

# Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme<sup>\*</sup>

Alberto Battistello<sup>1</sup>, Jean-Sébastien Coron<sup>2</sup>, Emmanuel Prouff<sup>3\*\*</sup>, and Rina Zeitoun<sup>1</sup>

<sup>1</sup> Oberthur Technologies, France  
{a.battistello,r.zeitoun}@oberthur.com

<sup>2</sup> University of Luxembourg  
jean-sebastien.coron@uni.lu

<sup>3</sup> Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA,  
Laboratoire d'Informatique de Paris 6 (LIP6), Équipe PolSys, 4 place Jussieu, 75252 Paris Cedex 05, France

July 27, 2016

**Abstract.** A common countermeasure against side-channel attacks consists in using the masking scheme originally introduced by Ishai, Sahai and Wagner (ISW) at Crypto 2003, and further generalized by Rivain and Prouff at CHES 2010. The countermeasure is provably secure in the probing model, and it was showed by Duc, Dziembowski and Faust at Eurocrypt 2014 that the proof can be extended to the more realistic noisy leakage model. However the extension only applies if the leakage noise  $\sigma$  increases at least linearly with the masking order  $n$ , which is not necessarily possible in practice.

In this paper we investigate the security of an implementation when the previous condition is not satisfied, for example when the masking order  $n$  increases for a constant noise  $\sigma$ . We exhibit two (template) horizontal side-channel attacks against the Rivain-Prouff's secure multiplication scheme and we analyze their efficiency thanks to several simulations and experiments.

We also describe a variant of Rivain-Prouff's multiplication that is still provably secure in the original ISW model, and also heuristically secure against our new attacks. Finally, we describe a new mask refreshing algorithm with complexity  $\mathcal{O}(n \log n)$ , instead of  $\mathcal{O}(n^2)$  for the classical algorithm.

## 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. To secure implementations against this threat, security developers usually apply techniques inspired from *secret sharing* [Bla79, Sha79] or *multi-party computation* [CCD88]. The idea is to randomly split a secret into several shares such that the adversary needs all of them to reconstruct the secret. For these schemes, the number of shares  $n$  in which the key-dependent data are split plays the role of a security parameter.

A common countermeasure against side-channel attacks consists in using the masking scheme originally introduced by Ishai, Sahai and Wagner (ISW) [ISW03]. The countermeasure achieves provable security in the so-called *probing security model* [ISW03], in which the adversary can recover a limited number of intermediate variables of the computation. This model has been argued to be practically relevant to address so-called *higher-order* side-channel attacks and it has been the basis of several efficient schemes to protect block ciphers [BFG15, CGP<sup>+</sup>12, Cor14, CRV15, GPQ11, GPS14, PR11, RP10].

More recently, it has been shown in [DDF14] that the probing security of an implementation actually implies its security in the more realistic *noisy leakage model* introduced in [PR13]. More precisely, if an implementation obtained by applying the compiler in [ISW03]

---

\* An extended abstract will appear at CHES 2016; this is the full version.

\*\* Part of this work has been done at Safran Identity and Security, and while the author was at ANSSI, France.

is secure at order  $n$  in the probing model, then [DFS15, Theorem 3] shows that the success probability of distinguishing the correct key among  $|\mathbb{K}|$  candidates is bounded above by  $|\mathbb{K}| \cdot 2^{-n/9}$  if the leakage  $L_i$  on each intermediate variable  $X_i$  satisfies:

$$I(X_i; L_i) \leq 2 \cdot (|\mathbb{K}| \cdot (28n + 16))^{-2} ,$$

where  $I(\cdot; \cdot)$  denotes the *mutual information* and where the index  $i$  ranges from 1 to the total number of intermediate variables.

In this paper we investigate what happens when the above condition is not satisfied. Since the above mutual information  $I(X_i; L_i)$  can be approximated by  $k/(8\sigma^2)$  in the Hamming weight model in  $\mathbb{F}_{2^k}$ , where  $\sigma$  is the noise in the measurement (see Appendix A), this amounts to investigating the security of Ishai-Sahai-Wagner’s (ISW) implementations when the number of shares  $n$  satisfies:

$$n > c \cdot \sigma$$

As already observed in previous works [VGS14, CFG<sup>+</sup>10], the fact that the same share (or more generally several data depending on the same sensitive value) is manipulated several times may open the door to new attacks which are not taken into account in the probing model. Those attacks, sometimes called *horizontal* [CFG<sup>+</sup>10] or (*Template*) *algebraic* [ORSW12, VGS14] exploit the algebraic dependency between several intermediate results to discriminate key hypotheses.

In this paper, we exhibit two (horizontal) side channel attacks against the ISW multiplication algorithm. These attacks show that the use of this algorithm (and its extension proposed by Rivain and Prouff in [RP10]) may introduce a weakness with respect to horizontal side channel attacks if the sharing order  $n$  is such that  $n > c \cdot \sigma^2$ , where  $\sigma$  is the measurement noise. While the first attack is too costly (even for low noise contexts) to make it applicable in practice, the second attack, which essentially iterates the first one until achieving a satisfying likelihood, shows very good performances. For instance, when the leakages are simulated by noisy Hamming weights computed over  $\mathbb{F}_{2^8}$  with  $\sigma = 1$ , it recovers all the shares of a 21-sharing. We also confirm the practicality of our attack with a real life experiment on a development platform embedding the **ATMega328** processor (see Section 7). Actually, in this context where the leakages are multivariate and not univariate as in our theoretical analyses and simulations, the attack appears to be more efficient than expected and recovers all the shares of a  $n$ -sharing when  $n \geq 40$ .

We also describe a variant of Rivain-Prouff’s multiplication that is still provably secure in the original ISW model, and also heuristically secure against our new attacks. Our new countermeasure is similar to the countermeasure in [FRR<sup>+</sup>10], in that it can be divided in two steps: a “matrix” step in which starting from the input shares  $x_i$  and  $y_j$ , one obtains a matrix  $x_i \cdot y_j$  with  $n^2$  elements, and a “compression” step in which one uses some randomness to get back to a  $n$ -sharing  $c_i$ . Assuming a leak-free component, the countermeasure in [FRR<sup>+</sup>10] is proven secure in the noisy leakage model, in which the leakage function reveals all the bits of the internal state of the circuit, perturbed by independent binomial noise. Our countermeasure does not use any leak-free component, but is only heuristically secure in the noisy leakage model (see Section 8.2 for our security analysis).

