J can be perfectly simulated from input shares indexed by I_1, …, I_ℓ such that $|I_j| \leq |\mathcal{W}|$ for every $1 \leq j \leq \ell$. In the TRPE1 property, the set J of output shares can be any set of size $|J| \leq t$ 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 $|\mathcal{W}| + |J| \leq \tau$ and hence the existence of the sets I_1, …, I_ℓ allowing the simulation with $|I_j| \leq |\mathcal{W}|$. And $|\mathcal{W}| < t + 1$ implies $|I_j| \leq \min(t, |\mathcal{W}|)$ 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). □

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.

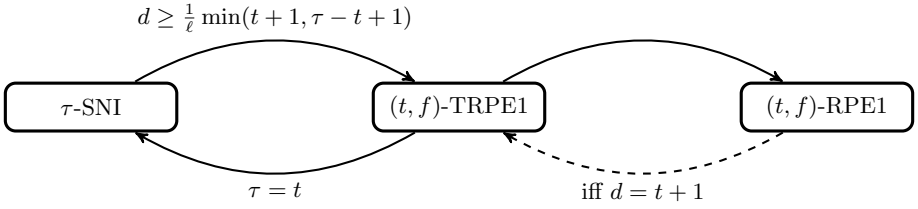

Fig. 1: Summary of relations between the different notions.

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 Appendix C).

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_{add}

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

We have the following lemma (see the proof in Appendix D).

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, $\{r_{ij}\}_{1 \leq i < j \leq n}$ 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 Appendix E.

Lemma 9. *Let $n \in \mathbb{N}$. For every $t \leq n-2$, 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 $t \leq n-2$, 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 $d \leq \min(t+1, 2(n-t))$ which gives $d \leq \frac{2n+2}{3}$ 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 Appendix F.

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 Appendix G.

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 : $(a_1, \ldots, a_n)$,$(b_1, \ldots, b_n)$ input sharings, $\{r_{ij}\}_{1 \leq i < j \leq n}$ random values
Output: $(c_1, \ldots, c_n)$ 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 $(c_1, \ldots, c_n)$;

We have the following lemma (see the proof in Appendix H).

Lemma 12. *Let $n \in \mathbb{N}$. For every $t \leq n-2$, 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 $t \leq n-2$, 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 $d \leq \min((t+1)/2, (n-t))$ which gives $d \leq (n+1)/3$ 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

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

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

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

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].

