

# How to Securely Compute with Noisy Leakage in Quasilinear Complexity

Dahmun Goudarzi<sup>1,2</sup>, Antoine Joux<sup>3</sup>, and Matthieu Rivain<sup>1</sup>

<sup>1</sup> CryptoExperts, Paris, France

<sup>2</sup> ENS, CNRS, INRIA and PSL Research University, Paris, France

<sup>3</sup> 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  $\lambda$ ). 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<sup>+</sup>09]). In particular for a leakage function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^\lambda$ , the parameter  $\lambda$  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  $\delta$ . 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  $(x_1, x_2, \dots, x_n)$  satisfying  $x_1 + x_2 + \dots + x_n = x$  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  $x_1, x_2, \dots, x_n$  the information on  $x$  becomes negligible as  $n$  grows. Specifically, they show that for any  $\delta$ -noisy leakage function  $f$ , the mutual information between  $x$  and the leakage  $(f(x_1), f(x_2), \dots, f(x_n))$  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  $\delta$ -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  $\delta$ -*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  $\delta'$ -noisy leakage  $f(x)$  can be simulated from a  $\delta$ -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  $\delta$ -random probing leakage of a computation contains no significant information, then neither does any  $\delta'$ -noisy leakage of this computation as long as  $\delta \leq \delta' \cdot |\mathcal{X}|$ . The ISW scheme is therefore secure against  $\delta'$ -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  $\delta$ -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 their 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<sup>+</sup>15, DIK10]. As explained in [ADD<sup>+</sup>15], the use of packed secret sharing allows to securely compute an addition or a multiplication on  $\ell$  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  $\ell$  from  $O(sn^2 \log n)$  to  $O(s \log sn^2 \log n/\ell)$ . 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<sup>+</sup>16, BBP<sup>+</sup>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]).<sup>4</sup> In our scheme, each element  $a \in \mathbb{F}$  is encoded into a random tuple  $(a_0, a_1, \dots, a_{n-1})$  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

<sup>4</sup> 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  $\omega$  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  $\delta$ -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  $\delta$ -random-probing adversary can be upper bounded by  $\text{negl}(\lambda) + \text{negl}'(n)$  where  $\text{negl}$  and  $\text{negl}'$  are some negligible functions and where  $\lambda$  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  $\omega$ ).

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  $\delta'$ -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  $\Pi$  secure under a  $\delta$ -random-probing leakage gives rise to a functionally equivalent program  $\Pi'$  that is secure under a  $\delta'$ -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  $\delta'$ -noisy leakage model with  $\delta' = O(1/((\log n)^2 \log \log n))$  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  $\rho$  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  $\rho$ . For the sake of simplicity, in the sequel we shall omit the parameter  $\rho$  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  $f : \mathcal{X} \rightarrow \mathcal{Y}$  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  $(x | f(x))$  is assumed to be bounded by some bias  $\varepsilon$ . 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:

$$\Delta(X; X') = \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr(X = x) - \Pr(X' = x)| . \quad (1)$$

The notion of noisy leakage function is then formalized as follows:

**Definition 1 ([PR13]).** A  $\varepsilon$ -noisy leakage function is a randomized function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  satisfying

$$\sum_{y \in \mathcal{Y}} \Pr(f(X) = y) \cdot \Delta(X; (X | f(X) = y)) \leq \varepsilon , \quad (2)$$

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.<sup>5</sup>

**Definition 2.** A  $\varepsilon$ -random-probing leakage function is a randomized function  $\phi : \mathcal{X} \rightarrow \mathcal{X} \cup \{\perp\}$  satisfying

$$\phi(x) = \begin{cases} \perp & \text{with probability } 1 - \varepsilon \\ x & \text{with probability } \varepsilon \end{cases} \quad (3)$$

It can be checked that such a function is a special case of  $\varepsilon$ -noisy leakage function.<sup>6</sup> 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  $\phi$  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  $f : \mathcal{X} \rightarrow \mathcal{Y}$  be a  $\varepsilon$ -noisy leakage function with  $\varepsilon < \frac{1}{|\mathcal{X}|}$ . There exists a  $\delta$ -random-probing leakage function  $\phi : \mathcal{X} \rightarrow \mathcal{X} \cup \{\perp\}$  and a randomized function  $f' : \mathcal{X} \cup \{\perp\} \rightarrow \mathcal{Y}$  such that for every  $x \in \mathcal{X}$  we have

$$f(x) = f'(\phi(x)) \quad \text{and} \quad \delta \leq \varepsilon \cdot |\mathcal{X}| . \quad (4)$$

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  $\delta$ -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

$$\psi(\ell, N) = \exp\left(-\frac{(\ell - \delta N)^2}{\ell + \delta N}\right) \quad (5)$$

If  $\ell \leq \alpha n$  and  $N = \beta n$ , for some  $\alpha, \beta$  and  $n$  with  $\alpha/\beta > \delta$ , the above gives

$$\psi(\alpha n, \beta n) = \exp\left(-\frac{(\alpha - \delta\beta)^2}{\alpha + \delta\beta} n\right) \quad (6)$$

<sup>5</sup> Note that we use a different terminology from [DDF14] where these are called  $\varepsilon$ -identity functions.

<sup>6</sup> To be tighter: a  $\varepsilon$ -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*  $\text{Enc}$  is a randomized function that maps an element  $x \in \mathbb{F}$  to a  $n$ -tuple  $\text{Enc}(x) \in \mathbb{F}^n$ , where  $n$  is called the encoding length, and for which a deterministic function  $\text{Dec} : \mathbb{F}^n \rightarrow \mathbb{F}$  exists that satisfies  $\Pr(\text{Dec}(\text{Enc}(x)) = x) = 1$  for every  $x \in \mathbb{F}$  (where the latter probability is taken over the encoding randomness).