Eventually, we describe in Section 9 a new mask refreshing algorithm with complexity  $\mathcal{O}(n \cdot \log n)$  instead of  $\mathcal{O}(n^2)$  for the classical algorithm in [BBD<sup>+</sup>15]. A completely different mask refreshing algorithm is described in [ADF16], with complexity only  $\mathcal{O}(n)$ ; however our new algorithm is significantly simpler. Our new mask refreshing algorithm enables to decrease the complexity of the previous variant of Rivain-Prouff’s multiplication from  $\mathcal{O}(n^2 \log n)$  to  $\mathcal{O}(n^2)$ , hence the same complexity as the original Rivain-Prouff multiplication.

## 2 Preliminaries

For two positive integers  $n$  and  $d$ , a  $(n, d)$ -*sharing* of a variable  $x$  defined over some finite field  $\mathbb{F}_{2^k}$  is a random vector  $(x_1, x_2, \dots, x_n)$  over  $\mathbb{F}_{2^k}$  such that  $x = \sum_{i=1}^n x_i$  holds (*completeness equality*) and any tuple of  $d - 1$  shares  $x_i$  is a uniform random vector over  $(\mathbb{F}_{2^k})^{d-1}$ . If  $n = d$ , the terminology simplifies to  $n$ -*sharing*. An algorithm with domain  $(\mathbb{F}_{2^k})^n$  is said to be  $(n - 1)$ th-order secure in the probing model if on input an  $n$ -sharing  $(x_1, x_2, \dots, x_n)$  of some variable  $x$ , it admits no tuple of  $n - 1$  or fewer intermediate variables that depends on  $x$ .

Calligraphic letters, like  $\mathcal{X}$ , are used to denote finite sets (*e.g.*  $\mathbb{F}_{2^n}$ ). The corresponding large letter  $X$  denotes a random variable over  $\mathcal{X}$ , while the lower-case letter  $x$  a value over  $\mathcal{X}$ . The probability of an event  $ev$  is denoted by  $\Pr[ev]$ . The *probability distribution function* (pdf for short) of a *continuous* random variable  $X$  is denoted by  $f_X(\cdot)$ . It will sometimes be denoted by  $p_X(\cdot)$  if  $X$  is discrete. The pdf of the random variable  $X|Y$  is denoted by  $f_{X|Y}(\cdot)$ . The expectation and the variance of a random variable  $X$  are respectively denoted by  $\mathbb{E}[X]$  and  $\text{Var}[X]$ . The covariance between two random variables  $X$  and  $Y$  is denoted by  $\text{Cov}[X, Y]$ . The *Signal to Noise Ratio* (SNR) of a univariate noisy observation  $L$  of a random variable  $X$  defined as *the signal*, is defined as  $\text{SNR} \doteq \frac{\text{Var}[\mathbb{E}[L|X]]}{\mathbb{E}[\text{Var}[L|X]]}$  (where we recall that  $\mathbb{E}[L|X]$  and  $\text{Var}[L|X]$  are both viewed as functions of the random variable  $X$ ).

The Gaussian distribution of dimension  $t$  with  $t$ -size expectation vector  $\boldsymbol{\mu}$  and  $t \times t$  covariance matrix  $\Sigma$  is denoted by  $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ . We recall that the corresponding probability density function (pdf) is defined for every  $\boldsymbol{\ell} \in \mathbb{R}^t$  as:

$$f(\boldsymbol{\ell}) = \frac{1}{\sqrt{(2\pi)^t \det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{\ell} - \boldsymbol{\mu})' \cdot \Sigma^{-1} \cdot (\boldsymbol{\ell} - \boldsymbol{\mu})\right), \quad (1)$$

where  $(\cdot)'$  denotes the transposition operation and  $\det(\cdot)$  denotes the matrix determinant. The corresponding cumulative distribution function (cdf)  $F$  is defined for every  $(a_i, b_i)_{i \in [1..t]} \in ((\mathbb{R} \cup \{-\infty, +\infty\})^2)^t$  by

$$F(\mathbf{a}, \mathbf{b}) = \int_{a_t}^{b_t} \cdots \int_{a_2}^{b_2} \int_{a_1}^{b_1} f(\ell_1, \ell_2, \dots, \ell_t) d\ell_1 d\ell_2 \cdots d\ell_t \doteq \int_{\mathbf{a}}^{\mathbf{b}} f(\boldsymbol{\ell}) d\boldsymbol{\ell}, \quad (2)$$

with  $\boldsymbol{\ell} \doteq (\ell_1, \ell_2, \dots, \ell_t)$ ,  $\mathbf{a} \doteq (a_1, a_2, \dots, a_t)$  and  $\mathbf{b} \doteq (b_1, b_2, \dots, b_t)$ .

If the dimension  $t$  equals 1, then the Gaussian distribution is said to be *univariate* and its covariance matrix is reduced to the variable variance denoted  $\sigma^2$ . Otherwise, it is said to be *multivariate*.

The *entropy*  $\mathbb{H}(X)$  of a discrete r.v.  $X$  defined over  $\mathbb{F}_{2^k}$  aims at measuring the amount of information provided by an observation of  $X$ . It is defined by  $\mathbb{H}(X) = -\sum_{x \in \mathbb{F}_{2^k}} f_X(x) \log f_X(x)$ . The *differential entropy* extends the notion of entropy to continuous, and possibly  $t$ -dimensional random variables; contrary to the entropy, the differential entropy can be negative. In the case of a real valued random variable  $\mathbf{L}$ , it is defined by:

$$\mathbb{H}(\mathbf{L}) = -\int_{\boldsymbol{\ell} \in \mathbb{R}^t} f_{\mathbf{L}}(\boldsymbol{\ell}) \log(f_{\mathbf{L}}(\boldsymbol{\ell})) d\boldsymbol{\ell}. \quad (3)$$

If  $\mathbf{L}$  is a  $t$ -dimensional Gaussian r.v. with covariance matrix  $\Sigma$  (*i.e.* its pdf is defined by (1)), then its entropy satisfies the following equality:

$$\mathbb{H}(\mathbf{L}) = \frac{1}{2} \log((2\pi e)^t \det(\Sigma)). \quad (4)$$

In the general case, there is no analytical expression for the differential entropy of a r.v.  $X$  whose pdf mixes more than one Gaussian pdf. However, upper and lower bounds can be derived [CP00].