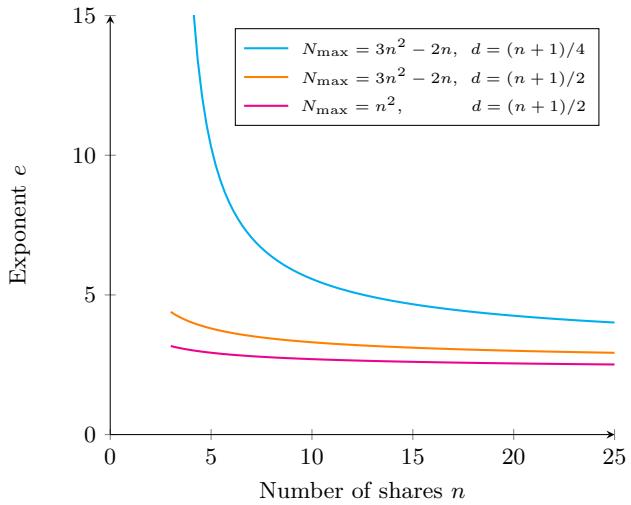

Fig. 2: Evolution of the complexity exponent $e = \log(N_{\max})/\log(d)$ with respect to the number of shares n. The blue curve matches the instantiation with the ISW-based gadgets; the orange curve assumes the optimal amplification order (*i.e.* an improvement of the multiplication gadget); the pink 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 : $(a_1, \ldots, a_n)$,$(b_1, \ldots, b_n)$ input sharings, $\{r_{ij}\}_{1 \leq i \leq n, 1 \leq j \leq n}$ random values, refresh gadget G_{refresh}
Output: $(c_1, \ldots, c_n)$ sharing of $a \cdot b$

```
for i ← 1 to n do
    (b_1^(i), ..., b_n^(i)) ← G_refresh(b_1, ..., b_n);
end
for i ← 1 to n do
    for j ← 1 to n do
        p_{i,j} ← a_i × b_j^(i) + r_{i,j};
    end
end
(v_1, ..., v_n) ← (0, ..., 0);
(x_1, ..., x_n) ← (0, ..., 0);
for i ← 1 to n do
    for j ← 1 to n do
        v_i ← v_i + p_{i,j};
        x_i ← x_i + r_{i,j};
    end
end
for i ← 1 to n do
    c_i ← v_i + x_i;
end
return (c_1, ..., 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 Appendix I.

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 the same 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**	$\mathbf{-7.50}$

Remark 1. The copy gadget G_{copy} instantiated in [9] which uses a refresh scheme with 3 randoms for each output, also reaches the amplification order 2. It tolerates a better leakage probability (*i.e.* $2^{-5.9}$) than the one provided here, but with a higher complexity of $(12, 9, 0, 6)$. If it is used to replace the 3-share copy gadget, the maximum tolerated leakage probability by the compiler from Table 1 would be of $2^{-7.4}$ slightly better than the current value of $2^{-7.5}$ but with a higher complexity of $\mathcal{O}(|C| \cdot \kappa^{4.08})$ instead of $\mathcal{O}(|C| \cdot \kappa^{3.9})$. Another copy gadget can be constructed by using the refresh scheme with 3 random values from [9] for one of the outputs, and the refresh scheme presented in this section with 2 random values for the second output. This gadget tolerates a maximum leakage probability of around $2^{-7.1}$ with a complexity of $(10, 8, 0, 5)$. Using it would bring the complexity of the compiler from Table 1 to $\mathcal{O}(|C| \cdot \kappa^4)$, while tolerating a leakage probability of $2^{-7.4}$, the same as that of the used multiplication gadget.

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 with ten random values which reaches the upper bound on the amplification order:

$$\begin{aligned} G_{\text{add}} : c_1 &\leftarrow \big((r_1 + r_2) + a_1\big) + \big((r_6 + r_7) + b_1\big) \\ c_2 &\leftarrow \big((r_2 + r_3) + a_2\big) + \big((r_7 + r_8) + b_2\big) \\ c_3 &\leftarrow \big((r_3 + r_4) + a_3\big) + \big((r_8 + r_9) + b_3\big) \\ c_4 &\leftarrow \big((r_4 + r_5) + a_4\big) + \big((r_9 + r_{10}) + b_4\big) \\ c_5 &\leftarrow \big((r_5 + r_1) + a_5\big) + \big((r_{10} + r_6) + b_5\big) \end{aligned}$$

and a copy gadget with also ten random values and which also reaches the upper bound on the amplification order:

$$\begin{aligned} G_{\text{copy}} : c_1 &\leftarrow (r_1 + r_2) + a_1; & d_1 &\leftarrow (r_6 + r_7) + a_1 \\ c_2 &\leftarrow (r_2 + r_3) + a_2; & d_2 &\leftarrow (r_7 + r_8) + a_2 \\ c_3 &\leftarrow (r_3 + r_4) + a_3; & d_3 &\leftarrow (r_8 + r_9) + a_3 \\ c_4 &\leftarrow (r_4 + r_5) + a_4; & d_4 &\leftarrow (r_9 + r_{10}) + a_4 \\ c_5 &\leftarrow (r_5 + r_1) + a_5; & d_5 &\leftarrow (r_{10} + r_6) + a_5. \end{aligned}$$

Multiplication Gadget. The following construction is a 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:

$$
\begin{aligned}
G_{\text{mult}} : i_{1,1} &\leftarrow (r_1 + r_2) + b_1; \quad i_{1,2} \leftarrow (r_2 + r_3) + b_2; \quad i_{1,3} \leftarrow (r_3 + r_4) + b_3; \\
i_{1,4} &\leftarrow (r_4 + r_5) + b_4; \quad i_{1,5} \leftarrow (r_5 + r_1) + b_5 \\
\\
i_{2,1} &\leftarrow (r_6 + r_7) + b_1; \quad i_{2,2} \leftarrow (r_7 + r_8) + b_2; \quad i_{2,3} \leftarrow (r_8 + r_9) + b_3; \\
i_{2,4} &\leftarrow (r_9 + r_{10}) + b_4; \quad i_{2,5} \leftarrow (r_{10} + r_6) + b_5 \\
\\
i_{3,1} &\leftarrow (r_{11} + r_{12}) + b_1; \quad i_{3,2} \leftarrow (r_{12} + r_{13}) + b_2; \quad i_{3,3} \leftarrow (r_{13} + r_{14}) + b_3; \\
i_{3,4} &\leftarrow (r_{14} + r_{15}) + b_4; \quad i_{3,5} \leftarrow (r_{15} + r_{11}) + b_5 \\
\\
i_{4,1} &\leftarrow (r_{16} + r_{17}) + b_1; \quad i_{4,2} \leftarrow (r_{17} + r_{18}) + b_2; \quad i_{4,3} \leftarrow (r_{18} + r_{19}) + b_3; \\
i_{4,4} &\leftarrow (r_{19} + r_{20}) + b_4; \quad i_{4,5} \leftarrow (r_{20} + r_{16}) + b_5 \\
\\
i_{5,1} &\leftarrow (r_{21} + r_{22}) + b_1; \quad i_{5,2} \leftarrow (r_{22} + r_{23}) + b_2; \quad i_{5,3} \leftarrow (r_{23} + r_{24}) + b_3; \\
i_{5,4} &\leftarrow (r_{24} + r_{25}) + b_4; \quad i_{5,5} \leftarrow (r_{25} + r_{21}) + b_5 \\
\\
a'_1 &\leftarrow (r_{26} + r_{27}) + a_1; \quad a'_2 \leftarrow (r_{27} + r_{28}) + a_2; \quad a'_3 \leftarrow (r_{28} + r_{29}) + a_3; \\
a'_4 &\leftarrow (r_{29} + r_{30}) + a_4; \quad a'_5 \leftarrow (r_{30} + r_{26}) + a_5
\end{aligned}
$$

$$
\begin{aligned}
c_1 &\leftarrow (a'_1 \cdot i_{1,1} + r_{1,1}) + (a'_1 \cdot i_{1,2} + r_{1,2}) + (a'_1 \cdot i_{1,3} + r_{1,3}) + (a'_1 \cdot i_{1,4} + r_{1,4}) \\
&\quad + (a'_1 \cdot i_{1,5} + r_{1,5}) + (r_{1,1} + r_{2,1} + r_{3,1} + r_{4,1} + r_{5,1}) \\
c_2 &\leftarrow (a'_2 \cdot i_{2,1} + r_{2,1}) + (a'_2 \cdot i_{2,2} + r_{2,2}) + (a'_2 \cdot i_{2,3} + r_{2,3}) + (a'_2 \cdot i_{2,4} + r_{2,4}) \\
&\quad + (a'_2 \cdot i_{2,5} + r_{2,5}) + (r_{1,2} + r_{2,2} + r_{3,2} + r_{4,2} + r_{5,2}) \\
c_3 &\leftarrow (a'_3 \cdot i_{3,1} + r_{3,1}) + (a'_3 \cdot i_{3,2} + r_{3,2}) + (a'_3 \cdot i_{3,3} + r_{3,3}) + (a'_3 \cdot i_{3,4} + r_{3,4}) \\
&\quad + (a'_3 \cdot i_{3,5} + r_{3,5}) + (r_{1,3} + r_{2,3} + r_{3,3} + r_{4,3} + r_{5,3}) \\
c_4 &\leftarrow (a'_4 \cdot i_{4,1} + r_{4,1}) + (a'_4 \cdot i_{4,2} + r_{4,2}) + (a'_4 \cdot i_{4,3} + r_{4,3}) + (a'_4 \cdot i_{4,4} + r_{4,4}) \\
&\quad + (a'_4 \cdot i_{4,5} + r_{4,5}) + (r_{1,4} + r_{2,4} + r_{3,4} + r_{4,4} + r_{5,4}) \\
c_5 &\leftarrow (a'_5 \cdot i_{5,1} + r_{5,1}) + (a'_5 \cdot i_{5,2} + r_{5,2}) + (a'_5 \cdot i_{5,3} + r_{5,3}) + (a'_5 \cdot i_{5,4} + r_{5,4}) \\
&\quad + (a'_5 \cdot i_{5,5} + r_{5,5}) + (r_{1,5} + r_{2,5} + r_{3,5} + r_{4,5} + r_{5,5}).
\end{aligned}
$$

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 Annual ACM Symposium on Theory of Computing*, pages 715–724, San Jose, CA, USA, June 6–8, 2011. ACM Press.
2. Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.
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, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
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: 23rd Conference on Computer and Communications Security*, pages 116–129, Vienna, Austria, October 24–28, 2016. ACM Press.

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, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.
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. `https://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, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
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, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.
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, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.
11. Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.
12. Jean-Sébastien Coron, Aurélien Greuet, and Rina Zeitoun. Side-channel masking with pseudo-random generator. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 342–375, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.
13. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany.
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, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.
15. Stefan Dziembowski, Sebastian Faust, and Karol Zebrowski. Simple refreshing in the noisy leakage model. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 315–344, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.
16. Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Heidelberg, Germany.
17. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.
18. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.
19. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.
20. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225

of *Lecture Notes in Computer Science*, pages 413–427, Santa Barbara, CA, USA, August 17–20, 2010. Springer, Heidelberg, Germany.

A Random Probing Expandability 1 & 2

Definition 7 (Random Probing Expandability 1). *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)-RPE1 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 $J \subseteq [n]$, such that $|J| \leq t$, and for every $p \in [0, 1]$, the random experiment*

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

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*

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{8}$$

   with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),
2. *the output distribution satisfies*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_J\big) \tag{9}$$

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

Definition 8 (Random Probing Expandability 2). *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)-RPE2 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 $p \in [0, 1]$, the random experiment*

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

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*

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{10}$$

   with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),
2. *J is such that $J \subseteq [n]$ with $|J| = n - 1$*
3. *the output distribution satisfies*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_J\big) \tag{11}$$

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

B Tight Random Probing Expandability

Definition 9 (Tight Random Probing Expandability). *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)-tight random probing expandable (TRPE) *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 $\mathcal{F}_1 \equiv \big(|I_1| > \min(t, |\mathcal{W}|)\big)$ and $\mathcal{F}_2 \equiv \big(|I_2| > \min(t, |\mathcal{W}|)\big)$ verify*

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{12}$$

   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*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_{J'}\big) \tag{13}$$

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

Definition 10 (Tight Random Probing Expandability 1). *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)-tight random probing expandable (TRPE) *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 $J \subseteq [n]$, such that $|J| \leq t$, and for every $p \in [0, 1]$, the random experiment*

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

ensures that

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

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{14}$$

   with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),
2. *the output distribution satisfies*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_{J}\big) \tag{15}$$

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

Definition 11 (Tight Random Probing Expandability 2). *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)-tight random probing expandable (TRPE) *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 $p \in [0, 1]$, the random experiment*

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

ensures that

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

$$\Pr(\mathcal{F}_1) = \Pr(\mathcal{F}_2) = \varepsilon \quad \textit{and} \quad \Pr(\mathcal{F}_1 \wedge \mathcal{F}_2) = \varepsilon^2 \tag{16}$$

   with $\varepsilon = f(p)$ (in particular $\mathcal{F}_1$ and $\mathcal{F}_2$ are mutually independent),
2. *J is such that $J \subseteq [n]$ with $|J| = n - 1$*
3. *the output distribution satisfies*

$$\mathit{out} \overset{id}{=} \big(\mathsf{AssignWires}(G, \mathcal{W}, (\widehat{x}, \widehat{y}))\ ,\ \widehat{z}|_J\big) \tag{17}$$

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

C Proof of Lemma 7

Proof. To prove that G_{copy} is TRPE achieving the same amplification order d as the underlying refresh gadget G_{refresh}, we need to prove that any set of leaking wires $\mathcal{W}$ such that $|\mathcal{W}| \leq d - 1$ can be perfectly simulated together with any sets of outputs wires $J_1, J_2 \subseteq [n]$ (such that J_1 refers to the first output e and J_2 to the second output f) from a set of input wires I such that $|I| \leq \min(t, |\mathcal{W}|)$. In addition, we know from Lemma 1 that the maximal amplification order achievable in the TRPE setting is $d_{\max} \leq \min(t + 1, 2(n - t))$. Since we consider sets $\mathcal{W}$ of size at most $|\mathcal{W}| \leq d-1 \leq \min(t+1, 2(n-t))-1 \leq t$ then we need to prove that $|I| \leq \min(t, |\mathcal{W}|) = |\mathcal{W}|$.

The leaking set $\mathcal{W}$ can be split into two distinct subsets $\mathcal{W}_1$ and $\mathcal{W}_2$ such that $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$ where $\mathcal{W}_1$ (resp. $\mathcal{W}_2$) is the set of leaking wires of G_{refresh} for the output e (resp. f). Let $J_1, J_2 \subseteq [n]$. We consider four cases:

– $|J_1| \leq t, |J_2| \leq t$: since $|\mathcal{W}| \leq d - 1$, then $|\mathcal{W}_1|, |\mathcal{W}_2| \leq d - 1$. Since G_{refresh} achieves an amplification order d, then by definition of TRPE, the sets $\mathcal{W}_1$ and J_1 can be simulated with a set of input shares I_1 such that $|I_1| \leq \min(|\mathcal{W}_1|, t)$. Similarly, the sets $\mathcal{W}_2$ and J_2 can be simulated with a set of input shares I_2 such that $|I_2| \leq \min(|\mathcal{W}_2|, t)$. As a consequence, set I defined as $I = I_1 \cup I_2$ is enough to simulate $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$ and both output shares J_1 and J_2. Furthermore, we have

$$|I| \leq |I_1| + |I_2| \leq \min(|\mathcal{W}_1|, t) + \min(|\mathcal{W}_2|, t) \leq |\mathcal{W}| = \min(|\mathcal{W}|, t)$$

– $|J_1| > t, |J_2| > t$: in this case, we need to prove the existence of a set of input shares I such that $|I| \leq \min(t, |W|) = |\mathcal{W}|$ (since $|\mathcal{W}| \leq d-1 \leq t$) for which we can perfectly simulate $\mathcal{W}$ together with two chosen output sets J_1' and J_2' such that $|J_1'| = |J_2'| = n-1$. Since we have $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$ such that $|\mathcal{W}_1| \leq d-1, |\mathcal{W}_2| \leq d-1$, then by definition of TRPE, there exists $J_1', |J_1'| = n-1$ such that $\mathcal{W}_1$ and J_1' can be perfectly simulated from a set of inputs shares I_1 such that $|I_1| \leq \min(|\mathcal{W}_1|, t)$. Similarly, there exists $J_2', |J_2'| = n-1$ such that $\mathcal{W}_2$ and J_2' can be perfectly simulated from a set of inputs shares I_2 such that $|I_2| \leq \min(|\mathcal{W}_2|, t)$. By choosing such sets J_1', J_2', the overall simulation of G_{copy} can be done with the set of input shares $I = I_1 \cup I_2$, and we have

$$|I| \leq |I_1| + |I_2| \leq \min(|\mathcal{W}_1|, t) + \min(|\mathcal{W}_2|, t) \leq |\mathcal{W}| = \min(|\mathcal{W}|, t)$$

– $|J_1| \leq t, |J_2| > t$: Since $|J_1| \leq t$, by definition of TRPE, $\mathcal{W}_1$ and J_1 can be perfectly simulated from a set of input shares I_1 such that $|I_1| \leq \min(|\mathcal{W}_1|, t)$. In addition, for $|J_2| > t$, we also know that we can choose a set J_2' such that $|J_2'| = n-1$ that can be perfectly simulated with $\mathcal{W}_2$ from a set of input shares I_2 with $|I_2| \leq \min(|\mathcal{W}_2|, t)$. By choosing such a set J_2', the overall simulation of G_{copy} can be achieved with the set of input wires $I = I_1 \cup I_2$, and we have

$$|I| \leq |I_1| + |I_2| \leq \min(|\mathcal{W}_1|, t) + \min(|\mathcal{W}_2|, t) \leq |\mathcal{W}| = \min(|\mathcal{W}|, t)$$

– $|J_1| > t, |J_2| \leq t$: the proof is exactly the reflect of the previous one.

Since in the four cases, there is no failure tuple $\mathcal{W}$ of size $|\mathcal{W}| < d$, then the gadget G_{copy} achieves an amplification order d. Lemma 1 finally completes the proof. □

D Proof of Lemma 8

Proof. We need to prove that when 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) of amplification order at least $\lfloor \frac{d}{2} \rfloor$. We will prove the property for the RPE setting, and the proof for the TRPE setting will be exactly the same except for the notion of failure event which changes. This amounts to proving that:

1. Any set of leaking wires $\mathcal{W}$ such that $|\mathcal{W}| < \lfloor \frac{d}{2} \rfloor$ can be simulated together with any set of outputs wires $J \subseteq [n]$ from sets of input wires I_1 on a and I_2 on b such that $|I_1| \leq t$ and $|I_2| \leq t$ (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$ and $|I_2| \leq \min(t, |\mathcal{W}|)$).