Consider an *arithmetic program*  $P$  taking a string  $\mathbf{x} \in \mathbb{F}^s$  as input and executing a sequence of instructions of the form  $\mathbf{m}_i \leftarrow \mathbf{m}_j * \mathbf{m}_k$ , where  $*$  denotes some operations over  $\mathbb{F}$  (addition, subtraction, or multiplication) and where  $[\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_T]$  denotes the memory of the program which is initialized with  $\mathbf{x}$  (and some constants). To achieve leakage security, the program  $P$  is transformed into a functionally equivalent arithmetic program  $\Pi$  taking as input an encoded input  $\text{Enc}(\mathbf{x})$  (where the encoding simply applies to each coordinate of  $\mathbf{x}$ ). According to the defined leakage model, each executed instruction of  $\Pi$  is then assumed to leak some noisy function  $f(\mathbf{m}_j, \mathbf{m}_k)$  of its pair of inputs. It is further assumed that  $\Pi$  includes random sampling instructions  $\mathbf{m}_i \leftarrow \mathbb{F}$  that each leaks a noisy function of its output  $f(\mathbf{m}_i)$ . We denote the overall leakage by  $\mathcal{L}(\Pi, \mathbf{x})$ . The compiler is then said to be *leakage secure* if an observation of  $\mathcal{L}(\Pi, \mathbf{x})$  does not reveal significant information about  $\mathbf{x}$ . More specifically, the leakage  $\mathcal{L}(\Pi, \mathbf{x})$  must be indistinguishable from the leakage  $\mathcal{L}(\Pi, \mathbf{x}')$  for every  $\mathbf{x}' \in \mathbb{F}^s$ . This security notion is formalized as follows:

**Definition 3 (Leakage Security).** *The program  $\Pi$  is  $\varepsilon$ -leakage secure if every adversary  $\mathcal{A}$  has advantage at most  $\varepsilon$  of distinguishing  $\mathcal{L}(\Pi, \mathbf{x}_0)$  from  $\mathcal{L}(\Pi, \mathbf{x}_1)$  for chosen  $\mathbf{x}_0$  and  $\mathbf{x}_1$ , i.e. we have:*

$$\text{Adv}_{\mathcal{A}}^{\Pi} := \left| \text{Succ}_{\mathcal{A}}^{\Pi} - \frac{1}{2} \right| \leq \varepsilon \quad (7)$$

where

$$\text{Succ}_{\mathcal{A}}^{\Pi} = \Pr \left( \begin{array}{l} (\mathbf{x}_0, \mathbf{x}_1, \mu) \leftarrow \mathcal{A}(\perp) \\ b \leftarrow \{0, 1\} \\ \ell \leftarrow \mathcal{L}(\Pi, \mathbf{x}_b) \end{array} : \mathcal{A}(\mathbf{x}_0, \mathbf{x}_1, \mu, \ell) = b \right). \quad (8)$$

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  $\mathbf{x}_0$  and  $\mathbf{x}_1$ . 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  $\mathcal{A}(\mathbf{x}_0, \mathbf{x}_1, \mu, \ell)$  takes values over  $\{0, 1\}$  for every input  $(\mathbf{x}_0, \mathbf{x}_1, \mu, \ell) \in \mathbb{F}^s \times \mathbb{F}^s \times \{0, 1\}^* \times \text{Im}(\mathcal{L})$ .

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  $\Pi$  be an arithmetic program that is  $\varepsilon$ -leakage secure wrt  $\delta$ -random-probing leakage functions. Then  $\Pi$  is  $\varepsilon$ -leakage secure wrt  $\delta'$ -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  $\xi$  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  $\omega$ -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  $\text{Enc}$  maps an element  $a \in \mathbb{F}$  to a random element  $\omega \in \mathbb{F}^*$  and a random uniform  $\omega$ -encoding of  $a$ :

$$\text{Enc}(a) = \langle \omega, (a_0, a_1, \dots, a_{n-1}) \rangle. \quad (9)$$

The corresponding decoding function  $\text{Dec}$  is defined as:

$$\text{Dec}(\langle \omega, (a_0, a_1, \dots, a_{n-1}) \rangle) := \text{Dec}_\omega(a_0, a_1, \dots, a_{n-1}) := \sum_{i=0}^{n-1} a_i \omega^i \quad (10)$$

It is easy to check that we have  $\Pr(\text{Dec}(\text{Enc}(a)) = a) = 1$  for every  $a \in \mathbb{F}$ . It is worth noting that the security of our scheme does not rely on the secrecy of  $\omega$  but on its uniformity. Besides, we will consider that  $\omega$  is systematically leaked to the adversary.

### 3.2 Multiplication of Encoded Variables

Let  $(a_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $a$  and  $(b_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $b$ . To compute an  $\omega$ -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  $j \in \llbracket 0, 2n-1 \rrbracket$ . These evaluations are then pairwise multiplied to get evaluations of the product  $s_j = (2n)^{-1} u_j \cdot v_j$  for  $j \in \llbracket 0, 2n-1 \rrbracket$  (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  $\omega$ -encoding from the  $2n$ -size  $\omega$ -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  $3n \log n$  arithmetic instructions (a detailed description is given in Appendix A).

The overall process is summarized as follows:

$$\begin{aligned} (u_0, u_1, \dots, u_{2n-1}) &\leftarrow \text{NTT}_\xi(a_0, a_1, \dots, a_{n-1}, 0, \dots, 0) \\ (r_0, r_1, \dots, r_{2n-1}) &\leftarrow \text{NTT}_\xi(b_0, b_1, \dots, b_{n-1}, 0, \dots, 0) \\ (s_0, s_1, \dots, s_{2n-1}) &\leftarrow (2n)^{-1} (u_0 \cdot r_0, u_1 \cdot r_1, \dots, u_{2n-1} \cdot r_{2n-1}) \\ (t_0, t_1, \dots, t_{2n-1}) &\leftarrow \text{NTT}_{\xi^{-1}}(s_0, s_1, \dots, s_{2n-1}) \\ (c_0, c_1, \dots, c_{n-1}) &\leftarrow \text{compress}(t_0, t_1, \dots, t_{2n-1}) \end{aligned}$$

*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  $\omega$  (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  $i \in \llbracket 0, n-1 \rrbracket$ . 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  $\omega$  be a uniform random element of  $\mathbb{F}^*$ , let  $(a_i)_{i=0}^{n-1}$  be a uniform  $\omega$ -encoding of some variable  $a$  and let  $\delta < 1/(6 \log n)$ . The  $\text{NTT}_\xi$  procedure on input  $(a_i)_{i=0}^{n-1}$  is  $\varepsilon$ -leakage secure in the  $\delta$ -random-probing leakage model, where

$$\varepsilon = \frac{n}{|\mathbb{F}|} + \exp\left(-\frac{(1 - 6\delta \log n)^2}{4} n\right). \quad (11)$$

The rest of the section gives a proof of Theorem 1. During the computation of the NTT on an  $\omega$ -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  $\alpha_i$ 's are constant coefficients over  $\mathbb{F}$ . In the following, we shall use the notation  $[v] = (\alpha_0, \alpha_1, \dots, \alpha_{n-1})$  for the vector of coefficients of such an intermediate variable. Similarly, we shall denote  $[a] = (1, \omega, \omega^2, \dots, \omega^{n-1})$  since we have  $a = \sum_{i=0}^{n-1} \omega^i a_i$  by definition. Moreover, we will denote by  $[v_0, v_1, \dots, v_\ell]$  the matrix with row vectors  $[v_0], [v_1], \dots, [v_\ell]$ . In particular, we have  $[a_0, a_1, \dots, a_{n-1}] = I_n$  (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  $[u_0, u_1, \dots, u_{n-1}]$  is a Vandermonde matrix.