### 3 Secure Multiplication Schemes

In this section, we recall the secure multiplication scheme over  $\mathbb{F}_2$  introduced in [ISW03] and its extension to any field  $\mathbb{F}_{2^k}$  proposed in [RP10].

**Ishai-Sahai-Wagner’s Scheme [ISW03].** Let  $x^*$  and  $y^*$  be binary values from  $\mathbb{F}_2$  and let  $(x_i)_{1 \leq i \leq n}$  and  $(y_i)_{1 \leq i \leq n}$  be  $n$ -sharings of  $x^*$  and  $y^*$  respectively. To securely compute a sharing of  $c = x^* \cdot y^*$  from  $(x_i)_{1 \leq i \leq n}$  and  $(y_i)_{1 \leq i \leq n}$ , the ISW method works as follows:

1. For every  $1 \leq i < j \leq n$ , pick up a random bit  $r_{i,j}$ .
2. For every  $1 \leq i < j \leq n$ , compute  $r_{j,i} = (r_{i,j} + x_i \cdot y_j) + x_j \cdot y_i$ .
3. For every  $1 \leq i \leq n$ , compute  $c_i = x_i \cdot y_i + \sum_{j \neq i} r_{i,j}$ .

The above multiplication scheme achieves security at order  $\lfloor n/2 \rfloor$  in the probing security model [ISW03].

**The Rivain-Prouff Scheme.** The ISW countermeasure was extended to  $\mathbb{F}_{2^k}$  by Rivain and Prouff in [RP10]. As showed in [BBD<sup>+</sup>15], the SecMult algorithm below is secure in the ISW probing model against  $t$  probes for  $n \geq t + 1$  shares; the authors also show that with some additional mask refreshing, the Rivain-Prouff countermeasure for the full AES can be made secure with  $n \geq t + 1$  shares.