2. Any set of leaking wires such that $\lfloor \frac{d}{2} \rfloor \leq |\mathcal{W}| < d$ can be simulated together with any set of outputs wires $J \subseteq [n]$ from sets of input wires I_1, I_2 such that $|I_1| \leq t$ or $|I_2| \leq t$ (because of the double failure, *i.e* failure on both inputs) (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$ or $|I_2| \leq \min(t, |\mathcal{W}|)$).

We proceed by building the necessary simulators for G_{add} from the simulators that already exist for G_{refresh}. Concretely, we split each set $\mathcal{W}$ of leaking wires, into four subsets $\mathcal{W} = \mathcal{W}_1^r \cup W_1^a \cup \mathcal{W}_2^r \cup W_2^a$ where $\mathcal{W}_1^r$ (resp. $\mathcal{W}_2^r$) is the set of leaking wires during the computation of $G_{\text{refresh}}(a_1, \ldots, a_n)$ (resp. $G_{\text{refresh}}(b_1, \ldots, b_n)$), and $\mathcal{W}_1^a$ (resp. $\mathcal{W}_2^a$) is the set of leaking wires of $(e_1, \ldots, e_n)$ (resp. $(f_1, \ldots, f_n)$).

From these notations, we build a leaking set $\mathcal{W}'$ which contains $\mathcal{W}_1^r$ and $\mathcal{W}_2^r$ and also each input or pair of inputs of gates whose output is a wire in $\mathcal{W}_1^a$ or $\mathcal{W}_2^a$. We have that

$$|\mathcal{W}'| \leq |\mathcal{W}_1^r| + |\mathcal{W}_2^r| + 2|\mathcal{W}_1^a| + 2|\mathcal{W}_2^a| \leq 2|\mathcal{W}|.$$

The new set $\mathcal{W}'$ can be split into two subsets $\mathcal{W}'_1$ and $\mathcal{W}'_2$ such that $\mathcal{W}'_1$ (resp. $\mathcal{W}'_2$) contains only leaking wires during the computation of $G_{\text{refresh}}(a_1, \ldots, a_n)$ (resp. $G_{\text{refresh}}(b_1, \ldots, b_n)$). We now demonstrate how we can simulate $\mathcal{W}'$ when the output set J is of size less that t ((T)RPE1) and when it is of size strictly more than t ((T)RPE2).

- if $|J| \leq t$ ((T)RPE1): we prove both properties 1 and 2:
  1. we assume that $|\mathcal{W}| < \lfloor \frac{d}{2} \rfloor$. Then we consider the set $\mathcal{W}' = \mathcal{W}'_1 \cup \mathcal{W}'_2$ (as previously defined) such that
  $$|\mathcal{W}'| \leq 2|\mathcal{W}| < 2\lfloor \frac{d}{2} \rfloor \leq d$$
  and hence,
  $$|\mathcal{W}'_1| < d \quad \text{and} \quad |\mathcal{W}'_2| < d.$$
  From the (t, f)-RPE property of G_{refresh} and its amplification order, there exists an input set of shares of a I_1 such that $|I_1| \leq t$ (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$) and I_1 perfectly simulates $\mathcal{W}'_1$ and any set J_1 of up to t variables e_i. Similarly, there exists an input set of shares of b I_2 such that $|I_2| \leq t$ (for TRPE we would have $|I_2| \leq \min(t, |\mathcal{W}|)$) and I_2 perfectly simulates $\mathcal{W}'_2$ and any set J_2 of up to t variables f_i. J_1 and J_2 are chosen as the inputs e_i and f_i respectively of wires $e_i + f_i$ in set J. Namely $|J_1| = |J_2| = |J|$.
  From these definitions, I_1 and I_2 together perfectly simulate $\mathcal{W}'$ and J and are both of size less than t (less than $\min(t, |\mathcal{W}|)$ for TRPE), which proves the first property in this scenario.
  2. we now assume that $\lfloor \frac{d}{2} \rfloor \leq |\mathcal{W}| < d$. Then we consider the set $\mathcal{W}' = \mathcal{W}'_1 \cup \mathcal{W}'_2$ such that
  $$|\mathcal{W}'| \leq 2|\mathcal{W}| < 2d$$
  and hence,
  $$|\mathcal{W}'_1| < d \quad \text{or} \quad |\mathcal{W}'_2| < d.$$
  Without loss of generality, let us consider that $|\mathcal{W}'_1| < d$ (the proof is similar in the opposite scenario). From the (t, f)-RPE property of G_{refresh} and its amplification order, there exists an input set of shares of a I_1 such that $|I_1| \leq t$ (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$) and I_1 perfectly simulates $\mathcal{W}'_1$ and any set J_1 of up to t variables e_i. There also exists an input set of shares of b I_2 which perfectly simulates $\mathcal{W}'_2$ and any set J_2 of up to t variables f_i but not necessarily of size less than t (less than $\min(t, |\mathcal{W}|)$ for TRPE). If J_1 and J_2 are chosen as the inputs e_i and f_i respectively of wires $e_i + f_i$ in set J, then sets I_1 and I_2 together perfectly simulate $\mathcal{W}'$ and J. In this case, we only have a failure on at most one of the inputs (b in this case), which concludes the proof for the second property.

  At this point, we proved that G_{add} achieves an amplification order greater than or equal to $\lfloor \frac{d}{2} \rfloor$ for RPE1 (for TRPE1 in the TRPE setting).
- if $|J| > t$ ((T)RPE2): we prove both properties 1 and 2:
  1. we assume that $|\mathcal{W}| < \lfloor \frac{d}{2} \rfloor$. Then we consider the set $\mathcal{W}' = \mathcal{W}'_1 \cup \mathcal{W}'_2$ (as previously defined) such that
  $$|\mathcal{W}'| \leq 2|\mathcal{W}| < d.$$
  $\mathcal{W}'_1$ and $\mathcal{W}'_2$ both point to leaking wires in instances of G_{refresh}. We denote by $\mathcal{W}''_1$ the set of leaking wires on the first instance of G_{refresh} (on input a) such that $\mathcal{W}''_1$ contains $\mathcal{W}'_1$ and all the wires that are leaking within the second instance of G_{refresh} (designated by $\mathcal{W}'_2$ in this

second instance). Hence, we have that $|\mathcal{W}_1''| \leq |\mathcal{W}_1' \cup W_2'| < d$. From the (t, f)-RPE ((t, f)-TRPE in the TRPE setting) property of G_{refresh} and its amplification order, there exists an input set of shares of a I_1 such that $|I_1| \leq t$ (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$) and a set of output shares e_i J_1' of size $n-1$ such that the input shares of I_1 perfectly simulate the wires designated by $\mathcal{W}_1''$ and J_1'. Similarly, as both instances of G_{refresh} are identical, the same set I_2 but of input shares b perfectly simulates $\mathcal{W}_2''$ (defined as the equivalent of $\mathcal{W}_1''$ on the second instance) and J_2' which points to the same output shares than J_1 but on f_i instead of e_i. We thus have two input sets I_1 and I_2 of size less than t (less than $\min(t, |\mathcal{W}|)$ for TRPE2) whose shares perfectly simulate the wires $\mathcal{W}'$ and the elements $e_i + f_i$ of a set J' of size $n-1$ with $i \in J_1' = J_2'$. That concludes the proof for the first property.

2. we now assume that $\lfloor \frac{d}{2} \rfloor \leq |\mathcal{W}| < d$. Then we consider the set $\mathcal{W}' = \mathcal{W}_1' \cup \mathcal{W}_2'$ such that

$$|\mathcal{W}'| \leq 2|\mathcal{W}| < 2d.$$

Without loss of generality, let us consider that $|\mathcal{W}_1'| < d$ (the proof is similar in the opposite scenario). From the (t, f)-RPE ((t, f)-TRPE in the TRPE setting) property of G_{refresh} and its amplification order, there exists a set J_1' such that $|J_1'| = n-1$ and a set of input shares I_1 such that I_1 perfectly simulates $\mathcal{W}_1'$ and J_1' and $|I_1| \leq t$ (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$). Thus, we can select a set J' of outputs of G_{add} such that J' corresponds to the outputs of J_1 (for each element e_i designated by J_1, $e_i + f_i$ is pointed by J). Then, by choosing $I_2 = [n]$, we have two input sets I_1 and I_2 which perfectly simulate $\mathcal{W}'$ and an output set J' of size $n-1$ such that $|I_1| \leq t$ (for TRPE we would have $|I_1| \leq \min(t, |\mathcal{W}|)$). That concludes the proof for the second property.

We thus proved that G_{add} achieves an amplification order greater than or equal to $\lfloor \frac{d}{2} \rfloor$ for RPE2 (for TRPE2 in the TRPE setting).

Since G_{add} has an amplification order greater than or equal to $\lfloor \frac{d}{2} \rfloor$ for RPE1 and RPE2 (resp. TRPE1 and TRPE2), then G_{add} is a (t, f')-RPE (resp. (t, f')-TRPE) addition gadget for some function f' of amplification order $d' \geq \lfloor \frac{d}{2} \rfloor$, which concludes the proof. □

E Proof of Lemma 9

Proof. We start by proving the first property of the lemma, i.e the amplification order d_1 for TRPE1. The n-share ISW refresh gadget was proven to be $(n-1)$-SNI [5], hence it follows from Lemma 6 that $d_1 \geq \min(t+1, n-t)$. In addition, we know from the proof of Lemma 1 and as explained in section 4.1 that $d_1 \leq t+1$. It remains to show that $d_1 \leq n-t$. We thus have to exhibit a simulation failure by carefully choosing $n-t$ leaking variables (the leaking set $\mathcal{W}$) together with t leaking output variables (indexed by the set J). Consider the set of output shares indexed by $J = \{1, \ldots, t\}$, which corresponds to the first t shares $c_1, \ldots, c_t$ of the output. Next, we construct the set of leaking wires $\mathcal{W}$ of size $n-t$. First, observe that the partial sums of the output shares are of the form

$$c_{i,j} = \begin{cases} a_i + r_{1,i} + \cdots + r_{j,i} & \text{if } j < i \\ a_i + r_{1,i} + \cdots + r_{i-1,i} + r_{i,i+1} + \cdots + r_{i,j} & \text{otherwise.} \end{cases}$$

Then, let $\mathcal{W} = \{c_{t+1,t}, \ldots, c_{n,t}\}$. We can prove that the constructed set $\mathcal{W}$ along with the set of indexes of output shares $J = \{1, \ldots, t\}$ cannot be perfectly simulated with at most $\min(t, |\mathcal{W}|)$

input shares. For this, we consider a variable $s = c_1 + \cdots + c_t + c_{t+1,t} + \ldots + c_{n,t}$, the sum of the t output shares indexed in J, and the leaking variables from $\mathcal{W}$. Each of the output shares $\{c_i\}_{1\leq i\leq t}$ is the sum of exactly one input share a_i and $n-1$ random values. Each of the leaking variables $\{c_{i,t}\}_{t+1\leq i\leq n}$ is the sum of exactly one input share a_i and t random values. In addition, it can be observed that each random value appears exactly twice in the set of expressions of the variables $\{c_i\}_{1\leq i\leq t} \cup \{c_{i,t}\}_{t+1\leq i\leq n}$, so all of the random values are eliminated in the expression of the variable s, which is the sum of all of these variables. Since each of the variables has one input share a_i appearing in its expression, then we have $s = a_1 + \ldots + a_n = a$. Thus, simulating the variable s requires the knowledge of the full input, and hence the leaking variables indexed by $\mathcal{W}$ and J cannot be perfectly simulated without the knowledge of the full input. Hence, the set $\mathcal{W}$ of size $n-t$ represents a failure set with respect to TRPE1, and so the function f_1 cannot be of amplification order higher than $n-t$, that is $d_1 \leq n-t$. From the three inequalities $d_1 \geq \min(t+1, n-t)$, $d_1 \leq t+1$ and $d_1 \leq n-t$, we obtain $d_1 = \min(t+1, n-t)$.

Next, we demonstrate the second part of the lemma, *i.e.* the amplification order d_2 for TRPE2. Let $\mathcal{W}$ be a set of leaking wires such that $|\mathcal{W}| < t+1$. We aim to prove that there exists a set J indexing $n-1$ output shares such that the leaking variables indexed by both $\mathcal{W}$ and J can be perfectly simulated with the input shares indexed by a set I such that $|I| \leq \min(t, |\mathcal{W}|) = |\mathcal{W}|$. First, we observe that the leaking wires in $\mathcal{W}$ are of the following forms:

1. input share a_i
2. random variable r_{ij} $\quad (i < j)$
3. partial sum $c_{ij} = \begin{cases} a_i + r_{1,i} + \cdots + r_{j,i} & \text{if } j < i \\ a_i + r_{1,i} + \cdots + r_{i-1,i} + r_{i,i+1} + \cdots + r_{i,j} & \text{otherwise.} \end{cases}$

We then build I from an empty set as follows. For every wire in $\mathcal{W}$ of the first or third form, we add index i to I. For every wire in $\mathcal{W}$ of the second form (r_{ij}), if $i \in I$, we add j to I, otherwise we add i to I. By construction we have $|I| \leq |\mathcal{W}| \leq t$. Moreover, the wires in the set $\mathcal{W}$ only depends of the input shares a_i with $i \in I$ which implies that we can perfectly simulate the variables indexed by $\mathcal{W}$ from the input shares indexed by I. We then build the set J as the union of two subsets J_1 and J_2 such that $J_1 = I$ and J_2 is any set satisfying $|J_2| = n-1-|I|$ and $J_1 \cap J_2 = \emptyset$. Now, we aim to show that the output shares determined by the indexes in $J = J_1 \cup J_2$ can be further perfectly simulated from the input shares indexed by I (namely given the previous simulation of the variables from $\mathcal{W}$). The simulation works as follows:

- each output share c_i such that $i \in J_1$ can be perfectly simulated with a_i (since $i \in I$) and $n-1$ uniformly random variables (the same generated r_{ij} can be reused for several c_i);
- for each output share c_i such that $i \in J_2$, we have $i \notin I$ and hence a_i is not available. Since by construction of J_1, all the variables observed through the set $\mathcal{W}$ are included in the set of variables observed through J_1, and since $|J_1| \leq |\mathcal{W}| \leq t \leq n-2$ and $|J_2| = n-1-|J_1|$, then each output wire c_i indexed in J_2 has at least one random value that does not appear in any other observation from $\mathcal{W}$ or J_1, so c_i can be assigned to a fresh random value. This produces a perfect simulation of all output wires indexed in J_2.

We thus obtain a perfect simulation of the output shares indexed by $J = J_1 \cup J_2$, such that $|J| = n-1$, together with the variables indexed by $\mathcal{W}$, from the input shares indexed by a set I of size $|I| \leq |\mathcal{W}| \leq t$, so $|I| \leq \min(t, |\mathcal{W}|)$. Hence the ISW refresh gadget is (t, f)-TRPE2 with an amplification order $d_2 \geq t+1$. In addition, we know from the proof of Lemma 1 and as explained in Section 4.1 that $d_2 \leq t+1$, hence $d_2 = t+1$ which concludes the proof. □

F Proof of Lemma 10

Proof. In order to prove that the amplification order d of G_{copy} instantiated with the ISW refresh gadget is equal to $\min(t+1, n-t)$, we first demonstrate that $d \geq \min(t+1, n-t)$ and then we show the existence of failure tuples to argue that $d \leq \min(t+1, n-t)$.

In fact, we already know that the ISW refresh gadget is (t, f_1)-TRPE of amplification order $d_1 = \min(t+1, n-t)$. Then from Lemma 7, we know that G_{copy} instantiated with ISW refresh is also (t, f_2)-TRPE of amplification order $d_2 = d_1 = \min(t+1, n-t)$. Then, from Lemma 4 we have that G_{copy} is (t, f)-RPE of amplification order $d \geq d_2 = \min(t+1, n-t)$. Next, we need to prove that $d \leq \min(t+1, n-t)$. In addition, we know from Lemma 1 that $d \leq t+1$. Hence, it remains to show that it is also upper bounded by $n-t$.

We know from the proof of Lemma 9 that, for the ISW refresh gadget, we can construct a set of leaking wires $\mathcal{W}$ of size $n-t$ along with a set of t indexes of output shares J such that a perfect simulation of both sets W and J requires the knowledge of the full input sharing *i.e.* $I = [n]$. Then, in the case of the copy gadget G_{copy}, let $\mathcal{W}$ be the set of leaking wires and $J_1, J_2 \subseteq [n]$ be the sets of output shares on the outputs e and f respectively. Then, we can split $\mathcal{W}$ into two distinct subsets $\mathcal{W}_1$ and $\mathcal{W}_2$ such that $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$, where $\mathcal{W}_1$ (resp. $\mathcal{W}_2$) is the set of leaking wires of ISW G_{refresh} for the output e (resp. f). Then, in the case where $|J_1| \leq t$, we can construct the set $\mathcal{W} = \mathcal{W}_1$ of size $n-t$ ($\mathcal{W}_2 = \emptyset$) in the exact same way as in the proof of Lemma 9, such that we have simulation failure of $\mathcal{W}_1$ along with the output shares indexed in J_1 on the input of the gadget. Otherwise, in the case where $|J_2| \leq t$, we can construct the set $\mathcal{W} = \mathcal{W}_2$ of size $n-t$ ($\mathcal{W}_1 = \emptyset$) in the exact same way, such that we have simulation failure of $\mathcal{W}_2$ along with the output shares indexed in J_2 on the input of the gadget. Hence, the amplification order d of G_{copy} is upper bounded by $n-t$.

From the three inequalities $d \geq \min(t+1, n-t)$, $d \leq t+1$ and $d \leq n-t$, we conclude that the copy gadget instantiated with ISW refresh is (t, f)-RPE of amplification order $d = \min(t+1, n-t)$. □

G Proof of Lemma 11

Proof. We proceed similarly to the proof of Lemma 8 to show that the function f_1 (resp. f_2) is of amplification order at least $\min(t+1, n-t)$ (resp. $(t+1)$). We split each set $\mathcal{W}$ of leaking wires, into four subsets $\mathcal{W} = \mathcal{W}_1^r \cup W_1^a \cup \mathcal{W}_2^r \cup W_2^a$ where $\mathcal{W}_1^r$ (resp. $\mathcal{W}_2^r$) is the set of leaking wires during the computation of $G_{\text{refresh}}(a_1, \ldots, a_n)$ (resp. $G_{\text{refresh}}(b_1, \ldots, b_n)$), and $\mathcal{W}_1^a$ (resp. $\mathcal{W}_2^a$) is the set of leaking wires of $(e_1, \ldots, e_n)$ (resp. $(f_1, \ldots, f_n)$). Then, we prove using previous lemmas that the amplification order for TRPE1 (resp. for TRPE2) at most $\min(t+1, n-t)$ (resp. $(t+1)$) by exhibiting failure tuples. Hence the final equalities.