First consider an adversary that recovers  $\ell < n$  intermediate variables in the computation of the NTT, denoted  $v_1, v_2, \dots, v_\ell$ . Without loss of generality, we assume that these intermediate variables are linearly independent (otherwise the adversary equivalently gets less than  $\ell$  intermediate variables), which means that the matrix  $[v_1, v_2, \dots, v_\ell]$  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  $v_1, v_2, \dots, v_\ell$  be a set of  $\ell < n$  intermediate variables of the NTT on input a uniform  $\omega$ -encoding of a variable  $a$ . The distribution of the tuple  $(v_1, v_2, \dots, v_\ell)$  is statistically independent of  $a$  iff*

$$[a] \notin \text{Im}([v_1, \dots, v_\ell]) . \quad (12)$$

*Proof.* If  $[a] \in \text{Im}([v_1, \dots, v_\ell])$  then there exists constants  $\gamma_1, \gamma_2, \dots, \gamma_\ell$  such that  $[a] = \sum_i \gamma_i [v_i]$  implying  $a = \sum_i \gamma_i v_i$ , and the distribution  $(v_1, v_2, \dots, v_\ell)$  is hence statistically dependent on  $a$ . On the other hand, if  $[a] \notin \text{Im}([v_1, \dots, v_\ell])$ , then the system

$$\begin{cases} a = \sum_{j=0}^{n-1} \omega^j a_j & = \gamma_0 \\ v_1 = \sum_{j=0}^{n-1} \alpha_{1,j} a_j & = \gamma_1 \\ v_2 = \sum_{j=0}^{n-1} \alpha_{2,j} a_j & = \gamma_2 \\ \vdots & \\ v_\ell = \sum_{j=0}^{n-1} \alpha_{\ell,j} a_j & = \gamma_\ell \end{cases}$$

has  $|\mathbb{F}|^{n-(\ell+1)}$  solutions  $(a_0, a_1, \dots, a_{n-1})$  for every  $(\gamma_0, \gamma_1, \dots, \gamma_\ell) \in \mathbb{F}^{\ell+1}$ . This implies the statistical independence between  $a$  and  $(v_1, v_2, \dots, v_\ell)$ .  $\square$

The following lemma gives an upper bound on the probability that the above condition is not fulfilled.

**Lemma 3.** *Let  $\omega$  be a uniform random element in  $\mathbb{F}^*$  and let  $v_1, v_2, \dots, v_\ell$  be a set of  $\ell < n$  linearly independent intermediate variables of the NTT on input an  $\omega$ -encoding of a variable  $a$ . We have:*

$$\Pr [ [a] \in \text{Im}([v_1, \dots, v_\ell]) ] \leq \frac{\ell}{|\mathbb{F}| - 1} < \frac{n}{|\mathbb{F}|} , \quad (13)$$

where the above probability is taken over a uniform random choice of  $\omega$ .

*Proof.* Let us denote  $A(x) = \sum_{i=0}^{n-1} a_i x^i$  so that  $[A(\alpha)] = (1, \alpha, \alpha^2, \dots, \alpha^{n-1})$  for every  $\alpha \in \mathbb{F}$ , and in particular  $[a] = [A(\omega)]$ . For any distinct  $\ell+1$  elements  $\alpha_1, \alpha_2, \dots, \alpha_{\ell+1} \in \mathbb{F}^*$ , the matrix  $[A(\alpha_1), A(\alpha_2), \dots, A(\alpha_{\ell+1})]$  has full rank since it is a Vandermonde matrix with distinct input entries. This directly implies:

$$\underbrace{\text{Im}([A(\alpha_1), A(\alpha_2), \dots, A(\alpha_{\ell+1})])}_{\dim \ell+1} \not\subseteq \underbrace{\text{Im}([v_1, \dots, v_\ell])}_{\dim \ell} , \quad (14)$$

hence the set  $\Omega = \{\alpha \mid [A(\alpha)] \in \text{Im}([v_1, \dots, v_\ell])\}$  contains at most  $\ell$  elements. By the uniform distribution of  $\omega$ , we then have a probability at most  $\ell/(|\mathbb{F}| - 1) \leq n/|\mathbb{F}|$  to have  $\omega \in \Omega$  that is to have  $[a] \in \text{Im}([v_1, \dots, v_\ell])$ .  $\square$

We now have all the ingredients to prove Theorem 1.



*Proof. (Theorem 1)* We will show that for any adversary  $\mathcal{A}$ , the advantage  $\text{Adv}_{\mathcal{A}}^{\text{NTT}}$  in distinguishing  $\mathcal{L}(\text{NTT}, \text{Enc}(a^{(0)}))$  from  $\mathcal{L}(\text{NTT}, \text{Enc}(a^{(1)}))$  for any chosen elements  $a^{(0)}, a^{(1)} \in \mathbb{F}$  is lower than  $\varepsilon$ , where  $\mathcal{L}(\text{NTT}, \text{Enc}(a))$  denotes the  $\delta$ -random-probing leakage of the procedure  $\text{NTT}_{\xi}$  on input  $\text{Enc}(a) = \langle \omega, (a_i)_{i=0}^{n-1} \rangle$ . Note that this leakage is a tuple in which each coordinate corresponds to an arithmetic instruction in the computation of  $\text{NTT}_{\xi}$  that either equals  $\perp$  (with probability  $1 - \delta$ ) or the input of the instruction. We recall that the advantage is defined as  $\text{Adv}_{\mathcal{A}}^{\text{NTT}} = |\text{Succ}_{\mathcal{A}}^{\text{NTT}} - \frac{1}{2}|$  where

$$\text{Succ}_{\mathcal{A}}^{\text{NTT}} = \Pr \left( \begin{array}{l} (a^{(0)}, a^{(1)}, \mu) \leftarrow \mathcal{A}(\perp) \\ b \leftarrow \{0, 1\} \\ \ell \leftarrow \mathcal{L}(\text{NTT}, a^{(b)}) \end{array} : \mathcal{A}(a^{(0)}, a^{(1)}, \mu, \ell) = b \right) \quad (15)$$