---

#### Algorithm 1 SecMult

---

**Input:** the  $n$ -sharings  $(x_i)_{i \in [1..n]}$  and  $(y_j)_{j \in [1..n]}$  of  $x^*$  and  $y^*$  respectively

**Output:** the  $n$ -sharing  $(c_i)_{i \in [1..n]}$  of  $x^* \cdot y^*$

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = i + 1$  to  $n$  do
3:      $r_{i,j} \leftarrow \mathbb{F}_{2^k}$ 
4:      $r_{j,i} \leftarrow (r_{i,j} + x_i \cdot y_j) + x_j \cdot y_i$ 
5:   end for
6: end for
7: for  $i = 1$  to  $n$  do
8:    $c_i \leftarrow x_i \cdot y_i$ 
9:   for  $j = 1$  to  $n$ ,  $j \neq i$  do  $c_i \leftarrow c_i + r_{i,j}$ 
10: end for
11: return  $(c_1, c_1, \dots, c_n)$ 

```

---

In Algorithm 1, one can check that each share  $x_i$  or  $y_j$  is manipulated  $n$  times, whereas each product  $x_i y_j$  is manipulated a single time. This gives a total of  $3n^2$  manipulations that can be observed through side channels.

## 4 Horizontal DPA Attack

### 4.1 Problem description.

Let  $(x_i)_{i \in [1..n]}$  and  $(y_i)_{i \in [1..n]}$  be respectively the  $n$ -sharings of  $x^*$  and  $y^*$  (namely, we have  $x^* = x_1 + \dots + x_n$  and  $y^* = y_1 + \dots + y_n$ ). We assume that an adversary gets, during the processing of Algorithm 1, a single observation of each of the following random variables for  $1 \leq i, j \leq n$ :

$$L_i = \varphi(x_i) + B_i \tag{5}$$

$$L'_j = \varphi(y_j) + B'_j \tag{6}$$

$$L''_{ij} = \varphi(x_i \cdot y_j) + B''_{ij} \tag{7}$$

where  $\varphi$  is an unknown function which depends on the device architecture, where  $B_i, B'_j$  are Gaussian noise of standard deviation  $\sigma/\sqrt{n}$ , and  $B''_{ij}$  is Gaussian noise with standard deviation  $\sigma$ . Namely we assume that each  $x_i$  and  $y_j$  is processed  $n$  times, so by averaging the standard deviation is divided by a factor  $\sqrt{n}$ , which gives  $\sigma/\sqrt{n}$  if we assume that the initial noise standard deviation is  $\sigma$ . The random variables associated to the  $i$ th share  $x_i$  and the  $j$ th share  $y_j$  are respectively denoted by  $X_i$  and  $Y_j$ . Our goal is to recover the secret variable  $x^*$  (and/or  $y^*$ ).

## 4.2 Complexity Lower Bound: Entropy Analysis of Noisy Hamming Weight Leakage

For simplicity, we first restrict ourselves to a leakage function  $\varphi$  equal to the Hamming weight of the variable being manipulated. In that case, the mutual information  $I(X; L)$  between the Hamming weight of a uniform random variable  $X$  defined over  $\mathbb{F}_{2^k}$  and a noisy observation  $L$  of this Hamming weight can be approximated as:

$$I(X; L) \simeq \frac{k}{8\sigma^2} , \quad (8)$$

if the noise being modeled by a Gaussian random variable has standard deviation  $\sigma$ . This approximation, whose derivation is given in Appendix A, is only true for large  $\sigma$ .

To recover a total of  $2n$  shares ( $n$  shares of  $x^*$  and  $y^*$  respectively) from  $3n^2$  Hamming weight leakages (namely each manipulation leaks according to (5)-(7) with  $\varphi = \text{HW}$ ), the total amount of information to be recovered is  $2n \cdot k$  if we assume that the shares are i.i.d. with uniform distribution over  $\mathbb{F}_{2^k}$ . Therefore, since we have a total of  $3n^2$  observations during the execution of Algorithm 1, we obtain from (8) that the noise standard deviation  $\sigma$  and the sharing order  $n$  must satisfy the following inequality for a side channel attack to be feasible:

$$3 \cdot n^2 \cdot \frac{k}{8\sigma^2} > 2n \cdot k . \quad (9)$$

We obtain an equality of the form  $n > c \cdot \sigma^2$  for some constant  $c$ , as in a classical (vertical) side channel attack trying to recover  $x^*$  from  $n$  observations of intermediate variables depending on  $x^*$  [CJRR99]. This analogy between horizontal and vertical attacks has already been noticed in previous papers like [CFG<sup>+</sup>10] or [BJPW13]. Note that in principle the constant  $c$  is independent of the field degree  $k$  (which has also been observed in previous papers, see for instance [SVO<sup>+</sup>10]).

## 4.3 Attack With Perfect Hamming Weight Observations

We consider the particular case of perfect Hamming weight measurements (no noise). We show that even with perfect observations of the Hamming weight, depending on the finite-field representation, we are not always guaranteed to recover the secret variable  $x^*$ .

More precisely, we consider the pdf of the random variable corresponding to perfect Hamming weight measurements:

$$\mathbf{H} \mid X_i \doteq (\text{HW}(X_i), \text{HW}(Y_j), \text{HW}(X_i \cdot Y_j)) \mid X_i$$

defined for every  $x_i \in \mathbb{F}_{2^k}$  and every triplet  $(h_1, h_2, h_3) \in [0..k]^3$  by:

$$f_{\mathbf{H} \mid X_i}((h_1, h_2, h_3), x_i) \doteq \Pr[\text{HW}(X_i) = h_1, \text{HW}(Y_j) = h_2, \text{HW}(X_i \cdot Y_j) = h_3 \mid X_i = x_i] . \quad (10)$$

If the probability distributions  $f_{\mathbf{H} \mid X_i = x_i}$  are distinct for every  $x_i \in \mathbb{F}_{2^k}$ , then one can recover the secret variable  $x^*$  with overwhelming probability from enough measurements.