- if $|J| \leq t$ (TRPE1): we prove two properties like in Lemma 8:
  1. we assume that $|\mathcal{W}| < \min(t+1, n-t)$, in particular $|\mathcal{W}_1^r| + |\mathcal{W}_1^a| < \min(t+1, n-t) \leq n-t$ and $|\mathcal{W}_2^r| + |\mathcal{W}_2^a| < \min(t+1, n-t) \leq n-t$. Since we have $|J| \leq t$, then $|\mathcal{W}_1^r| + |\mathcal{W}_1^a| + |J| \leq n-1$ and $|\mathcal{W}_2^r| + |\mathcal{W}_2^a| + |J| \leq n-1$. Then from the $(n-1)$-SNI property of the ISW refresh gadget, and by choosing J_1 and J_2 as the inputs e_i and f_i respectively of the wires $e_i + f_i$ in set J ($|J_1| = |J_2| = |J|$), we know that there exists an input set of shares of a I_1 such that $|I_1| \leq |\mathcal{W}_1^r| \leq |\mathcal{W}| \leq t$ and I_1 perfectly simulates $\mathcal{W}_1^r$, $\mathcal{W}_1^a$ and J_1. And there exists an

input set of shares of b I_2 such that $|I_2| \leq |\mathcal{W}_2^r| \leq |W| \leq t$ and I_2 perfectly simulates $\mathcal{W}_2^r$, $\mathcal{W}_2^a$ and J_2. Thus I_1 and I_2 together perfectly simulate $\mathcal{W}$ and J and are both of size less than $|\mathcal{W}| \leq t$ so less than $\min(t, |\mathcal{W}|)$. Hence, there is no failure on the inputs for any set of leaking wires $\mathcal{W}$ of size strictly less than $\min(t+1, n-t)$ along with a set J of at most t output shares.

2. we now assume that $\min(t+1, n-t) \leq |\mathcal{W}| < 2 \cdot \min(t+1, n-t)$. Without loss of generality, let us consider that $|\mathcal{W}_1^r| + |\mathcal{W}_1^a| < \min(t+1, n-t)$. We consider the set of input shares of b $I_2 = [n]$, which trivially simulates all of the wires in $\mathcal{W}_2^r$, $\mathcal{W}_2^a$ and the inputs f_i of the wires $e_i + f_i$ in set J. Next, since $|\mathcal{W}_1^r| + |\mathcal{W}_1^a| < \min(t+1, n-t) \leq n-t$, by choosing J_1 as the inputs e_i of the wires $e_i + f_i$ in set J, we know that $|\mathcal{W}_1^r| + |\mathcal{W}_1^a| + |J_1| \leq n-1$ and by the $(n-1)$-SNI property of ISW refresh gadget, there exists a set of input shares of a I_1 such that $|I_1| \leq |\mathcal{W}_1^r| \leq t$ so $|I_1| \leq \min(t, |\mathcal{W}_1^r|) \leq \min(t, |\mathcal{W}|)$ and I_1 perfectly simulates $\mathcal{W}_1^r$, $\mathcal{W}_1^a$ and J_1. Then I_1 and I_2 together perfectly simulate $\mathcal{W}$ and J, and we only have a failure on input b with $|I_2| = n > t$. Thus, for any set of leaking wires $\mathcal{W}$ such that $\min(t+1, n-t) \leq |\mathcal{W}| < 2 \cdot \min(t+1, n-t)$, we have a failure on at most one of the inputs.

From the above properties, we have that G_{add} is of amplification order $d_1 \geq \min(t+1, n-t)$ for TRPE1. Then, from the proof of Lemma 1, we know that there exists an immediate failure tuple of size $t+1$ (on the input shares), hence $d_1 \leq t+1$. From the same proof of Lemma 1, we also know that there exists a failure tuple of size $2(n-t)$ (with a set of t output shares). Since G_{add} has two inputs, then it results in the following lower bound: $d_1 \geq n-t$. From these three inequalities on d_1, we obtain $d_1 = \min(t+1, n-t)$.

– if $|J| > t$ (TRPE2): we also prove both properties 1 and 2:

1. we assume that $|\mathcal{W}| < t+1$ with $\mathcal{W} = \mathcal{W}_1^r \cup W_1^a \cup \mathcal{W}_2^r \cup W_2^a$. We need to prove that there exists a set J of $n-1$ output wires such that $\mathcal{W}$ and J can be perfectly simulated with sets of input shares I_1 and I_2 such that $|I_1| \leq \min(t, |\mathcal{W}|) = |\mathcal{W}|$ and $|I_2| \leq \min(t, |\mathcal{W}|) = |\mathcal{W}|$. Recall that all of the wires in $\mathcal{W}_1^r$ (resp. $\mathcal{W}_2^r$) are of the following forms:
   (a) input share a_i (resp. b_i).
   (b) random variable r_{ij} (resp. r'_{ij}) with $i < j$.
   (c) partial sum $e_{ij} = \begin{cases} a_i + r_{1,i} + \cdots + r_{j,i} & \text{if } j < i \\ a_i + r_{1,i} + \cdots + r_{i-1,i} + r_{i,i+1} + \cdots + r_{i,j} & \text{otherwise.} \end{cases}$
   resp. $f_{ij} = \begin{cases} b_i + r'_{1,i} + \cdots + r'_{j,i} & \text{if } j < i \\ b_i + r'_{1,i} + \cdots + r'_{i-1,i} + r'_{i,i+1} + \cdots + r'_{i,j} & \text{otherwise.} \end{cases}$

   In addition, the wires in $\mathcal{W}_1^a$ (resp. $\mathcal{W}_2^a$) are output wires of G_{refresh} of the form e_i (resp. f_i). We build I_1 and I_2 from empty sets as follows. For every wire in $\mathcal{W}_1^a \cup \mathcal{W}_2^a$, we add index i to I_1 and I_2. Next, for every wire in $\mathcal{W}_1^r \cup \mathcal{W}_2^r$ of the first or third form, we add index i both to I_1 and I_2. For every wire in $\mathcal{W}_1^r \cup \mathcal{W}_2^r$ of the second form, if $i \in I_1 (= I_2)$, we add j to I_1 and I_2. Otherwise, we add i to I_1 and I_2. It is clear that $I_1 = I_2$, and that $|I_1| = |I_2| \leq |\mathcal{W}| \leq t$. Following the t-SNI proof of the ISW refresh gadget, we can show that $\mathcal{W}_1^a$ and $\mathcal{W}_1^r$ are perfectly simulated using shares of indexes in I_1. Respectively, $\mathcal{W}_2^a$ and $\mathcal{W}_2^r$ are perfectly simulated using shares of indexes in I_2. So all wires in $\mathcal{W}$ are perfectly simulated using shares of indexes in I_1 and I_2. We then build the set J of $n-1$ indexes of output shares from two subsets J_1 and J_2. We define $J_1 = I_1 (= I_2)$, and J_2 as any set such that $J_2 = n - 1 - |J_1|$ and $J_1 \cap J_2 = \emptyset$. We now show that the output shares determined by the indexes in $J = J_1 \cup J_2$ can be perfectly simulated from I_1 and I_2:

- each output share $e_i + f_i$ such that $i \in J_1 \subseteq J$ can be perfectly simulated from e_i and f_i. Precisely, e_i can be perfectly simulated with a_i (since $i \in I_1$) and $n-1$ uniformly random variables. And each f_i can be perfectly simulated with b_i (since $i \in I_2$) and $n-1$ uniformly random variables.
- for each output share $e_i + f_i$ such that $i \in J_2$ so $i \notin I_1$, $i \notin I_2$, we show that we can still perfectly simulate e_i and f_i. By construction of the set J_1, all the variables observed through the set $\mathcal{W}$ are included in the set of variables observed through e_j and f_j for $j \in J_1$, and since $|J_1| = |I_1| \leq |\mathcal{W}| \leq t \leq n-2$ and $|J_2| = n-1-|J_1|$, then each of the wires e_i and f_i for which $e_i + f_i$ is indexed in J_2 has at least one random value that does not appear in any other observation from $\mathcal{W}$ or wires e_i and f_i for which $e_i + f_i$ is indexed in J_1. So e_i can be assigned to a fresh random value, and f_i can be assigned to a fresh random value. Thus $e_i + f_i$ is also assigned to a random value. This produces a perfect simulation of all output wires indexed in J_2.

Having a perfect simulation of J_1 and J_2, we conclude that we can perfectly simulate J along with $\mathcal{W}$ from the sets I_1 and I_2 with $|I_1| = |I_2| \leq |\mathcal{W}|$. So for every set of leaking wires $\mathcal{W}$ of size at most t, there exists a set of $n-1$ output wires J which can be perfectly simulated along with $\mathcal{W}$ from sets of input shares I_1 and I_2 of sizes at most $|\mathcal{W}| = \min(|\mathcal{W}|, t)$.

2. we now assume that $t+1 \leq |\mathcal{W}| < 2(t+1)$. Without loss of generality, let us consider that $|\mathcal{W}_1^r \cup \mathcal{W}_1^a| < t+1$ (the proof is similar in the opposite scenario). We consider the set of input shares of b $I_2 = [n]$ which trivially simulates all of the wires in $\mathcal{W}_2^r \cup \mathcal{W}_2^a$ and all of the inputs f_i of the output wires $e_i + f_i$. We construct the set I_1 for input shares of a similarly to the earlier construction (when $|\mathcal{W}| < t+1$), while only considering the sets $\mathcal{W}_1^r$ and $\mathcal{W}_1^a$. The corresponding set I_1 will produce a perfect simulation of $\mathcal{W}_1^r \cup \mathcal{W}_1^a$. So I_1 and I_2 perfectly simulate the set $\mathcal{W}$. Now we choose the set J from two subsets J_1 and J_2 such that $J_1 = I_1$ and J_2 as any set such that $J_2 = n-1-|J_1|$ and $J_1 \cap J_2 = \emptyset$. We now show that the output shares determined by the indexes in $J = J_1 \cup J_2$ can be perfectly simulated from I_1 and I_2:
   - each output share $e_i + f_i$ such that $i \in J_1 \subseteq J$ can be perfectly simulated from e_i and f_i. Precisely, e_i can be perfectly simulated with a_i (since $i \in I_1$) and $n-1$ uniformly random variables. And each f_i can be perfectly simulated with b_i (since $I_2 = [n]$) and $n-1$ uniformly random variables.
   - for each output share $e_i + f_i$ such that $i \in J_2$ so $i \notin I_1$, the input wire f_i can be perfectly simulated with b_i (since $I_2 = [n]$) and $n-1$ uniformly random variables. In addition, by construction of the set J_1, all the variables observed through the set $\mathcal{W}_1^r \cup \mathcal{W}_1^a$ are included in the set of variables observed through e_j for $j \in J_1$, and since $|J_1| = |I_1| \leq |\mathcal{W}| \leq t \leq n-2$ and $|J_2| = n-1-|J_1|$, then each of the wires e_i for which $e_i + f_i$ is indexed in J_2 has at least one random value that does not appear in any other observation from $\mathcal{W}_1^r \cup \mathcal{W}_1^a$ or wires e_i for which $e_i + f_i$ is indexed in J_1. So e_i can be assigned to a fresh random value. Thus $e_i + f_i$ is perfectly simulated from e_i and f_i. This produces a perfect simulation of all output wires indexed in J_2.

   Having a perfect simulation of J_1 and J_2, we conclude that we can perfectly simulate J along with $\mathcal{W}$ from the sets I_1 and I_2 with $|I_1| \leq t$ so $|I_1| \leq \min(t, |\mathcal{W}|) = t$ (recall that $|\mathcal{W}| \geq t+1$) and $|I_2| = n$, which is only a failure on one of the inputs b.

From the proof of both properties 1 and 2 for TRPE2, we thus have that G_{add} instantiated with ISW refresh achieves the amplification order $d_2 \geq t+1$ for TRPE2. In addition, we know from the proof of Lemma 1 and as explained in section 4.1 that $d_2 \leq t+1$. Hence $d_2 = t+1$.

We finally proved both amplification orders d_1 and d_2 for TRPE1 and TRPE2 respectively for G_{add} displayed in Algorithm 2 and instantiated with the n-share ISW refresh gadget, which concludes the proof. □

H Proof of Lemma 12

Proof. We start by proving the first property of the lemma. Since the n-share ISW multiplication gadget is $(n-1)$-SNI [5], then we know from Lemma 6 that

$$d_1 \geq \frac{\min(t+1, n-t)}{2}.$$

In addition, we know from the proof of Lemma 2 that

$$d_1 \leq \frac{t+1}{2}.$$

It remains to show that $d_1 \leq (n-t)/2$. In this purpose, we exhibit a simulation failure on both inputs by carefully choosing $n-t$ leaking variables, with t output variables. Consider the set of indexes of output shares $J = \{1, \ldots, t\}$, which corresponds to the first t output shares $c_1, \ldots, c_t$. Next, we construct the set of leaking wires $\mathcal{W}$ of size $n-t$. First, observe that the partial sums of the output shares are of the form

$$c_{i,j} = \begin{cases} a_i \cdot b_i + r_{i,1} + \cdots + r_{i,j} & \text{if } j < i \\ a_i \cdot b_i + r_{i,1} + \cdots + r_{i,i-1} + r_{i,i+1} + \cdots + r_{i,j} & \text{otherwise.} \end{cases}$$

Then, let $\mathcal{W} = \{c_{t+1,t}, \ldots, c_{n,t}\}$. We can prove that the constructed set $\mathcal{W}$ along with the set of output shares $J = \{1, \ldots, t\}$ cannot be perfectly simulated with at most t input shares. For this, we consider a variable $s = c_1 + \cdots + c_t + c_{t+1,t} + \ldots + c_{n,t}$, the sum of the t output shares indexed in J, and the leaking variables from $\mathcal{W}$. Each of the output shares $\{c_i\}_{1 \leq i \leq t}$ is the sum of

- one product of input shares $a_i \cdot b_i$
- $n-1$ random values,
- at most $n-1$ pairs of input shares products: $(a_i \cdot b_j,\ a_j \cdot b_i)$ with $i \neq j$.

Each of the leaking variables $\{c_{i,t}\}_{t+1 \leq i \leq n}$ is the sum of

- one product of input shares $a_i \cdot b_i$,
- t random values,
- at most t pairs of input shares products: $(a_i \cdot b_j,\ a_j \cdot b_i)$ with $i \neq j$.

In addition, each random value appears exactly twice in the set of expressions of the variables $\{c_i\}_{1 \leq i \leq t} \cup \{c_{i,t}\}_{t+1 \leq i \leq n}$, so all the random values are eliminated from the expression of the variable s, which is the sum of all of these variables. Hence, $s = a_1 \cdot b_1 + \ldots + a_n \cdot b_n + C$ where C is a variable containing other products of input shares of the form $a_i \cdot b_j$ and $a_j \cdot b_i$ with $i \neq j$. Thus, simulating the variable s requires the knowledge of the full inputs a and b. Since s is constructed from the set of leaking wires $\mathcal{W}$ and the output shares indexed in J, then $\mathcal{W}$ and J cannot be perfectly simulated without the knowledge of the full inputs a and b. Hence, the set $\mathcal{W}$ of size $n-t$ represents a failure tuple on both inputs, and so the function f_1 for RPE1 cannot be of amplification order higher than $(n-t)/2$. Thus, $d_1 \leq (n-t)/2$.

From the three inequalities $d_1 \geq \dfrac{\min(t+1, n-t)}{2}$, $d_1 \leq (t+1)/2$ and $d_1 \leq (n-t)/2$, we conclude that

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

Next, we demonstrate the second part of the lemma. Let $\mathcal{W}$ be a set of leaking wires such that $|\mathcal{W}| \leq t$. We aim to prove that there exists a set J of $n-1$ output wires such that $\mathcal{W}$ and J can be perfectly simulated with sets of input shares I_1 on a and I_2 on b such that $|I_1| \leq t$, $|I_2| \leq t$. First, observe that the leaking wires in $\mathcal{W}$ are of the following forms :

1. input shares a_i, b_i, product of shares $a_i \cdot b_i$.
2. partial sum $c_{i,j} = \begin{cases} a_i \cdot b_i + r_{i,1} + \cdots + r_{i,j} & \text{if } j < i \\ a_i \cdot b_i + r_{i,1} + \cdots + r_{i,i-1} + r_{i,i+1} + \cdots + r_{i,j} & \text{otherwise.} \end{cases}$
3. random variable r_{ij} for $i < j$, variable $r_{ji} = a_i \cdot b_j + r_{ij} + a_j \cdot b_i$ for $j > i$.
4. product of shares $a_i \cdot b_j$, or variable $a_i \cdot b_j + r_{ij}$ with $i \neq j$.

We build sets I_1 and I_2 from empty sets as follows. For every wire in $\mathcal{W}$ of the first or second form, we add index i to I_1 and I_2. For every wire in $\mathcal{W}$ of the third or fourth form, if $i \in I_1$, we add j to I_1, otherwise we add i to I_1, and if $i \in I_2$, we add j to I_2, otherwise we add i to I_2. Since $\mathcal{W}$ is of size at most t, then $|I_1| \leq t$ and $|I_2| \leq t$. Following the t-SNI property proof from [5], we can show that $\mathcal{W}$ is perfectly simulated using shares of indexes in I_1 and I_2. We now build the set J of $n-1$ indexes of output shares from two subsets J_1 and J_2. We define $J_1 = \{i \mid c_{i,j} \text{ is observed in } \mathcal{W}\}$. Next, we define J_2 as any set such that $|J_2| = n - 1 - |J_1|$ and $J_1 \cap J_2 = \emptyset$. Now, we show that the output shares determined by the indexes in $J = J_1 \cup J_2$ can be perfectly simulated from I_1 and I_2:

- First consider the output wires indexed in J_1, which have a partial sum observed. For each such variable c_i, the biggest partial sum which is observed is already simulated. For the remaining r_{ij} in c_i, if $i < j$, then r_{ij} is assigned to a fresh random value. Otherwise, if r_{ji} enters in the computation of any other internal observation, then $i, j \in I_1$ and $i, j \in I_2$, and so r_{ji} can be perfectly simulated from the input shares. If not, then r_{ji} is replaced by the random value r_{ij}. So all output wires indexed in J_1 are perfectly simulated from I_1 and I_2.
- Now consider the output wires indexed in J_2. None of the c_i indexed in J_2 has a partial sum observed. Meanwhile, each c_i indexed in J_2 is composed of $n-1$ random values, and at most one of them can enter in the expression of each other output wire c_j. Since by construction of J_1, all the variables observed through the set $\mathcal{W}$ are included in the set of variables observed through J_1, and since $|J_1| \leq |\mathcal{W}| \leq t \leq n-2$ and $|J_2| = n - 1 - |J_1|$, then each output wire c_i indexed in J_2 has at least one random value that does not appear in any other observation from $\mathcal{W}$ or J_1, so c_i can be assigned to a fresh random value. This produces a perfect simulation of all output wires indexed in J_2.

We conclude that the set J of $n-1$ wires is perfectly simulated along with $\mathcal{W}$ from the constructed sets I_1 and I_2 of sizes $|I_1| \leq |\mathcal{W}| \leq t$ and $|I_2| \leq |\mathcal{W}| \leq t$. So there is no failure set of observations of size at most t for RPE2 on any of the inputs. Hence $d_2 \geq (t+1)/2$. In addition, we know from the proof of Lemma 2 and as explained in section 4.1 that $d_2 \leq (t+1)/2$. Hence, $d_2 = (t+1)/2$, which concludes the proof for RPE2. □