Without loss of generality, we assume  $\text{Succ}_{\mathcal{A}}^{\text{NTT}} \geq \frac{1}{2}$ . Indeed, for any adversary with success probability  $\frac{1}{2} - \text{Adv}_{\mathcal{A}}^{\text{NTT}}$ , there exists an adversary  $\mathcal{A}'$  with success probability  $\frac{1}{2} + \text{Adv}_{\mathcal{A}}^{\text{NTT}}$  (defined as  $\mathcal{A}'(a^{(0)}, a^{(1)}, \ell) = 1 - \mathcal{A}(a^{(0)}, a^{(1)}, \mu, \ell)$ ).

The procedure  $\text{NTT}_{\xi}$  is composed of  $N = 3n \log n$  arithmetic instructions. In the  $\delta$ -random-probing model, each of these instructions leaks its input(s) with probability  $\delta$ . The number of instructions that leak hence follows a binomial distribution with parameters  $N$  and  $\delta$ . Let us denote by  $\text{max}_{\ell}$  the event that  $\ell$  or less instructions leak in the random-probing leakage  $\mathcal{L}(\text{NTT}, \text{Enc}(a))$ . Since each instruction takes at most two inputs over  $\mathbb{F}$ , the adversary gets the values of at most  $2\ell$  intermediate variables whenever  $\text{max}_{\ell}$  occurs. By the Chernoff bound (see Corollary 1), the probability that more than  $\ell > N\delta$  arithmetic instructions leak, namely the probability that  $\neg \text{max}_{\ell}$  occurs, satisfies:

$$\Pr(\neg \text{max}_{\ell}) \leq \psi(\ell, N) . \quad (16)$$

From  $N = 3n \log n$  and  $\ell < \frac{n}{2}$ , we get that:

$$\Pr(\neg \text{max}_{\ell}) \leq \exp \left( - \frac{(1 - 6\delta \log n)^2}{2 + 12\delta \log n} n \right) \leq \exp \left( - \frac{(1 - 6\delta \log n)^2}{4} n \right) . \quad (17)$$

Now let assume that  $\text{max}_{\ell}$  occurs for some  $\ell < \frac{n}{2}$  and let denote  $v_1, v_2, \dots, v_{2\ell}$  the recovered intermediate variables. Without loss of generality, we assume that the recovered intermediate variables are linearly independent. Let us then denote by  $\text{free}$  the event that  $[a] \notin \text{Im}([v_1, \dots, v_{2\ell}])$ . By Lemma 3, we have

$$\Pr(\neg \text{free}) < \frac{n}{|\mathbb{F}|} . \quad (18)$$

And let finally denote by  $\text{succ}$  the event that  $\mathcal{A}$  outputs the right bit  $b$  on input  $(a^{(0)}, a^{(1)}, \mu, \ell)$  so that  $\text{Succ}_{\mathcal{A}}^{\text{NTT}} = \Pr(\text{succ})$ . We can then write:

$$\begin{aligned} \text{Succ}_{\mathcal{A}}^{\text{NTT}} &= \Pr(\text{max}_{\ell}) \Pr(\text{succ} \mid \text{max}_{\ell}) + \Pr(\neg \text{max}_{\ell}) \Pr(\text{succ} \mid \neg \text{max}_{\ell}) \\ &\leq \Pr(\text{succ} \mid \text{max}_{\ell}) + \Pr(\neg \text{max}_{\ell}) . \end{aligned} \quad (19)$$

In the same way, we have

$$\Pr(\text{succ} \mid \text{max}_{\ell}) \leq \Pr(\text{succ} \mid \text{max}_{\ell} \cap \text{free}) + \Pr(\neg \text{free}) . \quad (20)$$

By Lemma 2, we have that the leakage  $\ell$  is statistically independent of  $a^{(b)}$  in (15) whenever  $\text{max}_{\ell} \cap \text{free}$  occurs. This directly implies  $\Pr(\text{succ} \mid \text{max}_{\ell} \cap \text{free}) = \frac{1}{2}$ , which gives

$$\text{Succ}_{\mathcal{A}}^{\text{NTT}} < \frac{1}{2} + \Pr(\neg \text{max}_{\ell}) + \Pr(\neg \text{free}) . \quad (21)$$

Hence, we finally get

$$\text{Adv}_{\mathcal{A}}^{\text{NTT}} < \Pr(\neg \text{max}_{\ell}) + \Pr(\neg \text{free}) = \frac{n}{|\mathbb{F}|} + \exp \left( - \frac{(1 - 6\delta \log n)^2}{4} n \right) , \quad (22)$$

which concludes the proof.  $\square$

**Security of the Full Multiplication.** We now prove the security of the full multiplication. We have the following result:

**Theorem 2.** *Let  $\omega$  be a uniform random element of  $\mathbb{F}^*$ , let  $(a_i)_{i=0}^{n-1}$  and  $(b_i)_{i=0}^{n-1}$  be uniform  $\omega$ -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  $\varepsilon$ -leakage secure in the  $\delta$ -random-probing leakage model, where*

$$\varepsilon = \frac{2n}{|\mathbb{F}|} + 5 \exp\left(-\frac{(1-21\delta \log n)^2}{14} n\right). \quad (23)$$

*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  $\ell_1, \ell_2, \dots, \ell_5$  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  $\ell_3$  pairs  $(u_i, r_i)$ ,<sup>7</sup> 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  $\ell_5$  pairs  $(t_i, t_{i+n})$ ,<sup>8</sup> 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

$$\sum_{i=1}^n \alpha_{k,i} \cdot a_i = \eta_k \quad \text{for } k = 1, \dots, \ell_1^* \quad (24)$$

- up to  $\ell_2^* = 2\ell_2 + \ell_3$  linear equations of the form

$$\sum_{i=1}^n \beta_{k,i} \cdot b_i = \nu_k \quad \text{for } k = 1, \dots, \ell_2^* \quad (25)$$

- up to  $\ell_3^* = 2\ell_4 + 2\ell_5$  linear equations of the form

$$\sum_{j=1}^{2n} \gamma_{k,j} \cdot s_j = \chi_k \quad \text{for } k = 1, \dots, \ell_3^* \quad (26)$$

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  $\ell_3^*$  equations can be rewritten as:

$$\sum_{i=1}^n \gamma'_{k,i} \cdot b_i = \chi_k \quad \text{for } k = 1, \dots, \ell_3^* \quad (27)$$

where the  $\gamma'_{k,i}$ 's are coefficients that depend on the  $a_i$ 's.