However this property depends on the finite field  $\mathbb{F}_{2^k}$  and its representation. For example in  $\mathbb{F}_{2^4}$ , if the representation  $\mathbb{F}_{2^4} \simeq \mathbb{F}_2[t]/(t^4 + t + 1)$  is used then it may be checked that the finite field elements  $x_i = t + 1$  and  $x'_i = t^3 + t^2$  are associated to identical distributions  $f_{x_i}$  and  $f_{x'_i}$ ; so we cannot distinguish between  $x_i$  and  $x'_i$  (the other field elements have distinct probability distributions). In  $\mathbb{F}_{2^8}$ , for the representation  $\mathbb{F}_{2^8} \simeq \mathbb{F}_2[t]/(t^8 + t^4 + t^3 + t + 1)$  (as used in AES), all finite field elements have distinct probability distributions, but this is not always the case with other irreducible polynomials.

In summary, even with perfect observations of the Hamming weight, depending on the finite-field representation, we are not always guaranteed to recover the secret variable  $x^*$ ; however for the finite field representation used in AES the attack enables to recover the secret  $x^*$  for a large enough number of observations.

#### 4.4 Maximum Likelihood Attack: Theoretical Attack with the full ISW State

For most field representations and leakage functions, the maximum likelihood approach used in the previous section (in particular in (10)), recovers the  $i$ -th share of  $x^*$  from an observation of  $L_i$  and an observation of  $(L'_j, L''_{ij})$  for every  $j \in [1..n]$ . It extends straightforwardly to noisy scenarios and we shall detail this extension in Section 5.1. However, the disadvantage of this approach is that it recovers each share separately, before rebuilding  $x^*$  and  $y^*$  from them. From a pure information theoretic point of view this is suboptimal since (1) the final purpose is not to recover all the shares perfectly but only the shared values and (2) only  $3n$  observations are used to recover each share whereas the full tuple of  $3n^2$  observations brings more information. Actually, the most efficient attack in terms of leakage exploitation consists in using the joint distribution of  $(L_i, L'_j, L''_{ij})_{i,j \in [1..n]}$  to distinguish the correct hypothesis about  $x^* = x_1 + x_2 + \dots + x_n$  and  $y^* = y_1 + y_2 + \dots + y_n$ .

As already observed in Section 3, during the processing of Algorithm 1, the adversary may get a tuple  $(\ell_{ij})_{j \in [1..n]}$  (resp.  $(\ell'_{ij})_{i \in [1..n]}$ ) of  $n$  observations for each  $L_i$  (resp. each  $L'_j$ ) and one observation  $\ell''_{ij}$  for each  $L''_{ij}$ . The full tuple of observations  $(\ell_{ij}, \ell'_{ij}, \ell''_{ij})_{i,j}$  is denoted by  $\ell$ , and we denote by  $\mathbf{L}$  the corresponding random variable<sup>1</sup>. Then, to recover  $(x^*, y^*)$  from  $\ell$ , the *maximum likelihood approach* starts by estimating the pdfs  $f_{\mathbf{L}|X^*=x^*, Y^*=y^*}$  for every possible  $(x^*, y^*)$ , and then estimates the following vector of distinguisher values for every hypothesis  $(x, y)$ :

$$d_{\text{ML}}^*(\ell) \doteq (f_{\mathbf{L}|(X^*, Y^*)}(\ell, (x, y)))_{(x,y) \in \mathbb{F}_{2^k}^2} \quad (11)$$

The pair  $(x, y)$  maximizing the above probability is eventually chosen.

At a first glance, the estimation of the pdfs  $f_{\mathbf{L}|X^*=x^*, Y^*=y^*}$  seems to be challenging. However, it can be deduced from the estimations of the pdfs associated to the manipulations of the shares. Indeed, after denoting by  $p_{x,y}$  each probability value in the right-hand side of (11), and by using the law of total probability together with the fact that the noises are independent, we get:

$$2^{2kn} \cdot p_{x,y} = \sum_{\substack{x_1, \dots, x_n \in \mathbb{F}_{2^k} \\ x = x_1 + \dots + x_n}} \sum_{\substack{y_1, \dots, y_n \in \mathbb{F}_{2^k} \\ y = y_1 + \dots + y_n}} \prod_{i,j=1}^n f_{L_i|X_i}(\ell_{ij}, x_i) \cdot f_{L'_j|Y_j}(\ell'_{ij}, y_j) \cdot f_{L''_{ij}|X_i Y_j}(\ell''_{ij}, x_i y_j) \cdot$$

Unfortunately, even if the equation above shows how to deduce the pdfs  $f_{\mathbf{L}|(X^*, Y^*)}(\cdot, (x^*, y^*))$  from characterizations of the shares' manipulations, a direct processing of the probability

<sup>1</sup> In (5)-(7), it is assumed that the observations  $(\ell_{ij})_{j \in [1..n]}$  and  $(\ell'_{ij})_{i \in [1..n]}$  are averaged to build a single observation with noise divided by  $\sqrt{n}$ . This assumption is not done here in order to stay as general as possible.

has complexity  $O(2^{2nk})$ . By representing the sum over the  $x_i$ 's as a sequence of convolution products, and thanks to Walsh transforms processing, the complexity can be easily reduced to  $O(n2^{n(k+1)})$ . The latter complexity stays however too high, even for small values of  $n$  and  $k$ , which led us to look at alternatives to this attack.

## 5 First Attack: Maximum Likelihood Attack on a Single Matrix Row

### 5.1 Attack Description

In this section, we explain how to recover each share  $x_i$  of  $x^*$  separately, by observing the processing of Algorithm 1. Applying this attack against all the shares leads to the full recovery of the sensitive value  $x^*$  with some success probability, which is essentially the product of the success probabilities of the attack on each share separately.