I Proof of Lemma 13

Recall the procedure of the gadget. We consider that we have an n-share (t, f')-TRPE refresh gadget G_{refresh} achieving the amplification order $d \geq \min(t+1, n-t)$. First, the gadget G_{mult} performs n executions of the gadget G_{refresh} on the input sharing $(b_1, \ldots, b_n)$ to produce:

$$\begin{aligned}(b_1^{(1)}, \ldots, b_n^{(1)}) &\leftarrow G_{\text{refresh}}(b_1, \ldots, b_n)\\ &\cdots\\ (b_1^{(n)}, \ldots, b_n^{(n)}) &\leftarrow G_{\text{refresh}}(b_1, \ldots, b_n)\end{aligned}$$

then, the gadget constructs the matrix of the cross product of input shares using the refreshed input shares of b:

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

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 randomized 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 $(c_1, \ldots, c_n)^T = V + X$ such that

$$\begin{aligned} c_1 &= V_1 + X_1 \\ &\cdots \\ c_n &= V_n + X_n. \end{aligned}$$

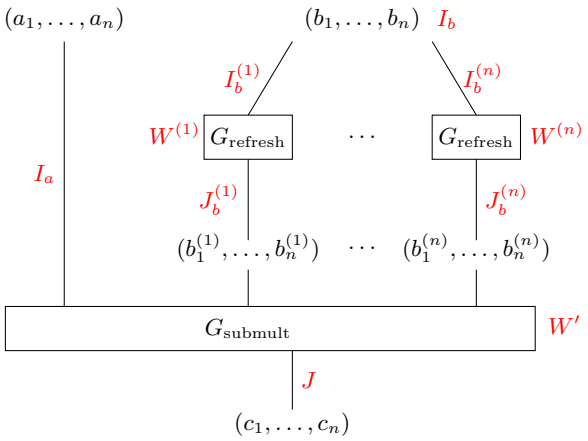

Fig. 3: G_{mult} gadget from Section 5.5.

Figure 3 represents the G_{mult} gadget from a high-level, composed of several blocks. First, a refresh gadget G_{refresh} is executed n independent times on the input sharing of b to produce n fresh copies $b^{(1)}, \ldots, b^{(n)}$. Then, the gadget G_{submult} takes as input $(a_1, \ldots, a_n)$ and the outputs of the refreshing gadgets $b^{(1)}, \ldots, b^{(n)}$ to produce the output of G_{mult}.

In the following proofs, we will denote W to be any set of probes on the global gadget G_{mult}, then W can be split as $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$ where $W^{(i)}$ is the set of probes on the internal wires of the execution of G_{refresh} for the fresh sharing $b^{(i)}$ of b, and W' is the set of probes on the internal wires of G_{submult}. We will also denote J to be any set of output wires of G_{mult} (which are the output wires of G_{submult}), and $J_b^{(i)}$ (resp. $I_b^{(i)}$) any set of output wires (resp. input wires) of the execution of G_{refresh} for the fresh sharing $b^{(i)}$ of b. Observe that any probe on the output wires of G_{refresh} for any sharing $b^{(i)}$ can be obtained through internal probes in W' on G_{submult}, so in the beginning we always consider that $J_b^{(i)} = \emptyset$ for all $i \in [n]$.

Observe that any probe in the set W' on the internal wires of G_{submult} is of one of the following forms:

(a) $a_i,\ b_j^{(i)},\ a_i \cdot b_j^{(i)},\ r_{i,j},\ p_{i,j} = a_i \cdot b_j^{(i)} + r_{i,j}$,
(b) $V_{i,j}$ partial sum of the first j terms of V_i. Observe that $V_{i,n} = V_i$,
(c) $X_{i,j}$ partial sum of the first j terms of X_i. Observe that $X_{i,n} = X_i$.

Also observe that each random value $r_{i,j}$ only appears in the expression of the wires $r_{i,j}$, $p_{i,j}$, $V_{i,j}$, or $X_{j,i}$ (so also $c_i = V_i + X_i$ and $c_j = V_j + X_j$), and does not appear anywhere else in the wires.

We will first start by proving some simple claim.

Claim 1 *Let J be a set of output shares of G_{mult} and $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$ be a set of leaking wires as described above such that $|J| + |W'| \leq n - 1$ (we only consider the set W' of probes on the internal wires to $G_{submult}$). Then, for any $i \in J$ such that $V_{i,j} \notin W$ for any $j \in [n]$, the output wire c_i can be perfectly simulated by generating a uniform random value without knowing any of the input shares.*

Proof. Let $i \in J$ such that $V_{i,j} \notin W'$ for any $j \in [n]$. Then we know that the expression of V_i in $c_i = V_i + X_i$ contains $n - 1$ random values since $V_i = p_{i,1} + \ldots + p_{i,n}$ and each $p_{i,j} = a_i \cdot b_j^{(i)} + r_{i,j}$

(without counting the random $r_{i,i}$ because it is cancelled out in c_i as it appears in V_i and X_i and $c_i = V_i + X_i$). Observe that each random value $r_{i,k}$ in V_i appears in exactly one other output share $c_k = V_k + X_k$ that comes from the expression of $X_k = r_{1,k} + \ldots + r_{i,k} + \ldots + r_{n,k}$. In other terms, each output share c_k has exactly one random value in common with V_i in c_i. Then, by probing $|J|$ output shares in J including c_i, there are at least $n - |J|$ remaining random values in V_i that do not appear in any other expression of the output shares. In addition, observe that any probed variable in W' can have in its expression at most one random value in common with V_i (because each random value $r_{i,j}$ appears exactly once in each of the wires $p_{i,j}$, $r_{i,j}$ or X_j). Then, since $|W'| \leq n - |J| - 1$ (because $|J| + |W'| \leq n - 1$), there is at least $n - |J| - (n - |J| - 1) = 1$ remaining random value $r_{i,\ell}$ where $\ell \in [n]$ in V_i, that does not appear in any other expression of the probed values in W' or J. So $c_i = V_i + X_i$ can be perfectly simulated by generating the uniform random value $r_{i,\ell}$, which concludes the proof.

In the following, we will separately prove the TRPE1 then the TRPE2 property on G_{mult} via Lemmas 14 and 17 to demonstrate Lemma 13.

I.1 Proof for TRPE1 property

Lemma 14. *The multiplication gadget G_{mult} is (t, f_1)-TRPE1 of amplification order $d = \min(t + 1, n - t)$*

Proof. We proceed in two steps through the following two lemmas 15 and 16, considering the leaking wires in two distinct ranges.

Lemma 15. *Let J be a set of at most t output shares of G_{mult}. Let W be a set of leaking wires as described above such that $|W| \leq d - 1 \leq t$. Then W and J can be perfectly simulated from at most $\min(t, |W|) = |W|$ shares of each of the inputs a and b.*

Proof. Let J be the set of t output shares of G_{mult} (i.e of G_{submult}), and let $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$ with $|W| \leq d - 1 \leq t$ be the set of probes on the global gadget G_{mult} and decomposed as explained earlier. We organize the proof in two steps:

1. We first identify the set of input shares I_a and the sets $J_b^{(i)}$ for $i \in [n]$ which are necessary to perfectly simulate J and W' in G_{submult}.
2. Then, we show that we can perfectly simulate the sets $J_b^{(i)}$ and $W^{(i)}$ for $i \in [n]$ using the simulator of the gadget G_{refresh}. This will determine the sets $I_b^{(i)}$ necessary for each of the n simulations of G_{refresh}, and thus determine the set I_b of input shares on b as $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)}$.

Using I_b, we will be able to perfectly simulate $J_b^{(i)}$ and $W^{(i)}$ for $i \in [n]$. Then using I_a and $J_b^{(i)}$ for $i \in [n]$, we will be able to perfectly simulate W' and J. This will lead to a perfect simulation of all probes W and output shares in J on the global gadget G_{mult}.

We first start by constructing the set of input shares indices I_a and the sets $J_b^{(k)}$ for $k \in [n]$ depending on the probes in the set W' as follows[3]:

(a) For probes of form (a), we add index i to I_a, and index j to $J_b^{(k)}$ for $k \in [n]$.

[3] We consider that all $J_b^{(k)}$ are empty at first since all the output shares of G_{refresh} can be probed directly in W'.

(b) For probes of form (b), we add index i to I_a and to $J_b^{(k)}$ for $k \in [n]$.
(c) For probes of form (c), we add index i to $J_b^{(k)}$ for $k \in [n]$.

Observe that since $|W| \leq d-1$, then in particular $|W'| \leq d-1 \leq t$, then $|I_a| \leq |W'| \leq |W| \leq \min(t, |W|)$ so we have no failure on the input a. We also have $|J_b^{(k)}| \leq |W'| \leq t$.

Simulation of W': probes of the form (a) can be perfectly simulated from the corresponding input shares in I_a and $J_b^{(k)}$, and by generating uniformly random values $r_{i,j}$ when necessary. Probes of the form (c) are also perfectly simulated by simply generating uniformly random values, since $X_{i,j} = r_{1,i} + \ldots + r_{j,i}$. As for probes of the form (b), we know that $i \in I_a$ and $i \in J_b^{(i)}$, then we look at each of the terms $p_{i,j'}$ for $j' \in [j]$ in $V_{i,j} = p_{i,1} + \ldots p_{i,j}$. In particular, if $j \geq i$, the term $p_{i,i}$ is in the partial sum $V_{i,j}$ and is perfectly simulated using the input shares a_i and $b_i^{(i)}$ and by generating the random value $r_{i,i}$. Next, for each $p_{i,j'}$ such that $j' \neq i$, if $j' \in J_b^{(i)}$, then $p_{i,j'}$ can be perfectly simulated from the corresponding input shares and by generating uniformly at random $r_{i,j'}$. Otherwise, if $j' \notin J_b^{(i)}$, then that means that the wires $p_{i,j'}, r_{i,j'}$ and $X_{j'}$ are not probed in W' because otherwise j' would have been added to all $J_b^{(k)}$ for $k \in [n]$. Since the random value $r_{i,j'}$ only appears in the expression of the wires $p_{i,j'}, r_{i,j'}$ and $X_{j'}$ (besides $V_{i,j}$ which is already probed), and of the output wire $c_{j'} = V_{j'} + X_{j'}$, we need to consider two cases:

- $j' \notin J$: in this case, the random value $r_{i,j'}$ can be used to mask the expression of $p_{i,j'}$ in the partial sum $V_{i,j}$, perfectly simulating it without the need to the share $b_{j'}^{(i)}$.
- $j' \in J$: $c_{j'} = V_{j'} + X_{j'}$, and $r_{i,j'}$ is the one of the summed terms in the expression of $X_{j'}$. We know that $V_{j',k} \notin W'$ for any $k \in [n]$ since otherwise j' would have been added to $J_b^{(i)}$. Since in addition we have $|J| + |W| \leq t + d - 1 \leq t + n - t - 1 \leq n - 1$, by claim 1, the output share c'_j can be masked by some random value $r_{j',\ell}$. Thus, $X_{j'}$ is masked and $r_{i,j'}$ does not appear anymore in $c_{j'}$. So $r_{i,j'}$ can be used to mask the expression of $p_{i,j'}$ in the partial sum $V_{i,j}$. This brings us to a perfect simulation of $p_{i,j'}$ simply by generating at random $r_{i,j'}$.

By perfectly simulating each of the terms $p_{i,j'}$ for $j' \in [j]$ in the probed wire $V_{i,j}$ independently, we can perfectly simulate their sum and thus perfectly simulate $V_{i,j}$. This brings us to a perfect simulation of the set W'.

Simulation of J: Let $i \in J$.

- if $V_{i,j} \notin W'$ for any $j \in [n]$, then by claim 1, c_i is perfectly simulated by simply generating a uniform random value $r_{i,\ell}$ for some $\ell \in [n]$.
- if $V_{i,j} \in W'$ for at least one $j \in [n]$, then let $V_{i,j'}$ be the largest of the probed partial sums. All of the partial sums including $V_{i,j'}$ are perfectly simulated as described earlier. Then, let us consider $c_i + V_{i,j'} = p_{i,j'+1} + \ldots + p_{i,n} + X_i$. The wire X_i can be perfectly simulated by generating uniform random values. As for each of the terms $p_{i,j'+1}, \ldots, p_{i,n}$, they can each be perfectly simulated in the exact same way each of the terms in $V_{i,j'}$ are simulated independently.

  In the particular case where $j' \leq i$ then the term $p_{i,i} = a_i \cdot b_i^{(i)} + r_{i,i}$ appears in the expression of $c_i + V_{i,j'}$, and in this case, the random value $r_{i,i}$ is cancelled out in the expression of $c_i + V_{i,j'}$ since it appears in both $p_{i,i}$ and X_i, and $c_i + V_{i,j'} = p_{i,j'+1} + \ldots + p_{i,i} + \ldots + p_{i,n} + X_i$. So to simulate the term $p_{i,i}$ in $c_i + V_{i,j'}$ we need both input shares a_i and $b_i^{(i)}$. This is already the case by construction because we assume that $V_{i,j'} \in W'$.
  Thus, by perfectly simulating $V_{i,j'}$ and $c_i + V_{i,j'}$, the output share c_i is also perfectly simulated.

Also, since $|J_b^{(k)}| \leq |W'| \leq t$ and $|W^{(k)}| \leq d-1$, and since G_{refresh} is (t, f')-TRPE achieving the amplification order d, then we can perfectly simulate sets $J_b^{(k)}$ and $W^{(k)}$ from the set of input shares $I_b^{(k)}$ such that $|I_b^{(k)}| \leq |W^{(k)}| \leq t$ for $k \in [n]$. Thus, we can let $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)}$ and we have $|I_b| \leq |W^{(1)}| + \ldots + |W^{(n)}| \leq |W| \leq \min(|W|, t)$, so we have no failure on the input b either. Until now, we have shown that we can simulate all sets $W^{(k)}$ and $J_b^{(k)}$ from I_b of size at most $\min(|W|, t)$. It remains to show that we can also perfectly simulate the sets W' and J from I_a and $J_b^{(k)}$ for $k \in [n]$.

We have shown that we can perfectly simulate any set of t output shares J and any set of probes W of size at most $d-1$, with at most $\min(|W|, t)$ shares of each of the inputs a and b. This concludes the proof of Lemma 15. □

Remark 2. We can observe that for this lemma to apply on G_{mult}, we do not need the pre-processing phase of the refresh on input b. In fact, we can see that during the construction of the sets $J_b^{(k)}$, we add each index to all of the sets for all $k \in [n]$. However, executing n refreshings on the input b is necessary to prove the next result, specifically when we consider W such that $d \leq |W| \leq 2d-1$.

To get back to the proof of Lemma 14, we also need the following result.

Lemma 16. *Let J be a set of at most t output shares of G_{mult}. Let W be a set of leaking wires as described above such that $d \leq |W| \leq 2d-1$. Then W and J can be perfectly simulated from the sets of input shares I_a and I_b such that $|I_a| \leq \min(|W|, t)$ **or** $|I_b| \leq \min(|W|, t)$. In other terms, we have a simulation failure on at most one of the inputs a or b.*

Proof. Recall that the set W can be split into subsets $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$ as described above. We can consider two cases.

Case 1: $|\mathbf{W'}| \leq \mathbf{d-1}$. This case is similar to the case of Lemma 15, so we can construct the set I_a in the same way as in the proof of Lemma 15, and we can eventually consider $I_b = [n]$. We know that $|I_a| \leq |W'| \leq d-1 \leq t$, so there is no failure on the input a. And all probes in W' can be simulated like in the proof of Lemma 15 with I_a and trivially with $I_b = [n]$. Also, all probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ can be trivially simulated since we have access to the full input b. As for output shares in J, whenever $i \in J \cap I_a$, then $c_i = V_i + X_i$ is easily simulated using $I_b = [n]$. If $i \in J$ but $i \notin I_a$, then $V_{i,j} \notin W'$ for any $j \in [n]$ and since $|J| + |W'| \leq t + d - 1 \leq t + n - t - 1 \leq n - 1$, c_i is perfectly simulated by a single random value thanks to claim 1. Thus, W and J are perfectly simulated with at most $|W'| \leq \min(|W|, t)$ shares of a and eventually n shares of b.

Case 2: $|\mathbf{W'}| \geq \mathbf{d}$ *(and thus* $|W^{(1)} \cup \ldots \cup W^{(n)}| \leq d-1$*).* In this case, we will construct the sets I_a and $J_b^{(k)}$ from empty sets, in a way that we will have a simulation failure on at most one of the inputs a or b, and we will be able to perfectly simulate W' and output shares in J using I_a and $J_b^{(k)}$. We will also show how to perfectly simulate all $J_b^{(k)}$ and $W^{(k)}$ using a set of input shares I_b.

First, we construct the sets I_a and $J_b^{(k)}$ depending on the probes in W' as follows:

(a) For probes of form (a), we add index i to I_a, and index j only to $J_b^{(i)}$.
(b) For probes of form (b), we add index i to I_a and only to $J_b^{(i)}$.
(c) For probes of form (c), we add index i to $J_b^{(k)}$ for all $k \in [n]$.

In the rest of the proof, we will show that if we have a failure on one of the inputs, we can still perfectly simulate W and J without a failure on the other input. In this purpose, we will consider two cases: in the first case (2.1), we will have a failure on input a (*i.e.*, more than $\min(t, |W|)$ shares of a are added to I_a) and in the second case (2.2), we won't have a failure on input a, and so we will eventually have a failure on input b.

Case 2.1: Simulation failure on input a. Notice that by construction we always have $|I_a| \leq |W'| \leq |W|$. Thus, a simulation failure on input a for TRPE1 means that the set I_a is of size $|I_a| \geq t+1 \geq d$. We will first start by showing that the sets $W^{(k)}$ and $J_b^{(k)}$ can be perfectly simulated using the simulator of G_{refresh} without a failure on the input b. Next, we will show that W' and output shares in J can be perfectly simulated using I_a and $J_b^{(k)}$.

Since we only add shares indices to I_a when we have probes of the form (a) or (b), this means that we have at least $t+1$ probes of these two forms with $t+1$ different values for the index i. In addition, since we have at least $t+1$ probes (a) or (b) with distinct values for the index i, then this also means that each of the sets $J_b^{(i)}$ built from these probes has at most one share of $b^{(i)}$ added to it by construction. In other terms, when we only consider probes of the form (a) and (b) with distinct i, we have $|J_b^{(k)}| \leq 1$ for each $k \in [n]$.