From these equations, the attacker gains the knowledge that:

<sup>7</sup> 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.

<sup>8</sup> 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

$$\mathcal{S}_1 = \{\mathbf{x} \in \mathbb{F}^n ; M_1 \cdot \mathbf{x} = \boldsymbol{\eta}\} \quad (28)$$

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  $\eta_k$ ,

2. the encoding  $(b_i)_{i=0}^{n-1}$  then belongs to some vectorial space

$$\mathcal{S}_2 = \{\mathbf{x} \in \mathbb{F}^n ; M_2 \cdot \mathbf{x} = (\boldsymbol{\nu}, \boldsymbol{\chi})\} \quad (29)$$

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  $\nu_k$  and  $\chi_k$ .

Following the demonstration of Lemma 2, it can be checked that if

$$(1, \omega, \dots, \omega^{2n-1}) \notin \text{Im}(M_1) \quad \text{and} \quad (1, \omega, \dots, \omega^{2n-1}) \notin \text{Im}(M_2) ,$$

then the full leakage of the multiplication is statistically independent of  $a$  and  $b$ , namely the leakage security holds. These two events are denoted  $\text{free}_1$  and  $\text{free}_2$  hereafter.

Then, following the demonstration of Lemma 3,  $\text{free}_1$  occurs with probability at least  $1 - \frac{n}{|\mathbb{F}|}$  over a random choice of  $\omega$ , provided that we have  $\text{rank}(M_1) < n$ . Then, since the vectorial space  $\mathcal{S}_1$  is independent of  $\omega$ , 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  $\omega$  and we have that  $\text{free}_2$  occurs with probability at least  $1 - \frac{n}{|\mathbb{F}|}$  over a random choice of  $\omega$  as long as we have  $\text{rank}(M_2) < n$ . The two conditions on the ranks of  $M_1$  and  $M_2$  are then fulfilled whenever we have

$$\ell_1^* = 2\ell_1 + \ell_3 < n , \quad (30)$$

and

$$\ell_2^* + \ell_3^* = 2\ell_2 + \ell_3 + 2\ell_4 + 2\ell_5 < n . \quad (31)$$

Let us denote  $\text{max}_i$  the event that the number of leaking operations  $\ell_i$  at step  $i$  is lower than  $n/7$ , for every  $i$ . If  $\text{max}_i$  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:

$$\Pr(\neg \text{max}_i) \leq \psi\left(\frac{n}{7}, N_i\right) , \quad (32)$$

where  $N_i$  is the number of operations at step  $i$ , which satisfies  $N_i \leq 3n \log n$ , which gives

$$\Pr(\neg \text{max}_i) \leq \psi\left(\frac{n}{7}, 3n \log n\right) \leq \exp\left(-\frac{(1 - 21\delta \log n)^2}{14} n\right) . \quad (33)$$

We finally get that the multiplication is  $\varepsilon$ -leakage secure with

$$\begin{aligned} \varepsilon &< \Pr(\neg \text{max}_1) + \Pr(\neg \text{max}_2) + \dots + \Pr(\neg \text{max}_5) \\ &+ \underbrace{\Pr(\neg \text{free}_1 \mid \text{max}_1 \wedge \dots \wedge \text{max}_5)}_{< n/|\mathbb{F}|} + \underbrace{\Pr(\neg \text{free}_2 \mid \text{max}_1 \wedge \dots \wedge \text{max}_5)}_{< n/|\mathbb{F}|} . \end{aligned} \quad (34)$$

□

## 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  $\omega$ -encoding of a variable  $a$ . We refresh it into an  $\omega$ -encoding  $(a'_i)_{i=0}^{n-1}$  of  $a$  as follows:

1. sample a random  $\omega$ -encoding  $(r_0, r_1, \dots, r_{n-1}) \leftarrow \text{Enc}_\omega(0)$
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  $\omega$ -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  $\text{Enc}_\omega(0)$  sampler which satisfies these two properties.

**Sampling Encodings of 0.** At the beginning of the computation of  $\Pi$ , a random  $\omega$ -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  $\omega$ , this encoding can be fully leaked to the adversary. Our sampler then works as follows:

1. pick  $n - 1$  random values  $u_0, u_1, \dots, u_{n-2}$  over  $\mathbb{F}$ ,
2. output  $(r_i)_{i=0}^{n-1} = \text{NTTMult}((u_0, u_1, \dots, u_{n-2}, 0), (e_0, e_1, \dots, e_{n-1}))$

where  $\text{NTTMult}$  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:

$$(u_0, u_1, \dots, u_{n-2}) \mapsto \text{NTTMult}((u_0, u_1, \dots, u_{n-2}, 0), (e_0, e_1, \dots, e_{n-1})), \quad (35)$$

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.<sup>9</sup> By discarding one column we can get a full-rank square matrix of dimension  $n - 1$ , allowing the recovery of the  $(u_0, u_1, \dots, u_{n-2})$  from output encoding. Therefore, we have a one-to-one mapping between the vectors  $(u_0, u_1, \dots, u_{n-2}) \in \mathbb{F}^{n-1}$  and the  $\omega$ -encodings of 0,  $(r_i)_{i=0}^{n-1} \in \mathbb{F}^n$  with  $\text{Dec}_\omega((r_i)_{i=0}^{n-1}) = 0$ .