Given a share  $x_i$ , the attack consists in collecting the leakages on  $(y_j, x_i \cdot y_j)$  for every  $j \in [1..n]$ . Therefore the attack is essentially a horizontal version of the classical (vertical) second-order side-channel attack, where each share  $x_i$  is multiplicatively masked over  $\mathbb{F}_{2^k}$  by a random  $y_j$  for  $j \in [1..n]$ .

The most efficient attack to maximize the amount of information recovered on  $X_i$  from a tuple of observations  $\boldsymbol{\ell} \doteq \ell_i, (\ell'_j, \ell''_{ij})_{j \in [1..n]} \leftrightarrow \mathbf{L} \doteq L_i, (L'_j, L''_{ij})_{j \in [1..n]}$  consists in applying a *maximum likelihood approach* [CJRR99, GHR15], which amounts to computing the following vector of distinguisher values:

$$d_{\text{ML}}(\boldsymbol{\ell}) \doteq (f_{\mathbf{L}|X_i}(\boldsymbol{\ell}, \hat{x}_i))_{\hat{x}_i \in \mathbb{F}_{2^k}} \quad (12)$$

and in choosing the candidate  $\hat{x}_i$  which maximizes the probability.

Let us respectively denote by  $f(\cdot, \cdot)$ ,  $f'(\cdot, \cdot)$  and  $f''(\cdot, \cdot)$  the pdfs  $f_{L_i|X_i}(\cdot, \cdot)$ ,  $f_{L'_j|Y_j}(\cdot, \cdot)$  and  $f_{L''_{ij}|X_i Y_j}(\cdot, \cdot)$ . Since the variables  $L_i | X_i = \hat{x}_i$  and all the variables  $(L'_j, L''_{ij} | X_i = \hat{x}_i)$  are mutually independent whatever  $\hat{x}_i \in \mathbb{F}_{2^k}$ , we have:

$$f_{\mathbf{L}|X_i}(\boldsymbol{\ell}, \hat{x}_i) = f(\ell_i, \hat{x}_i) \prod_{j=1}^n f_{(L'_j, L''_{ij})|X_i}((\ell'_j, \ell''_{ij}), \hat{x}_i) . \quad (13)$$

Applying the law of total probability, the pdf of  $(L'_j, L''_{ij}) | X_i = \hat{x}_i$  can moreover be developed such that:

$$f_{(L'_j, L''_{ij})|X_i}((\ell'_j, \ell''_{ij}), \hat{x}_i) = \sum_{y \in \mathbb{F}_{2^k}} f_{(L'_j, L''_{ij})|(X_i, Y_j)}((\ell'_j, \ell''_{ij}), (\hat{x}_i, y)) \cdot \text{PY}_j(y) , \quad (14)$$

that is

$$f_{(L'_j, L''_{ij})|X_i}((\ell'_j, \ell''_{ij}), \hat{x}_i) = 2^{-k} \sum_{y \in \mathbb{F}_{2^k}} f'(\ell'_j, y) \cdot f''(\ell''_{ij}, \hat{x}_i \cdot y) , \quad (15)$$

since the  $Y_j$ 's are assumed to have uniform distribution and since the variables  $L'_j | Y_j = y$  and  $L''_{ij} | X_i Y_j = \hat{x}_i \cdot y$  are independent.

Eventually, each score  $f_{\mathbf{L}|X_i}(\boldsymbol{\ell}, \hat{x}_i)$  in (12) may be computed based on the following expression:

$$f_{\mathbf{L}|X_i}(\boldsymbol{\ell}, \hat{x}_i) = 2^{-nk} f(\ell_i, \hat{x}_i) \cdot \prod_{j=1}^n \left( \sum_{y \in \mathbb{F}_{2^k}} f'(\ell'_j, y) \cdot f''(\ell''_{ij}, \hat{x}_i \cdot y) \right) ,$$

where all the distributions are Gaussian ones (and hence can be easily evaluated). Hence, an approximation of the pdf in (12) can be deduced from the approximations (aka *templates*) of the distributions associated to the manipulations of the shares  $X_i$ ,  $Y_j$  and  $X_i Y_j$ .

In practice, one often makes use of the equivalent (averaged) log-likelihood distinguisher  $d'_{\text{ML}}(\cdot)$  which, in our case, may be defined as:

$$d'_{\text{ML}}(\ell) = \frac{1}{n} \log d_{\text{ML}}(\ell) + k \log 2 \quad (16)$$

$$\simeq \left( \frac{1}{n} \left( \log f(\ell_i, \hat{x}_i) + \sum_{j=1}^n \log \left\{ \sum_{y \in \mathbb{F}_{2^k}} f'(\ell'_j, y) \cdot f''(\ell''_{ij}, \hat{x}_i \cdot y) \right\} \right) \right)_{\hat{x}_i \in \mathbb{F}_{2^k}} . \quad (17)$$

*Remark 1.* The same approach described in this section can be applied to iteratively recover each share  $y_j$  of  $y$ . The attack description can be straightforwardly deduced by exchanging the roles of  $X_i$  and  $Y_j$  (and the indices  $i$  and  $j$ ). For instance, (14) becomes:

$$f_{(L_i, L''_{ij})|Y_j}((\ell_i, \ell''_{ij}), \hat{y}_j) = \sum_{x \in \mathbb{F}_{2^k}} f_{(L_i, L''_{ij})|(X_i, Y_j)}((\ell_i, \ell''_{ij}), (x, \hat{y}_j)) \cdot p_{X_i}(x) . \quad (18)$$

## 5.2 Complexity Analysis

As mentioned previously, given a share  $x_i$ , the attack consists in collecting the leakages on  $(y_j, x_i \cdot y_j)$  for every  $j \in [1..n]$ . Therefore the attack is essentially an horizontal version of the classical (vertical) second-order side-channel attack. In principle the number  $n$  of leakage samples needed to recover  $x_i$  with good probability (aka the attack complexity) should consequently be  $n = \mathcal{O}(\sigma^4)$  [CJRR99, GHR15, SVO<sup>+</sup>10]. This holds when multiplying two leakages both with noise having  $\sigma$  as standard deviation. However here the leakage on  $y_j$  has a noise with a standard deviation  $\sigma/\sqrt{n}$  instead of  $\sigma$  (thanks to the averaging step). Therefore the noise of the product becomes  $\sigma^2/\sqrt{n}$  (instead of  $\sigma^2$ ), which gives after averaging with  $n$  measurements a standard deviation of  $\sigma^2/n$ , and therefore an attack complexity satisfying  $n = \mathcal{O}(\sigma^2)$ , as in a classical first-order side-channel attack.