Now let us consider the remaining probes in W which are either in W' of the form (c), in W' of the form $(a)/(b)$ for which $i \in I_a$ or in $W^{(1)} \cup \ldots \cup W^{(n)}$. Since $|I_a| \geq t+1 \geq d$, then there are at most $d-1$ of these remaining probes. Without loss of generality, we consider that there are exactly $d-1$ instead of at most $d-1$ probes. Let m be the number of probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ and $d-1-m$ the remaining in W' of the form (c) or of the form $(a)/(b)$ for which $i \in I_a$.

Since each wire in W' of the form (c) or of the form $(a)/(b)$ for which $i \in I_a$ results in adding at most one more share index to each $J_b^{(k)}$ for $k \in [n]$, then we have $|J_b^{(k)}| \leq 1 + (d-1-m) = d-m$. And $|W^{(1)} \cup \ldots \cup W^{(n)}| \leq m$, in particular $|W^{(k)}| \leq m$ for any $k \in [n]$.

- if $m = 0$, then $W^{(k)} = \emptyset$ for any $k \in [n]$, and $|J_b^{(k)}| \leq d \leq \min(t+1, n-t) \leq \lfloor \frac{n+1}{2} \rfloor \leq n-1$, so by the TRPE property of G_{refresh} for any $t \leq n-1$, all of the $J_b^{(k)}$ sets can be perfectly simulated with no knowledge of the input shares of b since $W^{(k)} = \emptyset$, so $I_b^{(k)} = \emptyset$. Hence, $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)} = \emptyset$ and we have no simulation failure on the input b.
- if $m > 0$, then $|J_b^{(k)}| \leq d-1 \leq t$ and $|W^{(1)} \cup \ldots \cup W^{(n)}| \leq d-1$, in particular $|W^{(k)}| \leq d-1$ for each $k \in [n]$. Thus, by the (t, f)-TRPE property of the refresh gadget G_{refresh} achieving the amplification order d for any $t \leq n-1$, we can perfectly simulate both sets for each $k \in [n]$ with $I_b^{(k)}$ such that $|I_b^{(k)}| \leq |W^{(k)}|$. Thus, we can let $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)}$ so we can have $|I_b| \leq |W^{(1)} \cup \ldots \cup W^{(n)}| \leq d-1 \leq t$, and we can perfectly simulate $W^{(1)} \cup \ldots \cup W^{(n)}$ along with $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$ from the set I_b without a simulation failure on input b.

So far we proved that if we have $|I_a| \geq t+1$, then we must have $|I_b| \leq |W^{(1)} \cup \ldots \cup W^{(n)}| \leq d-1 \leq t$, and $W^{(1)} \cup \ldots \cup W^{(n)}$ can be perfectly simulated along with $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$ from the set I_b. Next we need to prove that we can perfectly simulate W' and J from these sets I_a and $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$.

Case 2.1.1: $I_a = [n]$. This only occurs by construction in the case where $|W| = 2d-1 = n$ so when $d = d_{\max} = \lfloor \frac{n+1}{2} \rfloor$ for $t = \lceil \frac{n-1}{2} \rceil$. In this case, since $|W| \leq 2d-1 \leq (n+1)-1 \leq n$, then all probes in W are all in W' of the form (a) or (b) with n distinct values for the index i and so

$|J_b^{(i)}| \leq 1$ for all $i \in [n]$. In other words, for each $i \in [n]$ there is exactly one probe in W' of the form (a) or (b) and no probe of the form (c) i.e $X_{i,j}$ nor probes in $W^{(1)} \cup \ldots \cup W^{(n)}$. We will prove that all the probes in W and in J can be perfectly simulated from these constructed sets I_a and $J_b^{(i)}$ for $i \in [n]$. For this, for each $i \in [n]$ we consider three cases:

- $V_{i,j} \notin W'$ for any $j \in [n]$, then we know that there exists a probe of the form (a) in W' with index i, in other terms, $a_i \in W'$, or $\exists!\ j \in [n]$ such that $b_j^{(i)}$ or $a_i \cdot b_j^{(i)}$ or $r_{i,j}$ or $p_{i,j} = a_i \cdot b_j^{(i)} + r_{i,j}$ is probed in W'. The corresponding probe is perfectly simulated by construction of the sets I_a and $J_b^{(i)}$.
  If we also have $i \in J$, then we know that we only have one probe of the form (a) for the considered index i in W' and no probe of the form (b) or any probe of the form (c). And since there are t output shares probed in J, then there are at least $n - t - 1 > 1$ (since $t = \lceil \frac{n-1}{2} \rceil$) remaining random values which only appear in the expression of c_i, and any of them can be used to perfectly simulate c_i without the knowledge of the input shares (*i.e.*, to mask c_i).
- $V_{i,n} \in W'$ then $V_{i,n}$ contains in its expression n random values $r_{i,1}, \ldots, r_{i,n}$. Since there are no probes of the form (a) for the index i, and no probes of the form (c), then each of these random values appears at most once in each of the expressions of the probed outputs c_j in J. With t probed output shares, there are $n - t > 1$ (since $t = \lceil \frac{n-1}{2} \rceil$) remaining random values which only appear in the expression of $V_{i,n}$ and any of them can be used to perfectly simulate $V_{i,n}$, *i.e.*, mask $V_{i,n}$.
  If in addition we have $i \in J$, then the output share c_i is perfectly simulated by simulating $V_{i,n}$ and simulating $c_i + V_{i,n} = X_i$ which is perfectly simulated by generating uniform random values.
- $V_{i,j} \in W'$ for some $j \in [n]$ such that $1 < j < n$ ($j > 1$ because otherwise it would be the wire $p_{i,1}$ which is probed). Thus, $V_{i,j}$ is the sum of at least two wires p_{i,j_1} and p_{i,j_2}.
  - If $i \notin J$, then c_i is not probed and $V_{i,j}$ is the sum of at most $n-1$ terms of the form $p_{i,1} = a_i \cdot b_1^{(i)} + r_{i,1}, \ldots p_{i,j} = a_i \cdot b_j^{(i)} + r_{i,j}$. We have that $i \in I_a$ by construction and $j \in J_b^{(i)}$. In fact we can reconstruct $J_b^{(i)}$ into $J_b^{(i)} = \{1, \ldots, j\}$ such that $|J_b^{(i)}| \leq n-1$ and since $W^{(i)} = \emptyset$, then by the (t, f)-TRPE1 property of G_{refresh} for any $t \leq n-1$, we still have no failure on the input b and we still have $|I_b^{(i)}| \leq |W^{(i)}| = 0$. In addition, we can perfectly simulate this way all of the summed terms in $V_{i,j}$ by using the corresponding input shares and thus we can perfectly simulate $V_{i,j}$. Since we have no probes of the form (a) for this same index i, then reconstructing $J_b^{(i)}$ does not affect the simulation of the probes.
  - If $i \in J$, then we consider $V_{i,j}$ and $c_i + V_{i,j}$. Since we have no probes of the form (a) for the index i, then as proven before, with t probed output shares, there are at least $n - t > 1$ remaining random values which only appear in the expression of $V_{i,j}$ or $c_i + V_{i,j}$. Any of these random values can be used to mask the expression of $V_{i,j}$ or $c_i + V_{i,j}$. In the case where the expression of $V_{i,j}$ is masked, then we can reconstruct as before the set $J_b^{(i)}$ with at most $n-1$ output shares of $b^{(i)}$ in order to perfectly simulate all the terms $p_{i,k}$ in $c_i + V_{i,j}$ including the shares of $b^{(i)}$ and thus perfectly simulate $c_i + V_{i,j}$ (the rest of the terms are just random values to be generated uniformly at random). In the other case where the expression of $c_i + V_{i,j}$ is masked, we can also reconstruct the set $J_b^{(i)}$ with at most $n-1$ output shares of $b^{(i)}$ in order to perfectly simulate all the summed terms in $V_{i,j}$. In either case, by perfectly simulating one term ($V_{i,j}$ or $c_i + V_{i,j}$) masked by a random value, and perfectly simulating

the remaining one with $i \in I_a$ and the reconstructed set $J_b^{(i)}$, we can perfectly simulate both $V_{i,j}$ and $c_i + V_{i,j}$ and hence also perfectly simulate the output share c_i.

So we proved that we can perfectly simulate the sets W' and J from the constructed set I_a and from sets $J_b^{(i)}$ such that $|J_b^{(i)}| \leq n-1$ for all $i \in [n]$. Furthermore, from the TRPE property of G_{refresh} for any $t \leq n-1$ and the fact that $W^{(i)} = \emptyset$ for all $i \in [n]$, we have no simulation failure on the input b. This concludes the simulation of W and output shares in J for the case where $I_a = [n]$.

Case 2.1.2: $I_a \subset [n]$ *with* $|I_a| \leq n-1$. In this case, we have at least one index $k \in [n] \setminus I_a$ for which there are no probes in W' of the form (a) or (b). In other terms, no partial sum of V_k is probed, no product of shares $a_k \cdot b_j^{(k)}$ or $p_{k,j}$ is probed, and no random value $r_{k,j}$ is probed since otherwise we would have $k \in I_a$ by construction.

On another hand, since $|I_a| \geq t+1$, there are at most $d-1 \leq n-t-1$ remaining probes of the form (c) in W', and since we have t output shares in the set J, there exists at least one wire X_ℓ such that $\ell \notin J$ and for which there is no partial sum $X_{\ell,j}$ probed in W'.

These two wires X_ℓ and V_k for $\ell, k \in [n]$ will be very important for the simulation of the sets W' and J. In particular, we need the two following claims.

Claim 2 *Let $i \in J$. Suppose that $i \notin I_a$. Then the expression of $c_i = V_i + X_i$ can be masked by the random value $r_{i,\ell}$, in other terms $c_i \leftarrow r_{i,\ell}$.*

Proof. This claim can be proved easily, since we suppose that $i \notin I_a$ so the random value $r_{i,\ell}$ and $p_{i,\ell}$ are not probed in W'. In addition, since $\ell \notin J$ and $X_{\ell,j} \notin W'$ for all $j \in [n]$, then the random value $r_{i,\ell}$ does not appear in any other probed wire expression except in c_i, then c_i can be masked by the random value $r_{i,\ell}$. □

Claim 3 *Let $i \in J$. Suppose that $X_{i,j} \notin W'$ for any $j \in [n]$. Suppose that $i \in I_a$. Then the expression of $c_i = V_i + X_i$ can be masked by the random value $r_{k,i}$, in other terms $c_i \leftarrow r_{k,i}$.*

Proof. Since we suppose that $k \notin I_a$, then the random value $r_{k,i}$ or $p_{k,i}$ or $V_{k,j}$ for all $j \in [n]$ are not probed in W'. Then, if $k \notin J$, then the random value $r_{k,i}$ does not appear in the expression of any other probed wire in W' or J and c_i can be masked by the random value $r_{k,i}$. Otherwise, if $k \in J$, then by Claim 2, $c_k = V_k + X_k$ can be masked by $r_{k,\ell}$ and so c_i can also be masked by $r_{k,i}$ since $i \neq \ell$ (because $i \in J$ and $\ell \notin J$). □

From these two claims, we are now ready to show that W' and J can be perfectly simulated with the sets I_a and $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$ as constructed earlier with respect to the probes in the set W'. Recall that all probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ and $J_b^{(1)} \cup \ldots \cup J_b(n)$ are perfectly simulated using I_b and the simulator of G_{refresh}.

Simulation of W'. Probes of the form (a) and (c) are trivially simulated by construction of the sets of input shares and by generating uniformly at random the necessary random values. Let us now check the probes of the form (b). Let $V_{i,j} = p_{i,1} + \ldots + p_{i,j}$ be such a probe. Let us consider each of the terms $p_{i,j'}$ for $j' \in [j]$. if $j' = i$, then by construction $p_{i,i}$ is perfectly simulated using a_i and $b_i^{(i)}$ and by generating the random value $r_{i,i}$ if needed. Otherwise, let $j' \neq i$. If $j' \in J_b^{(i)}$ then the simulation of $p_{i,j'}$ is straightforward. Otherwise if $j' \notin J_b^{(i)}$, then we know that none of the wires

$r_{i,j'}$ or $p_{i,j'}$ or $X_{j',s}$ for all $s \in [n]$ are probed in W'. Thus, $r_{i,j'}$ can be eventually used to mask the expression of $p_{i,j'}$ without the need of the share $b^{(i)}_{j'}$ for the simulation. Meanwhile, we still need to check if $j' \in J$, since $X_{j'}$ appears in the expression of $c_{j'} = V_{j'} + X_{j'}$. Then we consider two cases:

- If $j' \notin I_a$, then by claim 2, $c_{j'}$ can be masked by the random value $r_{j',\ell}$ and so $r_{i,j'}$ does not appear in the expression of $X_{j'}$ in $c_{j'}$ anymore, and $r_{i,j'}$ can be used to mask $p_{i,j'}$.
- Otherwise, if $j' \in I_a$, then by claim 3, $c_{j'}$ can be masked by the random value $r_{k,j'}$ and so $r_{i,j'}$ does not appear in the expression of $X_{j'}$ in $c_{j'}$ anymore, and $r_{i,j'}$ can be used to mask $p_{i,j'}$ (since $i \notin k$).

Thus, each term $p_{i,j'}$ in $V_{i,j}$ can be perfectly simulated and thus $V_{i,j} = p_{i,1} + \ldots + p_{i,j}$ can be perfectly simulated. This concludes the simulation of the set W'.

Simulation of J. Let $i \in J$. If $i \notin I_a$, then by claim 2, c_i is perfectly simulated by generating the random value $r_{i,\ell}$. Otherwise, let $i \in I_a$. If $X_{i,j} \notin W$ for any $j \in [n]$, then by claim 3, c_i is perfectly simulated by generating the random value $r_{k,i}$. Otherwise, we can show that we can perfectly simulate each term in $c_i = V_i + X_i$. In particular, each term in X_i can be simulated by generating the underlying random value uniformly. For each term in the sum V_i, we know in particular that $a_i \cdot b^{(i)}_i$ is perfectly simulated since $X_{i,j} \in W'$ for at least one $j \in [n]$ so $i \in J^{(i)}_b$ by construction. For the other terms in V_i, they can be perfectly simulated in the exact same way as we simulated the probes $V_{i,j}$ of the form (b) in the set W'. So c_i is perfectly simulated by summing all the perfectly simulated terms. This concludes the simulation proof for the set J.

Up until now, we have concluded that if we have a constructed set I_a of size at least $t+1$, then we can perfectly simulate the sets W and J without having a simulation failure on the input b. In the rest of the proof, we will consider that $|I_a| \leq t$ (along with $|I_a| \leq |W|$ by construction meaning that we have no failure on input a), and we will prove that we can perfectly simulate W and J with at most a simulation failure on b. Recall that we are also considering that $|W'| \geq d$ and $|W^{(1)} \cup \ldots \cup W^{(n)}| \leq d-1$.

Case 2.2: $|I_a| \leq t$. This means that the number of probes of the form (a) or (b) in W' with distinct values for the index i is at most t.

First, let us consider that $|I_a| \geq d$ (this is the case where $d = n - t \leq t+1$). Then, as proved earlier, and with t additional output shares in J of the form $c_i = V_i + X_i$, there are at least one X_ℓ remaining such that $\ell \notin J$ and $X_{\ell,j} \notin W$ for all $j \in [n]$. In this case, we can set $I_b = [n]$ and I_a as constructed with respect to the probes in W'. It is clear that all probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ and $J^{(1)}_b \cup \ldots \cup J^{(n)}_b$ are trivially simulated using $I_b = [n]$. In addition, all probes in W' are also perfectly simulated by construction of the set I_a and using $I_b = [n]$ and generating the necessary random values. This means that we can perfectly simulate all of the set of probes $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$. As for the set of output shares indexed in J. Let $i \in J$. If $i \in I_a$, then c_i is perfectly simulated using the share a_i and $I_b = [n]$, and by generating the necessary random values. Otherwise, if $i \notin I_a$, then in the same way as in claim 2, c_i can be masked by the random value $r_{i,\ell}$ (because $\ell \notin J$ and $X_{\ell,j} \notin W$ for all $j \in [n]$), so a_i is not needed for the simulation of c_i. This proves that we can perfectly simulate the output shares in J with I_a and $I_b = [n]$.

In the rest, we suppose that $|I_a| \leq d-1 \leq n-t-1$, *i.e.*, the number of probes of the form (a) or (b) in W' with distinct values for the index i is at most $d-1 \leq n-t-1$. In this case, and with t

additional output shares, we have at least one index k such that $k \notin J$ and for which there are no probes in W' of the form (a) or (b). In other terms, no partial sum of V_k is probed, no product of shares $a_k \cdot b_j^{(k)}$ or $p_{k,j}$ is probed, and no random value $r_{k,j}$ is probed. Now we reason on the number of probes of the form (c) in W':

- We first consider the special case where the number of $X_{i,j}$ probed (of form (c) in W') for distinct values of i is equal to n. In other terms, we have probes $X_{1,j_1}, \ldots, X_{n,j_n}$ for certain values $j_1, \ldots, j_n$. Since the set of probes W satisfies $|W| \leq n$ (because $2d-1 \leq n$), then this means that there are no remaining probes in the set W except for the n probes of the form (c) in W'. This is an easy case since we can let $I_b = [n]$ and $I_a = J$ (always without a failure on a since in the case where $|W| = n$, we have $d = t+1 = n-t$ so $|I_a| = |J| \leq \min(t, |W|)$ where $t \leq |W|$). This allows us to trivially simulate all output wires indexed in J, and since the remaining wires in W are just sums of random values, we can simulate them by generating the corresponding random values.