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  $\Pi$ . Each arithmetic instruction of  $P$  gives rise to a corresponding *gadget* in  $\Pi$  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  $\omega$ -encoding of  $a$ . The copy gadget compute an  $\omega$ -encoding  $(a'_i)_{i=0}^{n-1}$  of  $a$  as:

$$(a'_0, a'_1, \dots, a'_{n-1}) \leftarrow \text{refresh}(a_0, a_1, \dots, a_{n-1})$$

<sup>9</sup> 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  $\omega$ -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):

$$(a_i)_{i=0}^{n-1} \rightarrow (a_i^{(2)})_{i=0}^{n-1} \rightarrow \dots \rightarrow (a_i^{(N)})_{i=0}^{n-1}$$

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  $\omega$ -encoding of  $a$  and  $(b_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $b$ . To compute an  $\omega$ -encoding  $(c_i)_{i=0}^{n-1}$  of  $c = a + b$ , we simply compute:

$$(c_0, c_1, \dots, c_{n-1}) \leftarrow \text{refresh}(a_0 + b_0, a_1 + b_1, \dots, a_{n-1} + b_{n-1})$$

**Subtraction gadget.** Let  $(a_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $a$  and  $(b_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $b$ . To compute an  $\omega$ -encoding  $(c_i)_{i=0}^{n-1}$  of  $c = a - b$ , we simply compute:

$$(c_0, c_1, \dots, c_{n-1}) \leftarrow \text{refresh}(a_0 - b_0, a_1 - b_1, \dots, a_{n-1} - b_{n-1})$$

**Multiplication gadget.** Let  $(a_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $a$  and  $(b_i)_{i=0}^{n-1}$  be an  $\omega$ -encoding of  $b$ . To compute an  $\omega$ -encoding  $(c_i)_{i=0}^{n-1}$  of  $c = a \cdot b$ , we simply compute:

$$(c_0, c_1, \dots, c_{n-1}) \leftarrow \text{refresh}(\text{NTTMult}((a_0, a_1, \dots, a_{n-1}), (b_0, b_1, \dots, b_{n-1})))$$

where  $\text{NTTMult}$  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  $\omega$ -encoding  $(a'_i)_{i=0}^{n-1}$  in output of the refreshing procedure is uniformly distributed and independent of the input  $\omega$ -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  $\text{Enc}_\omega(0)$  sampler. The I/O linear separability holds from the output linearity of the  $\text{Enc}_\omega(0)$  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  $\sum_i \gamma_i a_i$ .

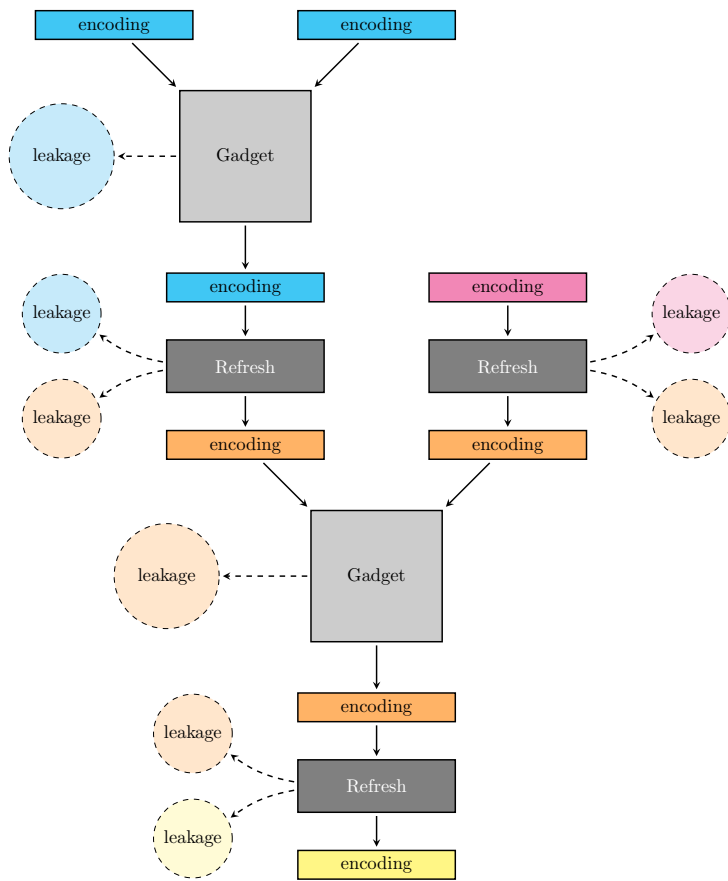
The I/O linear separability of the refreshing procedure implies that its leakage can be split into some leakage depending only on its input encoding, which is the output (before refreshing) from a previous gadget, and some leakage depending only on its output encoding, which is the input of a next gadget. 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  $\mathbf{x} \in \mathbb{F}^s$  and let  $\Pi$  denotes the corresponding program protected with  $n$ -size encodings as described above. For every  $\delta < 1/(33 \log n)$ ,  $\Pi$  is  $\varepsilon$ -leakage secure in the  $\delta$ -random-probing model where*

$$\varepsilon = 3|P| \cdot \left( 2 \exp \left( - \frac{(1 - 33\delta \log n)^2}{22} n \right) + \frac{2n}{|\mathbb{F}|} \right), \quad (36)$$

where  $|P|$  denotes the size of  $P$  i.e. its number of arithmetic instructions.



**Fig. 1.** Illustration of our compositional security.

*Proof.* Let  $|II|$  denotes the number of gadgets in  $II$ . Since the output of each gadget is refreshed (and nothing more), the number of call to the refreshing procedure is also  $|II|$ . Each arithmetic instruction in  $P$  gives rise to one associated gadget, plus up to 2 copy gadgets if necessary. We hence deduce  $|II| \leq 3|P|$ .

Let us denote by  $\text{rmax}$  the event that at most  $\frac{n}{11}$  operations leak in each refreshing. By applying the Chernoff bound (see Corollary 1), we have

$$\Pr(\neg\text{rmax}) \leq |II| \cdot \psi\left(\frac{n}{11}, N_{\text{ref}}\right), \quad (37)$$

where  $N_{\text{ref}}$  denotes the number of elementary operations in the refreshing procedure. Let us further denote by  $\text{gmax}$  the event that at most  $\frac{n}{11}$  operations leak in each gadget (without refreshing). In the same way as above, we have

$$\Pr(\neg\text{gmax}) \leq \sum_{i=1}^{|II|} \psi\left(\frac{n}{11}, N^{(i)}\right) \leq |II| \cdot \psi\left(\frac{n}{11}, N_{\text{gad}}\right), \quad (38)$$

where  $N^{(i)}$  denotes the number of elementary operations in the  $i$ th gadget and where  $N_{\text{gad}}$  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  $i$ th gadget of  $II$  and by  $(c_j^{(i)})_j$  the output encoding (before refreshing) of the  $i$ th gadget of  $II$ . Let us further denote by  $\mathcal{L}$  the full  $\delta$ -random-probing leakage of  $II$ , so that we have:

$$\mathcal{L} = \bigcup_{i=1}^{|II|} \mathcal{G}^{(i)} \cup \bigcup_{i=1}^{|II|} \mathcal{R}^{(i)} \quad (39)$$

where  $\mathcal{G}^{(i)}$  denotes the leakage from the  $i$ th gadget (without refreshing) and where  $\mathcal{R}^{(i)}$  denotes the leakage of the  $i$ th refresh. Specifically,  $\mathcal{G}^{(i)}$  and  $\mathcal{R}^{(i)}$  are families of intermediate variables (inputs of elementary operations) that are revealed by the  $\delta$ -random-probing leakage. If  $\text{rmax}$  and  $\text{gmax}$  occurs, we have  $|\mathcal{G}^{(i)}| \leq \frac{2n}{11}$  and  $|\mathcal{R}^{(i)}| \leq \frac{2n}{11}$ .