## 5.3 Numerical Experiments

The attack presented in Sect. 5.1 has been implemented against each share  $x_i$  of a value  $x$ , with the leakages being simulated according to (5)-(7) with  $\varphi = \text{HW}$ . For the noise standard deviation  $\sigma$  and the sharing order  $n$ , different values have been tested to enlighten the relation between these two parameters. We stated that an attack succeeds iff the totality of the  $n$  shares  $x_i$  have been recovered, which leads to the full recovery of  $x^*$ . We recall that, since the shares  $x_i$  are manipulated  $n$  times, measurements for the leakages  $L_i$  and  $L'_j$  have noise standard deviations  $\sigma/\sqrt{n}$  instead of  $\sigma$ . For efficiency reasons, we have chosen to work in the finite field  $\mathbb{F}_{2^4}$  (namely  $k = 4$  in previous analyses).

For various noise standard deviations  $\sigma$  with  $\text{SNR} = k(2\sigma)^{-2}$  (*i.e.*  $\text{SNR} = \sigma^{-2}$  for  $k = 4$ ), Table 1 gives the average minimum number  $n$  of shares required for the attack to succeed with probability strictly greater than 0.5 (the averaging being computed over 300 attack iterations). The attack complexity  $n = \mathcal{O}(\sigma^2)$  argued in Sect. 5.2 is confirmed by the trend of these numerical experiments. Undeniably, this efficiency is quickly too poor for practical applications where  $n$  is small (clearly lower than 10) and the SNR is high (smaller than 1). However, it must be noticed that the attack quickly recovers 90% of the shares even for  $\sigma = 2, 3$  (see Figure 1a), and the shares which are not recovered (because they are not given the maximum likelihood) have a good ranking. Consequently, combining this attack with one of the Key Enumeration (KEA) techniques recently developed (see *e.g.* [GGP<sup>+</sup>15, MOOS15]) should significantly increase the attack efficiency.

$\sigma$ (SNR)	0 ( $+\infty$ )	0.2 (25)	0.4 (6.25)	0.6 (2.77)	0.8 (1.56)	1 (1)
$n$	12	14	30	73	160	284

Table 1: First attack: number of shares  $n$  as a function of the noise  $\sigma$  to succeed with probability  $> 0.5$

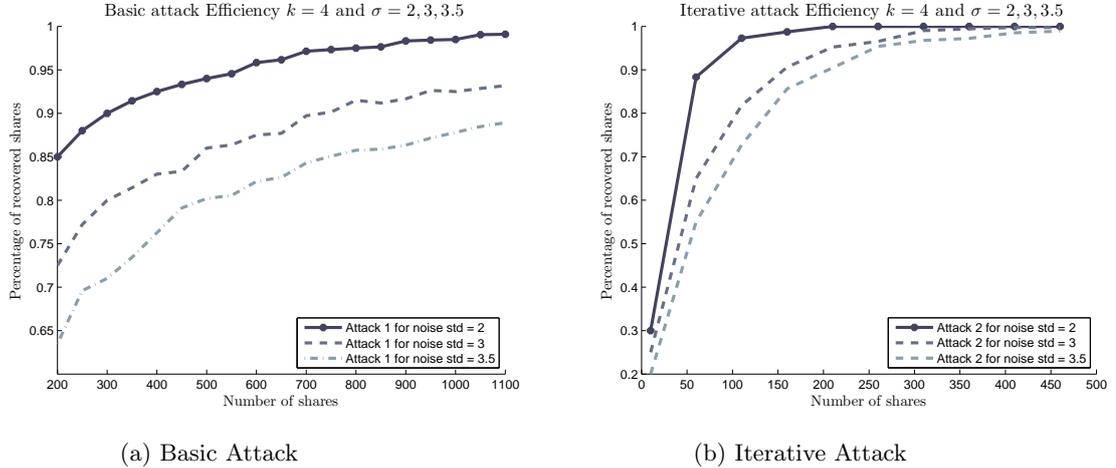


Fig. 1: Percentage of recovered shares with respect to  $n$  for  $\sigma = 2, 3, 3.5$  and  $k = 4$

## 6 Second Attack: Iterative Attack

### 6.1 Attack Description

From the discussions in Sect. 4.4, and in view of the poor efficiency of the previous attack, we investigated another strategy which targets all the shares simultaneously. Essentially, the core idea of our second attack described below is to apply several attacks recursively on the  $x_i$ 's and  $y_j$ 's, and to refine step by step the likelihood of each candidate for the tuple of shares. Namely, we start by applying the attack described in Section 5.1 in order to compute, for every  $i$ , a likelihood probability for each hypothesis  $X_i = x$  ( $x$  ranging over  $\mathbb{F}_{2^k}$ ); then we apply the same attack in order to compute, for every  $j$ , a likelihood probability for each hypothesis  $Y_j = y$  ( $y$  ranging over  $\mathbb{F}_{2^k}$ ) with the single difference that the probability  $p_{X_i}(x)$  in (18) is replaced by the likelihood probability which was just computed<sup>2</sup>. Then, one reiterates the attack to refine the likelihood probabilities  $(p_{X_i}(x))_{x \in \mathbb{F}_{2^k}}$ , by evaluating (14) with the uniform distribution  $p_{Y_j}(y)$  being replaced by the likelihood probability  $\text{new-}p_{Y_j}(y)$  which has been previously computed. The scheme is afterwards repeated until the maximum taken by the pdfs of each share  $X_i$  and  $Y_j$  is greater than some threshold  $\beta$ . In order to have better results, we perform the whole attack a second time, by starting with the computation of the likelihood probability for each hypothesis  $Y_j = y$  instead of starting by  $X_i = x$ .

We give the formal description of the attack processing in Algorithm 2 (in order to have the complete attack, one should perform the while loop a second time, by rather starting with the computation of  $\text{new-}p_{Y_j}(y)$  instead of  $\text{new-}p_{X_i}(x)$ ).

### 6.2 Numerical Experiments

The iterative attack described in Algorithm 2 has been tested against leakages simulations defined exactly as in Section 5.3. As previously we stated that an attack succeeds if the

<sup>2</sup> Actually to get the probability of  $X_i | \mathbf{L}$  instead of  $\mathbf{L} | X_i$ , Bayes' Formula is applied which explains the division by the sum of probabilities in the lines 14 and 19 in Algorithm 2.

---

**Algorithm 2** Iterative Maximum Likelihood Attack

---

**Input:** a threshold  $\beta$ , an observation  $\ell_i$  of each  $L_i$ , an observation  $\ell'_j$  of each  $L'_j$  and one observation  $\ell''_{ij}$  of each  $L''_{ij}$  (the random variables being defined as in (5)-(7))

**Output:** a  $n$ -tuple of pdfs  $(p_{X_i})_i$  (resp.  $(p_{Y_i})_i$ ) such that, for every  $i \in [1..n]$ , at least one  $\hat{x}_i$  (resp.  $\hat{y}_i$ ) satisfies  $p_{X_i}(\hat{x}_i) \geq \beta$  (resp.  $p_{Y_i}(\hat{y}_i) \geq \beta$ )