- Next, we consider that there is at least one index ℓ such that $X_{\ell,j} \notin W$ for all $j \in [n]$ (in other terms, the number of probes of the form $X_{i,j}$ for distinct values of i is at most $n-1$). Notice that this case is slightly different than the case of claims 2 and 3, since ℓ can be in the set J. In this case, we can let $I_b = [n]$ so that we can perfectly simulate all wires in $W^{(1)} \cup \ldots \cup W^{(n)}$ and $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$ using $I_b = [n]$, and we can perfectly simulate all wires in W' using I_a by construction and $I_b = [n]$ and generating the necessary random values. Next, we need to prove that we can perfectly simulate all output shares in J. Let $i \in J$. If $i \in I_a$, then c_i is perfectly simulated using a_i and $I_b = [n]$ and generating the necessary random values. Next, if $i \notin I_a$, then if $\ell \notin J$, we can use claim 2 to prove that we can replace the expression of $c_i = V_i + X_i$ by the random value $r_{i,l}$ and so the share a_i is not needed for the simulation of c_i (even if $X_{i,j}$ for a certain j is probed, the expression of V_i is still masked by $r_{i,l}$ and a_i is not needed to simulate X_i which is a sum of random values). Meanwhile, if $\ell \in J$, then we cannot directly use the random value $r_{i,\ell}$ to mask the expression of c_i. But since $X_{\ell,j} \notin W$ for all $j \in [n]$, and since $r_{k,\ell} \notin W$ because $k \notin I_a$ by assumption, then c_ℓ can be masked by the random value $r_{k,\ell}$, i.e $c_\ell = V_\ell + X_\ell \leftarrow r_{k,\ell}$. Since $i \in J$ and $k \notin J$, then $i \neq k$ and the random value $r_{i,\ell}$ does not appear anymore in X_ℓ in the expression of c_ℓ. Since $i \notin I_a$ then $r_{i,\ell}$ can be used to mask the expression of the output share c_i indexed in J and so the share a_i is not needed for the simulation of c_i. This proves that we can perfectly simulate all shares in J with the constructed sets I_a and $I_b = [n]$.

We managed to show that whenever the construction of the set I_a gives $|I_a| \leq t$, then we can perfectly simulate the sets W and J with at most a failure on input b and while still having $|I_a| \leq t$ and $|I_a| \leq |W|$.

By considering both cases $|I_a| \geq t+1$ and $|I_a| \leq t$, we covered all the cases for the simulation, and we proved that we can always perfectly simulate the set of probes W along with the set of output shares J while having a failure on at most one of the inputs. This concludes the proof of Lemma 16. □

I.2 Proof for TRPE2 property

Lemma 17. *The above multiplication gadget is (t, f_2)-TRPE2 of amplification order $d \geq \min(t+1, n-t)$*

Proof. To prove the lemma, we proceed in two steps through the following two lemmas 18 and 19.

Lemma 18. *Let W be a set of leaking wires as described above such that $|W| < \min(t+1, n-t)$. Then there exists a set J of $n-1$ output shares, such that W and J can be perfectly simulated from at most $\min(|W|, t) = |W|$ shares of each of the inputs a and b.*

Proof. We will construct the set of input shares indices I_a and the sets of output shares $J_b^{(k)}$ for $k \in [n]$ depending on the probes in the set W' (recall that $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$) as follows (we consider that all $J_b^{(k)}$ are empty at first since all the output shares of G_{refresh} can be probed directly in W'):

(a) For probes of form (a), we add index i to I_a, and index j to $J_b^{(k)}$ for $k \in [n]$.
(b) For probes of form (b), we add index i to I_a and to $J_b^{(k)}$ for $k \in [n]$.
(c) For probes of form (c), we add index i to $J_b^{(k)}$ for $k \in [n]$.

Observe that since $|W| < \min(t+1, n-t)$, then in particular $|W'| \leq \min(t+1, n-t) - 1 \leq t$, then $|I_a| \leq |W'| \leq |W| \leq t$ so we have no failure on the input a. Also, since $|J_b^{(k)}| \leq |W'| \leq t$ and $|W^{(k)}| < \min(t+1, n-t)$, then by the (t, f')-TRPE1 property of G_{refresh}, we will be able to simulate sets $J_b^{(k)}$ and $W^{(k)}$ from the set of input shares $I_b^{(k)}$ such that $|I_b^{(k)}| \leq |W^{(k)}| \leq t$ for $k \in [n]$. Thus, we can let $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)}$ and we have $|I_b| \leq |W^{(1)} \cup \ldots \cup |W^{(n)}| \leq |W| \leq t$, so we have no failure on the input b either. Until now, we have shown that we can simulate all sets $W^{(k)}$ and $J_b^{(k)}$ from I_b of size at most $\min(|W|, t) = |W|$. It remains to show that we can also perfectly simulate the set W' and a well chosen set J of $n-1$ output shares, from I_a and $J_b^{(k)}$ for $k \in [n]$. We will choose the set J from two subsets $J = J_1 \cup J_2$, where $J_1 = \{i \mid i \in J_b^{(k)} \text{ for any } k \in [n]\}$, and $J_2 \subset [n]$ is any set such that $J_1 \cap J_2 = \emptyset$ and $|J_1 \cup J_2| = n-1$. Let $\ell \in [n]$ be the index such that $\ell \notin J$. Since $|W| \leq \min(t+1, n-t) - 1 \leq n-1$, then by construction of the sets $J_b^{(k)}$, we have that $|J_b^{(1)} \cup \ldots \cup J_b^{(n)}| \leq n-1$, then for the index ℓ, we have that $\ell \notin J_b^{(k)}$ for all $k \in [n]$, then $X_{\ell,j} \notin W$ for any $j \in [n]$ by construction of the sets $J_b^{(k)}$. The value of X_ℓ will be useful to use the following claim.

Claim 4 *Let $i \in J$. Suppose that $V_{i,j} \notin W$ for all $j \in [n]$. Then the expression of $c_i = V_i + X_i$ can be masked by the random value $r_{i,\ell}$, in other terms $c_i \leftarrow r_{i,\ell}$.*

Proof. The proof of this claim is quite straightforward since we suppose that $V_{i,j} \notin W$ for all $j \in [n]$, so none of the partial sums $V_{i,j}$ has been probed. Then V_i in c_i contains $n-1$ random values. In particular, we know that $r_{i,\ell}$ and $p_{i,\ell}$ only appear in the expression of the probed output c_i, because if they were probed in W then we would have $\ell \in J_b^{(i)}$ by construction, but we suppose that $\ell \notin J_b^{(k)}$ for all $k \in [n]$. In addition, since $X_{\ell,j} \notin W$ for all $j \in [n]$ (because otherwise then by construction $\ell \in J_b^{(k)}$ which does not hold), then $r_{i,\ell}$ does not appear in any other expression of the probed wires in W, so we can simply use it to perfectly simulate c_i. □

We can now show that the sets W' and J can be perfectly simulated from the constructed sets I_a and $J_b^{(k)}$.

Simulation of W'. Probes of the form (a) can be perfectly simulated from the corresponding input shares in I_a and $J_b^{(k)}$, and by generating uniformly random values $r_{i,j}$ when necessary. Probes of the form (c) are also perfectly simulated by simply generating uniformly random values, since $X_{i,j} = r_{1,i} + \ldots + r_{j,i}$. As for probes of the form (b), we know that $i \in I_a$, then we look at each of the terms $p_{i,j'}$ for $j' \in [j]$ in $V_{i,j} = p_{i,1} + \ldots p_{i,j}$. For each $p_{i,j'}$, if $j' \in J_b^{(i)}$, then $p_{i,j'}$ can be perfectly simulated from the corresponding input shares and by generating uniformly at random $r_{i,j'}$. Otherwise, if $j' \notin J_b^{(i)}$, then that means that the wires $p_{i,j'}, r_{i,j'}$ and $X_{j'}$ are not probed in W'. That means that we can potentially replace $p_{i,j'}$ by a random value $r_{i,j'}$ since $r_{i,j'}$ does not appear in any other expression of the variables probed in W'. Meanwhile, we also need to check the case where $j' \in J$, since $c_{j'} = V_{j'} + X_{j'}$, and $r_{i,j'}$ is the one of the summed terms in the expression of $X_{j'}$:

- If $j' \notin J$, then we can replace $p_{i,j'}$ by a random value $r_{i,j'}$ since $r_{i,j'}$ does not appear in any other expression of the variables probed in W' and is not probed either through $c_{j'}$.
- If $j' \in J$, then we also know that $V_{j'} \notin W'$ (because otherwise we would have by construction $j' \in J_b^{(k)}$ for $k \in [n]$ which does not hold), then we know from claim 4 that $c_{j'}$ can be masked by the random value $r_{j',\ell}$, which masks $V_{j'} + X_{j'}$. Since $\ell \neq j'$ (because $\ell \notin J$ while $j' \in J$), then $r_{i,j'}$ does not appear anymore in any other wire expression of the probed variables in W or J except in the term $p_{i,j'}$ of $V_{i,j}$, so $r_{i,j'}$ can be used to mask the expression of $p_{i,j'}$.

By perfectly simulating each term $p_{i,j'}$ in $V_{i,j}$, we can perfectly simulate $V_{i,j}$. Thus, we can perfectly simulate all wires in W'.

Simulation of J. Let $i \in J$. Let us first consider the case where $V_{i,j} \notin W'$ for any $j \in [n]$, then by claim 4, the output share c_i can be masked by the random variable $r_{i,\ell}$, so c_i is perfectly simulated by generating a fresh random value. Otherwise, if $V_{i,j} \in W'$ for a certain $j \in [n]$, then we know that the value of $V_{i,j}$ is perfectly simulated as proven above. Now, let us check each term $p_{i,j'}$ for $j' \in [j+1, n]$. Actually, we can also perfectly simulate each of these terms like the terms $p_{i,j'}$ for $j' \in [j]$. Plus, the term $p_{i,i}$ is perfectly simulated by construction of the sets I_a and $J_b^{(i)}$ (because $V_{i,j} \in W'$). In addition, all terms in X_i in $c_i = V_i + X_i$ can be perfectly simulated by generating a fresh random value. Thus, c_i can be perfectly simulated by summing all of the perfectly simulated terms in it. This brings us to a perfect simulation of all output shares in J. We have shown that we can perfectly simulate any set of probes W of size at most $\min(t+1, n-t) - 1$ with a chosen set J of $n-1$ output shares, with at most $\min(|W|, t) = |W|$ shares of each of the inputs a and b. This concludes the proof of Lemma 18. □

Remark 3. We can observe that for this lemma to apply on G_{mult}, we don't need the pre-processing phase of the refresh on input b. In fact, you can see that during the construction of the sets $J_b^{(k)}$, we add each index to all of the sets for all $k \in [n]$. However, executing n refreshings on the input b will be necessary for the proof of the next result, specifically when we consider W such that $\min(t+1, n-t) \leq |W| < 2 \cdot \min(t+1, n-t)$.

To get back to the proof of Lemma 17, we also need the following result.

Lemma 19. *Let W be a set of leaking wires as described above such that $\min(t+1, n-t) \leq |W| < 2 \cdot \min(t+1, n-t)$. Then there exists a set J of $n-1$ output shares such that W and J can be perfectly simulated from sets of input shares I_a and I_b such that $|I_a| \leq \min(|W|, t)$ **or** $|I_b| \leq \min(|W|, t)$. In other terms, we have a simulation failure on at most one of the inputs a or b.*

Proof. Recall that the set W can be split into subsets $W = W' \cup W^{(1)} \cup \ldots \cup W^{(n)}$ as described above. We consider two cases.

Case 1: $|W'| < \min(t+1, n-t)$. This case is similar to the case of Lemma 18, so we can construct the set I_a in the same way as in the proof of Lemma 18, and we can eventually consider $I_b = [n]$. We know that $|I_a| \leq |W'| \leq \min(t+1, n-t) - 1 \leq t$, so there is no failure on the input a. And all probes in W' can be simulated like in the proof of Lemma 18 with I_a and trivially with $I_b = [n]$. Also, all probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ can be trivially simulated since we have access to the full input b. In addition, we choose the set J of size $n-1$ in the same way as in Lemma 18. Whenever $i \in J$ and $V_{i,j} \in W'$ for some $j \in [n]$, then $c_i = V_i + X_i$ is easily simulated using $I_b = [n]$ and the share a_i. If $i \in J$ but $V_{i,j} \notin W'$ for all $j \in [n]$, then as in the proof of Lemma 18, c_i in this case can be masked by the random value $r_{i,\ell}$ (because $|W'| < \min(t+1, n-t)$) and so simulating c_i amounts to generating uniformly at random the corresponding random value. Thus, W and J are perfectly simulated with at most $\min(|W|, t)$ shares of a and eventually the full input b.

Case 2: $|W'| \geq \min(t+1, n-t)$ *(and thus* $|W^{(1)} \cup \ldots \cup W^{(n)}| < \min(t+1, n-t)$*).* In this case, we will construct the sets I_a and $J_b^{(k)}$ from empty sets, in a way that we will have a simulation failure on at most one of the inputs a or b. We construct the mentioned sets depending on the probes in W' as follows:

(a) For probes of form (a), we add index i to I_a, and index j only to $J_b^{(i)}$.
(b) For probes of form (b), we add index i to I_a and only to $J_b^{(i)}$.
(c) For probes of form (c), we add index i to $J_b^{(k)}$ for all $k \in [n]$.

In the rest of the lemma, we will prove that if we have a failure on one of the inputs, we can still perfectly simulate W and a chosen set J of $n-1$ output shares without a failure on the other input. For this, we will consider two cases, the first where we have a failure on input a, the second where we don't have a failure on input a, and so we can eventually have a failure on input b.

Case 2.1: simulation failure on input a, i.e. $I_a > t$. This means that the set I_a is of size $|I_a| \geq t+1 \geq \min(t+1, n-t)$ (this is because by construction $|I_a| \leq |W|$, so to have $|I_a| > \min(|W|, t)$, we must have $|I_a| > t$). We will first start by showing that the sets $W^{(k)}$ and $J_b^{(k)}$ can be perfectly simulated using the simulator of G_{refresh} without a failure on the input b. Next, we will show that W' and a well chosen set of $n-1$ output shares in J can be perfectly simulated using I_a and $J_b^{(k)}$.

Since we only add shares indices to I_a, when we have probes of the form (a) or (b), this means that we have at least $t+1$ probes of these two forms with $t+1$ different values for the index i. In addition, since we have at least $t+1$ probes (a) or (b) with distinct values for the index i, then this also means that each of the sets $J_b^{(i)}$ has at most one share of $b^{(i)}$ added to it. In other terms, $|J_b^{(k)}| \leq 1$ for each $k \in [n]$ (from the probes (a) and (b) with distinct indices i).

Now let us consider the remaining probes in W which are either in W' of the form (c) or in $W^{(1)} \cup \ldots \cup W^{(n)}$ or of the forms (a) or (b) with $i \in I_a$. Since $|I_a| \geq t+1 \geq \min(t+1, n-t)$, then there are at most $\min(t+1, n-t) - 1$ of these remaining probes. Without loss of generality, we consider that there are exactly $\min(t+1, n-t) - 1$ instead of at most $\min(t+1, n-t) - 1$ probes. Let m be the number of probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ and $d - 1 - m$ the remaining probes in W' of the form (c) or (a)/(b) with $i \in I_a$. Since each wire in W' of the form (c) or (a)/(b) with $i \in I_a$ results in adding at most one more share index to each $J_b^{(k)}$ for $k \in [n]$, then we have

$|J_b^{(k)}| \leq 1 + (\min(t+1, n-t) - 1 - m) = d - \min(t+1, n-t)$. And $|W^{(1)} \cup \ldots \cup W^{(n)}| \leq m$, in particular $|W^{(k)}| \leq m$ for any $k \in [n]$.

- if $m = 0$, then $W^{(k)} = \emptyset$ for any $k \in [n]$, and $|J_b^{(k)}| \leq \min(t+1, n-t) \leq \lfloor \frac{n+1}{2} \rfloor \leq n-1$, so by the TRPE property of G_{refresh} for any $t \leq n-1$, all of the $J_b^{(k)}$ sets can be perfectly simulated with no knowledge of the input shares of b since $W^{(k)} = \emptyset$, so $I_b^{(k)} = \emptyset$. Hence, $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)} = \emptyset$ and we have no simulation failure on the input b.
- if $m > 0$, then $|J_b^{(k)}| \leq \min(t+1, n-t) - 1 \leq t$ and $|W^{(1)} \cup \ldots \cup W^{(n)}| < \min(t+1, n-t)$, in particular $|W^{(k)}| \leq \min(t+1, n-t) - 1$ for each $k \in [n]$. Thus, by the (t, f)-TRPE property of the refresh gadget G_{refresh} achieving the amplification order d, we can perfectly simulate both sets for each $k \in [n]$ with $I_b^{(k)}$ such that $|I_b^{(k)}| \leq |W^{(k)}|$. Thus, we can let $I_b = I_b^{(1)} \cup \ldots \cup I_b^{(n)}$ so we can have $|I_b| \leq |W^{(1)} \cup \ldots \cup W^{(n)}| \leq \min(t+1, n-t) - 1 \leq t$, and we can perfectly simulate $W^{(1)} \cup \ldots \cup W^{(n)}$ along with $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$ from the set I_b without a simulation failure on input b.

So far we proved that if we have $|I_a| > t$, then we must have $|I_b| \leq t$, and $W^{(1)} \cup \ldots \cup W^{(n)}$ can be perfectly simulated along with $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$ from the set I_b. Next we need to prove that we can perfectly simulate W' and a chosen set J of $n-1$ output shares, from these sets I_a and $J_b^{(1)} \cup \ldots \cup J_b^{(n)}$. We consider two sub-cases.

Case 2.1.1: $I_a = [n]$. In this case, since $|W| \geq n$ (from I_a) and $|W| < 2\min(t+1, n-t) \leq n+1$, then $|W| = n$ and all probes in W are all in W' of the form (a) or (b) with n distinct values for the index i. We neither have probes in $W^{(1)} \cup \ldots \cup W^{(n)}$ nor in W' of the form (c). Thus, we can reconstruct each $|J_b^{(k)}|$ of size at most $n-1$ without having a failure on the input b (since G_{refresh} is (t', f')-TRPE for any $t' \leq n-1$ achieving $d' = \min(t'+1, n-t')$ and all $W^{(k)}$ are empty). We consider two cases:

- Suppose that for each $i \in [n]$, we have at least one probe in W' of the form $r_{k,i}$ or $p_{k,i}$ for some $k \in [n]$, note this probe $q_{k,i} \in \{r_{k,i}, p_{k,i}\}$. Since also $I_a = [n]$, this means that we have probes $q_{k_1,1}, \ldots, q_{k_n,n}$, such that $k_1 \neq \ldots \neq k_n$. Because $|W'| = n$, then all probes in W' are of the form (a) (specifically $q_{k,i}$), and we have no probes of the form $V_{i,j}$ for any $i, j \in [n]$. In this case, the simulation of the probes in W' is straightforward by construction of the sets I_a and $J_b^{(k)}$. As for the set J, we let $J \subset [n]$ such that $|J| = n-1$ (any set of $n-1$ shares works), and let $\ell \in [n]$ such that $\ell \notin J$. Observe that out of all the random values $r_{i,\ell}$ in X_ℓ in the expression of $c_\ell = V_\ell + X_\ell$, only the random value $r_{k_\ell,\ell}$ appears in the expression of the probe $q_{k_\ell,\ell}$ in the set W', and all other random values $r_{i,\ell}$ for $i \neq k_\ell$ do not appear in any other probed variable in W' (since $W' = \{q_{k_1,1}, \ldots, q_{k_\ell,\ell}, \ldots, q_{k_n,n}\}$, such that $k_1 \neq \ldots \neq k_n$). Then, for each $i \in J$ such that $i \neq k_\ell$, the expression of $c_i = V_i + X_i$ can be masked by the random value $r_{i,\ell}$, so simulating c_i amounts to generating a fresh random value $r_{i,\ell}$. Now let's check $i = k_\ell \in J$. Since $q_{k_\ell,\ell}$ is probed, then we cannot mask the expression of c_{k_ℓ} using $r_{k_\ell,\ell}$. However, for each $i \in J$ with $i \neq k_\ell$, we have that c_i is masked by $r_{i,\ell}$. Since $c_i = V_i + X_i$, and the random value $r_{k_\ell,i}$ is one of the terms in X_i, then $r_{k_\ell,i}$ does not appear anymore in the expression of c_i. And since $W' = \{q_{k_1,1}, \ldots, q_{k_\ell,\ell}, \ldots, q_{k_n,n}\}$, such that $k_1 \neq \ldots \neq k_n$ and $q_{k_\ell,\ell} \in W'$, then $q_{k_\ell,i} \notin W'$ and $r_{k_\ell,i}$ only appears in the expression of c_{k_ℓ}, so c_{k_ℓ} can be masked by the random value $r_{k_\ell,i}$. Thus, we proved that we can perfectly simulate the sets W' and J using the sets I_a and $J_b^{(k)}$.

– Next, we suppose that there exists $\ell \in [n]$ such that we have no probes in W' of the form $q_{k,\ell} \in \{r_{k,\ell}, p_{k,\ell}\}$. In this case, we choose $J = [n] \setminus \{\ell\}$. Next, we show that we can perfectly simulate all probes in W' and output shares in J for each $i \in I_a = [n]$. For this, first let $i \in [n] \setminus \{\ell\}$ (notice that we automatically have $i \in J$):
  - if for the considered i, the probe in W' is of the form (a) i.e $a_i, b_j^{(i)}, a_i \cdot b_j^{(i)}, p_{i,j} = a_i \cdot b_j^{(i)} + r_{i,j}$, then the simulation of this probe is trivial by construction of the sets I_a and $J_b^{(i)}$. In addition, we know that $r_{i,\ell}$ and $p_{i,\ell}$ are not probed in W' by assumption, and since $X_{\ell,j} \notin W'$ for all $j \in [n]$, then the random value $r_{i,\ell}$ only appears in the expression of $c_i = V_i + X_i$ (specifically in V_i), and so can be used to mask c_i. So simulating c_i amounts to generating a fresh random value.
  - if for the considered i, the probe in W' is of the form (b), i.e $V_{i,j} \in W'$ for a certain $j \in [n]$ (there is a unique probe of this form), then:
    * either $j < \ell$, and so the random value $r_{i,\ell}$ can be used as before to mask the expression of $c_i + V_{i,j}$, and since in this case $V_{i,j}$ contains less than $n-1$ terms $p_{i,j'}$, then we can add all the necessary shares of $b^{(i)}$ to $J_b^{(i)}$without having a failure on b (recall that $W^{(i)} = \emptyset$). So we can perfectly simulate $V_{i,j}$ and $c_i + V_{i,j}$, and hence also simulate c_i.
    * or $j \geq \ell$, and so the random value $r_{i,\ell}$ can be used in this case to mask the expression of $V_{i,j}$ so simulating $V_{i,j}$ amounts to generating a fresh random value, and since $V_{i,j}$ is the sum of at least two terms of the form $p_{i,j'}$, then $c_i + V_{i,j}$ can be simulated with at most $n-1$ shares of $b^{(i)}$, so there is no simulation failure on input $b^{(i)}$. So we can perfectly simulate $V_{i,j}$ and $c_i + V_{i,j}$, and hence also simulate c_i.

  Next we consider the case of the probe $V_{\ell,j}$:
  - either $j < \ell$, and so in this case $V_{\ell,j}$ contains less than $n-1$ terms $p_{\ell,j'}$, then we can add all the necessary shares of $b^{(\ell)}$ to $J_b^{(\ell)}$without having a failure on b (recall that $W^{(\ell)} = \emptyset$). So we can perfectly simulate $V_{\ell,j}$ using the input share a_ℓ, the input shares of $b^{(\ell)}$ and by generating necessary random values.
  - or $j \geq \ell$, and so the random value $r_{\ell,\ell}$ can be used in this case to mask the expression of $V_{\ell,j}$ so simulating $V_{\ell,j}$ amounts to generating a fresh random value.

  Thus, also in this case, we can perfectly simulate W' and a chosen set of $n-1$ output shares without a failure on input b, using $I_a = [n]$.

This concludes the simulation for the special case where $I_a = [n]$.

Case 2.1.2: $I_a \subset [n]$ such that $|I_a| \leq n-1$. Let k such that $k \notin I_a$. Recall that $|I_a| \geq t+1 \geq \min(t+1, n-t)$ and $|W| < 2 \cdot \min(t+1, n-t)$, then there are at most $\min(t+1, n-t) - 1 \leq t \leq n-1$ probes remaining either in $W^{(1)} \cup \ldots \cup W^{(n)}$, of the form (c) in W', or of the form (a)/(b) with $i \in I_a$. Thus, there exists at least one index $\ell \in [n]$ such that $X_{\ell,j} \notin W'$ for all $j \in [n]$. In this case, we choose $J = [n] \setminus \{\ell\}$. Next, we will prove that we can perfectly simulate the sets W' and J from the constructed sets I_a and $J_b^{(k)}$, using the following claims.

Claim 5 *Let $i \in J$. Suppose that $i \notin I_a$. Then the expression of $c_i = V_i + X_i$ can be masked by the random value $r_{i,\ell}$, in other terms $c_i \leftarrow r_{i,\ell}$.*

Proof. This claim can be proved easily, since we suppose that $i \notin I_a$ so the random value $r_{i,\ell}$ and $p_{i,\ell}$ are not probed in W'. In addition, since $\ell \notin J$ and $X_{\ell,j} \notin W$ for all $j \in [n]$, then the random value $r_{i,\ell}$ does not appear in any other probed wire expression except in c_i, then c_i can be masked by the random value $r_{i,\ell}$. □

Claim 6 *Let $i \in J$. Suppose that $X_{i,j} \notin W$ for any $j \in [n]$. Suppose that $i \in I_a$. Then the expression of $c_i = V_i + X_i$ can be masked by the random value $r_{k,i}$, in other terms $c_i \leftarrow r_{k,i}$.*

Proof. Since we suppose that $k \notin I_a$, then the random value $r_{k,i}$ or $p_{k,i}$ or $V_{k,j}$ for all $j \in [n]$ are not probed in W. Then, if $k \notin J$, then the random value $r_{k,i}$ does not appear in the expression of any other probed wire in W or J and c_i can be masked by the random value $r_{k,i}$ (Recall that $c_i = V_i + X_i$ and $X_i = r_{1,i} + \ldots + r_{n,i}$). Otherwise, if $k \in J$, then by Claim 5, $c_k = V_k + X_k$ can be masked by $r_{k,\ell}$ and so c_i can also be masked by $r_{k,i}$ since $i \neq \ell$ (because $i \in J$ and $\ell \notin J$) and $i \neq k$ (because $i \in I_a$ and $k \notin I_a$). □

Probes of the forms (a) or (c) in W' are trivially simulated using the constructed sets of input shares, and generating the necessary random values. Let us now check the probes of the form (b). Let $V_{i,j} = p_{i,1} + \ldots + p_{i,j}$ be such a probe. Let us consider each of the terms $p_{i,j'}$ for $j' \in [j]$. if $j' = i$, then by construction $p_{i,i}$ is perfectly simulated using a_i and $b_i^{(i)}$ and by generating the random value $r_{i,i}$ if needed. Otherwise, let $j' \neq i$. If $j' \in J_b^{(i)}$ then the simulation of $p_{i,j'}$ is straightforward. Otherwise if $j' \notin J_b^{(i)}$, then we know that none of the wires $r_{i,j'}$ or $p_{i,j'}$ or $X_{j',s}$ for all $s \in [n]$ are probed in W'. Thus, $r_{i,j'}$ can be eventually used to mask the expression of $p_{i,j'}$ without the need of the share $b_{j'}^{(i)}$ for the simulation. Meanwhile, we still need to check if $j' \in J$, since $r_{i,j'}$ appears in $X_{j'}$ in the expression of $c_{j'} = V_{j'} + X_{j'}$.

- If $j' \in J$ and $j' \notin I_a$, then by claim 5, $c_{j'}$ can be masked by the random value $r_{j',\ell}$ and so $r_{i,j'}$ does not appear in the expression of $X_{j'}$ in $c_{j'}$ anymore, and $r_{i,j'}$ can be used to mask $p_{i,j'}$.
- Otherwise, if $j' \in J \cap I_a$, then by claim 6 $c_{j'}$ can be masked by the random value $r_{k,j'}$ and so $r_{i,j'}$ does not appear in the expression of $X_{j'}$ in $c_{j'}$ anymore, and $r_{i,j'}$ can be used to mask $p_{i,j'}$ (since $i \notin k$).

Thus, each term $p_{i,j'}$ in $V_{i,j}$ can be perfectly simulated and thus $V_{i,j} = p_{i,1} + \ldots + p_{i,j}$ can be perfectly simulated. This concludes the simulation of the set W'.

We now focus on the simulation of J. Let $i \in J$. If $i \notin I_a$, then by claim 5, c_i is perfectly simulated by generating the random value $r_{i,\ell}$. Otherwise, let $i \in I_a$. If $X_{i,j} \notin W$ for any $j \in [n]$, then by claim 6, c_i is perfectly simulated by generating the random value $r_{k,i}$. Otherwise, we can show that we can perfectly simulate each term in $c_i = V_i + X_i$. In particular, each term in X_i can be simulated by generating the underlying random value uniformly. For V_i, we know in particular that $a_i \cdot b_i^{(i)}$ is perfectly simulated since $X_{i,j} \in W'$ for at least one $j \in [n]$ so $i \in J_b^{(i)}$ by construction. For the other terms in V_i, they can be perfectly simulated in the exact same way as we simulated the probes $V_{i,j}$ of the form (b) in the set W'. So c_i is perfectly simulated by summing all the perfectly simulated terms. This concludes the simulation proof for the set J.

Up until now, we have concluded that if we have a constructed set I_a of size at least $t+1$, then we can perfectly simulate the sets W and a chosen set J of $n-1$ output shares, without having a simulation failure on the input b. In the rest of the proof, we will consider that $|I_a| \leq t$, and we will prove that we can perfectly simulate W and J with at most a simulation failure on b. Recall that we are also considering that $|W'| \geq d$ and $|\mathbf{W^{(1)}} \cup \ldots \cup \mathbf{W^{(n)}}| \leq \mathbf{d-1}$.

Case 2.2: $|I_a| \leq t$. This means that the number of probes of the form (a) or (b) in W' with distinct values for the index i is at most t.

First, let us check the special case where the number of probes of the form (c) in W' with different values for the index i is equal to n (notice that this cannot occur when we have $|I_a| \geq t+1$). Since $|W| \leq 2 \cdot \min(t+1, n-t) - 1 \leq n$, then we have $W = \{X_{1,j_1}, \ldots, X_{n,j_n}\}$ for certain $j_1, \ldots, j_n \in [n]$. So we can let $J_b^{(k)} = [n]$ for all $k \in [n]$ and $I_b = [n]$, and by construction $I_a = \emptyset$. In this case, we choose $J = [n-1]$. The simulation of the set W is straightforward since all wires of the form (c) are just sums of random values. Then, let us consider the output shares in J.

- If for at least one $\ell \in J$, we have $X_{\ell,n} \in W$, we can mask the expression of $X_{\ell,n}$ by the random value $r_{n,\ell}$ (because there are no probes of the form (a) or (b) in W and $n \notin J$, so $r_{n,\ell}$ only appears in the expression of $X_{\ell,n}$). Recall that $X_{\ell,n} = r_{1,\ell} + \ldots + r_{n,\ell}$, so $r_{n,\ell}$ masks all random values $r_{j,\ell}$ for $j \in [n-1]$. Each of the random values $r_{j,\ell}$ for $j \in [n-1] \setminus \{\ell\}$ can be used to mask the corresponding output share c_j for $j \in J$ because there are no probes of the form (a) or (b) in W and $X_{\ell,n}$ is already masked by $r_{n,\ell}$, so $r_{j,\ell}$ only appears in the expression of $c_j = V_j + X_j$, so $c_j \leftarrow r_{j,\ell}$. As for the output c_ℓ, we can let $I_a = \{\ell\}$ and we can perfectly simulate c_ℓ using a_ℓ and $I_b = [n]$. Since $|W| = n$, so $\min(t+1, n-t) > \dfrac{n}{2} \geq 1$, so we have no failure on the input a, and we can perfectly simulate the chosen set J and the set of probes W.
- Now we consider that for all $W = \{X_{1,j_1}, \ldots, X_{n,j_n}\}$, we have $j_1 < n, \ldots, j_n < n$. In this case, the set W is also trivially simulated by generating random values, and we let $J = [n-1]$. Since, $n \notin J$ and there are no probes of the form (a) or (b) in W, then the random values $r_{n,i}$ for $i \in [n-1]$ only appear in the expression of the output share $c_i = V_i + X_i$ each. And since all probes of the form $X_{i,j}$ are such that $j < n$, then we can let $r_{n,i}$ be used to mask the expression of $c_i + X_{i,j}$ because $r_{n,i}$ does not appear in $X_{i,j}$ for $j < n$, i.e $c_i + X_{i,j} \leftarrow r_{n,i}$. By perfectly simulating the masked expression of $c_i + X_{i,j}$ and the sum of random values $X_{i,j}$, we can perfectly simulate c_i. Thus, simulating all output shares in J amounts to generating random values uniformly. So we can perfectly simulate sets W and J from $I_a = \emptyset$ and $I_b = [n]$.

Next, we suppose that the number of probes of the form (c) in W' with different values for the index i is strictly smaller than n. So, there is at least one index ℓ such that $X_{\ell,j} \notin W'$ for all $j \in [n]$. We let $J = [n] \setminus \{\ell\}$. We also let $I_b = [n]$ and we keep the set I_a as constructed according to the probes in the set W'. Observe that all probes in W' are perfectly simulated by easily using the set I_a and $I_b = [n]$. As for the output shares in J, observe that for each $i \in J$ such that $i \notin I_a$, we can use claim 5 to mask the expression of c_i by $r_{i,\ell}$, and so the share a_i is not needed for the simulation of c_i. Otherwise, if $i \in J \cap I_a$, then c_i is perfectly simulated using a_i and $I_b = [n]$.

This proves that whenever $i \notin I_a$, the output share c_i can be simulated without the need of the share a_i. Since we suppose that $|I_a| \leq t$, then we conclude that we can perfectly simulate W and a chosen set of $n-1$ output shares J with at most a simulation failure on input b.

By considering both cases $|I_a| \geq t+1$ and $|I_a| \leq t$, we covered all the cases for the simulation, and we proved that we can always perfectly simulate the set of probes W along with a chosen set of $n-1$ output shares J while having a failure on at most one of the inputs. This concludes the proof of Lemma 19. □

From Lemmas 18 and 19, we conclude that G_{mult} is (t, f_2)-TRPE2 of amplification order $d \geq \min(t+1, n-t)$. This concludes the proof of Lemma 17. □

Abstract

We live in a world in which cryptography has become ubiquitous. Devices around us are constantly processing cryptographic computations to ensure the confidentiality and the authenticity of our communications. Over the last forty years, the scientific community and the industry have converged towards the paradigm of provable security for cryptographic algorithms and protocols: they should come with a security proof formally stating their security under well-studied computational hardness assumptions. Such a proof is usually stated in the black-box model in which the adversary is assumed to have an input-output access to the cryptographic mechanism.

Unfortunately, this black-box model was shown insufficient in the late 1990's with the discovery of side-channel attacks. These attacks exploit the physical leakage of a cryptographic implementation (e.g. its execution time, power consumption, or electromagnetic emanation) to practically break it, although the underlying mechanism might achieve strong black-box security. While a lot of progress was made over the last decades to design practical countermeasures against side-channel attacks, achieving provable security for cryptographic implementations under this threat is still largely a work in progress.

This thesis presents some contributions toward the provable security of cryptographic implementations in the presence of side-channel leakage. Our approach relies on masking whose principle is to apply secret sharing at the computation level. Our results have contributed to the formalization of masking security, the construction of efficient masking schemes, the formalization of practically-relevant side-channel leakage models, and the construction of masking schemes achieving provable security under these models.