According to the I/O linear separability property of the refreshing procedure, we can define a *separated leakage*  $\mathcal{L}'$  as

$$\mathcal{L}' = \bigcup_{i=1}^{|II|} (\mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}) \quad (40)$$

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  $\mathbf{x}$ , 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  $\omega$ ) that the family  $\mathcal{L}'$  is statistically dependent on  $\mathbf{x}$ , hereafter denoted  $\mathbf{x} \nabla \mathcal{L}'$ . We have

$$\mathbf{x} \nabla \mathcal{L}' = \bigvee_{i=1}^{|II|} (\mathbf{x} \nabla \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}). \quad (41)$$

By the uniformity property of the refreshing, we have that, given the program input  $\mathbf{x}$ , the different families of input/output shares  $\{(a_j^{(i)})_{j=0}^{n-1}, (b_j^{(i)})_{j=0}^{n-1}, (c_j^{(i)})_{j=0}^{n-1}\}$  are mutually independent. We hence get

$$\Pr(\mathbf{x} \nabla \mathcal{L}') \leq \sum_{i=1}^{|II|} \Pr(\mathbf{x} \nabla \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)}). \quad (42)$$

We can then upper bound the probability  $\Pr(\mathbf{x} \nabla \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)})$  when the  $i$ th 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  $1 \leq k \leq \ell_1^*$ ;
- 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  $1 \leq k \leq \ell_2^*$ ;
- 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  $1 \leq k \leq \ell_3^*$ ;

where  $\ell'_1$ ,  $\ell'_2$  and  $\ell'_3$ , 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  $\text{rmax}$  and  $\text{gmax}$  occurs, we get

$$\Pr(\mathbf{x} \nabla \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)} \mid \text{rmax} \wedge \text{gmax}) \leq \frac{2n}{|\mathbb{F}|}. \quad (43)$$

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 than  $\frac{n}{11}$  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

$$\begin{aligned} \Pr(\mathbf{x} \nabla \mathcal{L}') &\leq \Pr(\neg \text{rmax}) + \Pr(\neg \text{gmax}) + \sum_{i=1}^{|II|} \Pr(\mathbf{x} \nabla \mathcal{G}^{(i)} \cup \mathcal{A}^{(i)} \cup \mathcal{B}^{(i)} \cup \mathcal{C}^{(i)} \mid \text{rmax} \wedge \text{gmax}) \\ &\leq |II| \cdot \left( \psi\left(\frac{n}{11}, N_{\text{ref}}\right) + \psi\left(\frac{n}{11}, N_{\text{gad}}\right) + \frac{2n}{|\mathbb{F}|} \right), \end{aligned}$$

which together with  $N_{\text{ref}}, N_{\text{gad}} < 3n \log n$  concludes the proof.  $\square$

## 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  $\varepsilon$ -noisy leakage model, a logical program leaks an  $\varepsilon$ -noisy leakage function  $f(\mathbf{m}_j, \mathbf{m}_k)$  of the pair of inputs of each logical instruction  $\mathbf{m}_i \leftarrow \mathbf{m}_j * \mathbf{m}_k$ .

The security reduction of Duc *et al.* (Lemma 1) then implies that a logical program  $II$  that is secure against  $\delta$ -random-probing leakage is also secure against  $\delta'$ -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  $II$  be a  $\varepsilon$ -leakage secure arithmetic program in the  $\delta$ -random-probing model, then there exists a functionally equivalent logical program  $II'$  that is  $\varepsilon$ -leakage secure in the  $\delta'$ -random-probing model for some  $\delta'$  satisfying*

$$\delta' = 1 - (1 - \delta)^{1/N} \geq \frac{\delta}{N} \quad \text{with} \quad N = O\left(\frac{1}{w} \log |\mathbb{F}| \log\left(\frac{1}{w} \log |\mathbb{F}|\right)\right). \quad (44)$$

*Proof.* The logical program  $II'$  is simply the program  $II$  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  $N = O\left(\frac{1}{w} \log |\mathbb{F}|\right)$  elementary ( $w$ -bit) operations, and that the multiplication on  $\mathbb{F}$  can be computed from  $N = O\left(\frac{1}{w} \log |\mathbb{F}| \log\left(\frac{1}{w} \log |\mathbb{F}|\right)\right)$  elementary ( $w$ -bit) operations.



Assume that there exists an adversary  $\mathcal{A}'$  with advantage  $\varepsilon$  that makes use of a  $\delta'$ -random-probing leakage on  $\Pi'$ , then we show that there exists an adversary  $\mathcal{A}$  with advantage  $\varepsilon$  that makes use of a  $\delta$ -random-probing leakage on  $\Pi$ . Since by assumption no such adversary  $\mathcal{A}$  exists, then by contraposition neither does such adversary  $\mathcal{A}'$ , meaning that  $\Pi'$  is indeed  $\varepsilon$ -leakage secure in the  $\delta'$ -random-probing model.

We construct an adversary  $\mathcal{A}$  that is given the full input to an arithmetic instruction of  $\Pi$  whenever at least one of the corresponding logical instruction leaks in  $\Pi'$ . 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  $\delta'$ -random-probing leakage on  $\Pi'$  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  $\Pi'$  must leak), it can compute all the inputs of the corresponding logical instructions in  $\Pi'$ , 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  $\delta'$ . 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  $\delta$ -random-probing leakage on  $\Pi$  where

$$\delta = 1 - (1 - \delta')^N,$$

for  $N = O(\log |\mathbb{F}| \log \log |\mathbb{F}|)$ . Since by assumption no such adversary exists, this means that no adversary  $\mathcal{A}'$  exists with advantage  $\varepsilon$  that makes use of a  $\delta'$ -random-probing leakage on  $\Pi'$ .  $\square$

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  $\Pi$  be a  $\varepsilon$ -leakage secure arithmetic program in the  $\delta$ -random-probing model, then there exists a functionally equivalent logical program  $\Pi'$  that is  $\varepsilon$ -leakage secure in the  $\delta'$ -noisy leakage model for some  $\delta'$  satisfying*

$$\delta' = \frac{\delta}{O(\log |\mathbb{F}| \log \log |\mathbb{F}|)}. \quad (45)$$

### 5.3 Application to Our Scheme

In the previous section we have shown that for  $\delta = O(1/\log n)$  our construction is  $\varepsilon$ -leakage secure in the  $\delta$ -random-probing model with

$$\varepsilon = \text{negl}(\lambda) + \text{negl}'(n) \quad (46)$$

where  $\text{negl}$  and  $\text{negl}'$  are some negligible functions and where  $\lambda$  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  $\Pi'$  denotes the secure logical program corresponding to our construction (see Section 4).  $\Pi'$  is  $\varepsilon$ -leakage secure in the  $\delta$ -noisy leakage model where  $\varepsilon = \text{negl}(\lambda) + \text{negl}'(n)$  and  $\delta' = O(1/((\log n)^2 \log \log n))$ .*

## 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:

$$a \oplus b = a + b - 2ab, \quad (47)$$

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  $\alpha$  being large enough so that  $n/\alpha$  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  $\omega$ . However this might be a proof artefact and the scheme could be secure for some constant  $\omega$  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<sup>+</sup>15] since our encoding is a kind of randomized Shamir’s secret sharing.

## References

- [ADD<sup>+</sup>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<sup>+</sup>16] Sonia Belaïd, Fabrice Benhamouda, Alain Passetègue, 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<sup>+</sup>17] Sonia Belaïd, Fabrice Benhamouda, Alain Passetègue, 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<sup>+</sup>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  $A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{d-1}x^{d-1}$ . For a given primitive  $d$ -th root of unity  $\xi$ , the NTT maps the coefficients of  $A$  to the evaluations  $A(\xi^i)$  with  $1 \leq i \leq d$ :

$$\text{NTT}_\xi : (a_0, a_1, \dots, a_{d-1}) \mapsto (A(\xi^1), A(\xi^2), \dots, A(\xi^d)) . \quad (48)$$

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

$$\begin{aligned} A_0(x) &= a_0 + a_2x + a_4x^2 + \dots + a_{d-2}x^{\frac{d}{2}-1} \\ A_1(x) &= a_1 + a_3x + a_5x^2 + \dots + a_{d-1}x^{\frac{d}{2}-1} \end{aligned}$$

which satisfy

$$A(x) = A_0(x^2) + xA_1(x^2).$$

The problem of evaluating  $A(x)$  at each  $d$ -th root of unity  $\xi^i$ , for  $1 \leq i \leq d$ , is reduced to the problem of evaluating  $A_0(x)$  and  $A_1(x)$  at the points  $\xi^{2i}$ , 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  $\xi^{2i}$  with the same divide and conquer strategy, using the polynomials  $A_{00}, A_{01}, A_{10}, A_{11}$  satisfying

$$A_0(x) = A_{00}(x^2) + xA_{01}(x^2) \quad \text{and} \quad A_1(x) = A_{01}(x^2) + xA_{11}(x^2) .$$

This divide and conquer strategy can be iterated  $\log_2(d)$  times. At the  $t$ -th step we have  $2^t$  polynomials  $A_{\mathbf{u}}$  of degree  $\frac{d}{2^t}$  for  $\mathbf{u} \in \{0, 1\}^t$  that must be evaluated in  $\xi^j$  for  $j = 2^t, 2 \cdot 2^t, \dots, \frac{d}{2^t} \cdot 2^t$ , 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

$$A_{\mathbf{u}}(\xi^j) = A_{\mathbf{u}|0}(\xi^{2j}) + \xi^j A_{\mathbf{u}|1}(\xi^{2j}) \quad \text{and} \quad A_{\mathbf{u}}(\xi^{j+\frac{d}{2}}) = A_{\mathbf{u}|0}(\xi^{2j}) - \xi^j A_{\mathbf{u}|1}(\xi^{2j}) \quad (49)$$

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  $A_{\mathbf{u}} = a_{\varphi(\mathbf{u})}$  with  $\varphi(\mathbf{u})$  denoting the integer corresponding to the binary expansion  $\mathbf{u} \in \{0, 1\}^{\log_2(d)}$ . 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.  $(c_0, c_1, \dots, c_{d-1}) \leftarrow (a_0, a_1, \dots, a_{d-1})$
2. for  $t = \log_2(d) - 1$  down to 1:
3.      $j = 2^t; k = 2^{d-t-1}$
4.     for  $i \in \bigcup_{\ell=0}^{k-1} U_{j,\ell}$
5.          $(c_i, c_{i+j}) \leftarrow (c_i + \xi^j c_{i+j}, c_i - \xi^j c_{i+j})$

where  $U_{j,\ell} = \{(2\ell j, \dots, (2\ell + 1)j - 1)\}$  and where the index shiftings of  $c$  are done modulo  $d$ , *i.e.*  $c_{i+j} = c_{i+j \bmod d}$ . It can be checked that the above evaluation of  $\text{NTT}_\xi$  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  $d$ th 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:

$$\begin{aligned} (A(\xi^1), A(\xi^2), \dots, A(\xi^d)) &= \text{NTT}_\xi(a_0, a_1, \dots, a_{n-1}, 0, \dots, 0) \\ (B(\xi^1), B(\xi^2), \dots, B(\xi^d)) &= \text{NTT}_\xi(b_0, b_1, \dots, b_{n-1}, 0, \dots, 0) \end{aligned}$$

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  $(C(\xi), C(\xi^1), \dots, C(\xi^d))$ , which satisfies

$$(c_0, c_1, \dots, c_d) = \text{NTT}_\xi^{-1}(C(\xi^1), C(\xi^2), \dots, C(\xi^d)) = \text{NTT}_{\xi^{-1}}\left(\frac{1}{d}C(\xi^1), \frac{1}{d}C(\xi^2), \dots, \frac{1}{d}C(\xi^d)\right).$$