```
1: for  $i = 1$  to  $n$  do
2:   for  $x \in \mathbb{F}_{2^k}$  do                                     # Initialize the likelihood of each candidate for  $X_i$ 
3:      $p_{X_i}(x) = f_{L_i|X_i}(\ell_i, x)$ 
4:   end for
5:   for  $y \in \mathbb{F}_{2^k}$  do                                     # Initialize the likelihood of each candidate for  $Y_i$ 
6:      $p_{Y_i}(y) = f_{L'_i|Y_i}(\ell'_i, y)$ 
7:      $\text{new-}p_{Y_i}(y) = p_{Y_i}(y)$ 
8:   end for
9: end for

10: while  $\text{end} \neq n$  do
11:    $\text{end} \leftarrow 0$ 
12:   for  $i = 1$  to  $n$  do
13:     for  $x \in \mathbb{F}_{2^k}$  do                                     # Compute/Update the likelihood of each candidate for  $X_i$ 
14:        $\text{new-}p_{X_i}(x) = 2^{-(2n+1)k} \frac{p_{X_i}(x)}{\sum_{x' \in \mathbb{F}_{2^k}} p_{X_i}(x')} \prod_{j=1}^n \sum_{y \in \mathbb{F}_{2^k}} \frac{\text{new-}p_{Y_j}(y)}{\sum_{y' \in \mathbb{F}_{2^k}} \text{new-}p_{Y_j}(y')} \cdot f_{L''_{ij}|X_i Y_j}(\ell''_{ij}, x \cdot y)$ 
15:     end for
16:   end for
17:   for  $i = 1$  to  $n$  do
18:     for  $y \in \mathbb{F}_{2^k}$  do                                     # Compute/Update the likelihood of each candidate for  $Y_i$ 
19:        $\text{new-}p_{Y_i}(y) = 2^{-(2n+1)k} \frac{p_{Y_i}(y)}{\sum_{y' \in \mathbb{F}_{2^k}} p_{Y_i}(y')} \prod_{j=1}^n \sum_{x \in \mathbb{F}_{2^k}} \frac{\text{new-}p_{X_j}(x)}{\sum_{x' \in \mathbb{F}_{2^k}} \text{new-}p_{X_j}(x')} \cdot f_{L''_{ij}|X_i Y_j}(\ell''_{ij}, x \cdot y)$ 
20:     end for
21:   end for
22:   for  $i = 1$  to  $n$  do
23:     if  $\max_x(\text{new-}p_{X_i}(x)) \geq \beta$  and  $\max_y(\text{new-}p_{Y_i}(y)) \geq \beta$  then
24:        $\text{end} \leftarrow \text{end} + 1$ 
25:     end if
26:   end for
27: end while
```

---

totality of the  $n$  shares  $x_i$  have been recovered, which leads to the full recovery of  $x^*$ . For various noise standard deviations  $\sigma$  with  $\text{SNR} = k(2\sigma)^{-2}$ , Table 2 gives the average minimum number of shares  $n$  required for the attack to succeed with probability strictly greater than 0.5 (the averaging being computed over 300 attack iterations). The first row corresponds to  $k = 4$ , and the second row to  $k = 8$  (the corresponding SNRs are  $\text{SNR}_4 = \sigma^{-2}$  and  $\text{SNR}_8 = (\sqrt{2}\sigma^2)^{-1}$ ). Numerical experiments yield greatly improved results in comparison to those obtained by running the basic attack. Namely, in  $\mathbb{F}_{2^4}$ , for a noise  $\sigma = 0$ , the number of shares required is 2, while 12 shares were needed for the basic attack, and the improvement is even more confirmed with a growing  $\sigma$ : for a noise  $\sigma = 1$ , the number of shares required is 25, while 284 shares were needed for the basic attack. It can also be observed that the results for shares in  $\mathbb{F}_{2^4}$  and  $\mathbb{F}_{2^8}$  are relatively close, the number of shares being most likely slightly smaller for shares in  $\mathbb{F}_{2^4}$  than in  $\mathbb{F}_{2^8}$ . This observation is in-line with the lower bound in (9), where the cardinality  $2^k$  of the finite field plays no role. Once again, it may be observed that the attack quickly recovers 90% of the shares even for  $\sigma \in \{2, 3, 3.5\}$  (see Figure 1b), and the shares which are not recovered (because they are not given the maximum likelihood) have a good ranking. Consequently, combining this attack with KEA should still increase the attack efficiency.

$\sigma$ ( $\text{SNR}_4, \text{SNR}_8$ )	0 ( $+\infty, +\infty$ )	0.2 (25, 17.67)	0.4 (6.25, 4.41)	0.6 (2.77, 1.96)	0.8 (1.56, 1.10)	1 (1, 0.7071)
$n$ (for $\mathbb{F}_{2^4}$ )	2	2	3	6	13	25
$n$ (for $\mathbb{F}_{2^8}$ )	5	6	8	11	16	21

Table 2: Iterative attack: number of shares  $n$  as a function of the noise  $\sigma$  to succeed with probability  $> 0.5$  in  $\mathbb{F}_{2^4}$  (first row) and in  $\mathbb{F}_{2^8}$  (second row).

## 7 Practical Results

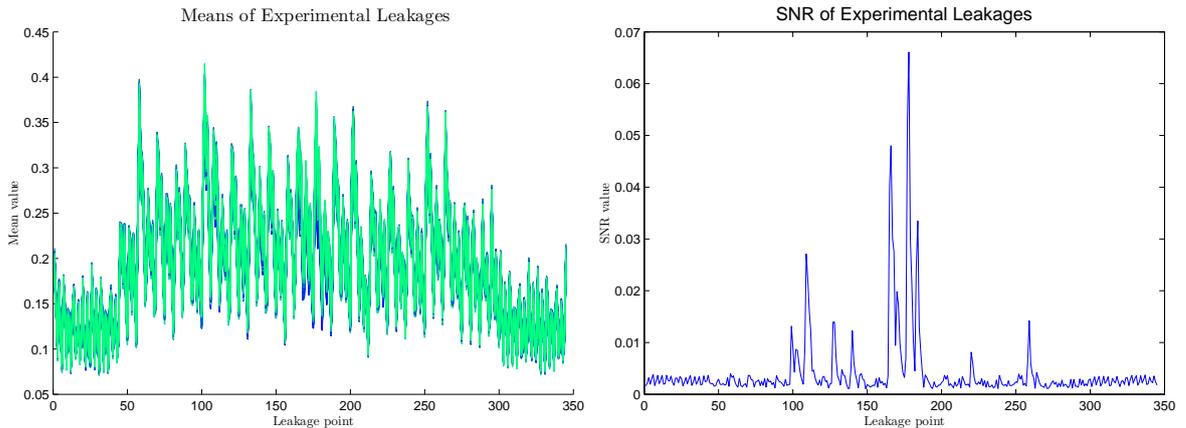
### 7.1 Setup

In order to provide real life experiments of the attack described in this work we have mounted it against a development platform embedding the `ATMega328` processor. The presence of decoupling capacitors between the ground pins of the processor and the reference ground did not allowed us to correctly measure the power consumption of the processor. We therefore de-soldered the ground pins of the processor and connected them to a 20 Ohm resistor whose other end was connected to the reference ground. After this preparation, we used a passive probe connected to an oscilloscope in order to register the current absorbed by the processor by measuring the difference of potential at the ends of the resistor. Thanks to a probe bandwidth of 500MHz, we pre-filtered all the frequencies higher than 200MHz. We moreover used a sampling rate of 100MHz on the oscilloscope as the best compromise between accuracy and measurement trace size.

### 7.2 Leakage Characterization

For our implementation of Algorithm 1, the `mov` instruction is used to manipulate the shares  $x_i, y_j$  and the multiplication results  $x_i \cdot y_j$ . We therefore chose to target it in our attack experiments. An advantage of targeting a single instruction that manipulates all the values is that we obtain homogeneous leakages for all the manipulated data. Furthermore, the `mov` instruction may be found in many different architectures and we therefore think that our attack can be reproduced quite easily.

For the leakage characterization phase, we measured 200,000 leakages of the `mov` instruction parametrized with randomly generated values. Each measure was composed of 340 points, which is essentially the size of a relevant sample of instantaneous measures for the `mov` instruction in our setup. This campaign allowed us to characterize the leakage related to the processing `mov y, x` for  $x$  ranging between 0 and 255 and for  $y$  being constant (our implementation uses the same destination register for all the shares); more precisely, each  $x$  was associated to a mean vector  $\mu_{\mathbf{x}} \in \mathbb{R}^{340}$  and a covariance matrix  $\Sigma_{\mathbf{x}} \in \mathbb{R}^{340} \times \mathbb{R}^{340}$ . The 256 means are plotted in Figure 2a. To reduce the dimension of our *templates*, we afterwards estimated the signal-to-noise ratio of the acquisitions at each point in order to identify the best points of interest for our attack. The results are plotted in Figure 2b.



(a) Mean vectors of all 256 classes.

(b) SNR on each leakage point.

Fig. 2: Results of the leakage characterization phase.

The peaks in 2b allowed us to choose a first set of points. In addition, a simple maximum likelihood attack on each of these points separately has also been mounted in order to find those who provided the most information. This step finally provided us with a set of 11 points to mount our attack. It may be observed that the best SNR obtained with our setup is around 0.07 which, for our simulations, corresponds to the case  $k = 16$  and  $\sigma = 3.75$ , and  $k = 256$  and  $\sigma = 3.2$ . Our experiments should therefore be compared to the simulations plotted in the gray dot-and-dashed curve in Fig. 1a (for the basic attack) and in Fig. 1b (for the iterative attack).

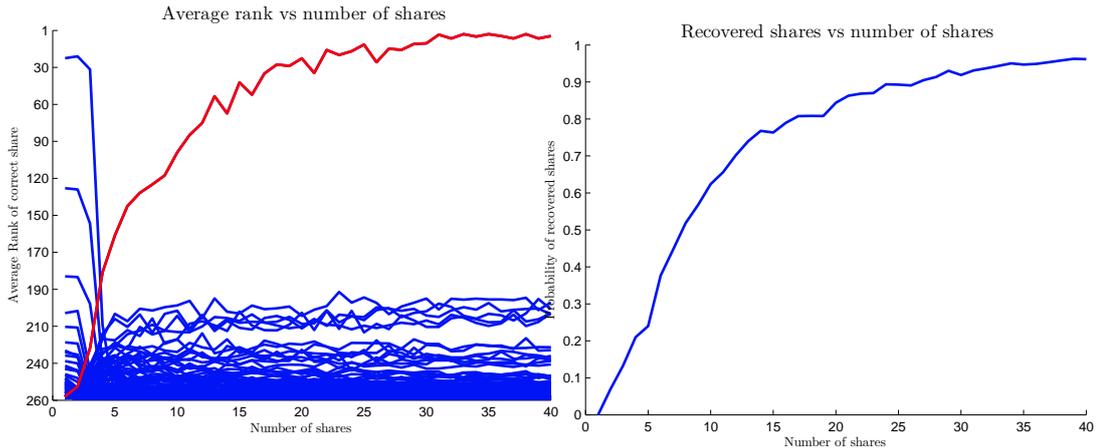
### 7.3 Attacks

Thanks to the characterization described in the previous section, we performed the attacks of Sect. 5.1 and Sect. 6 against our implementation of Algorithm 1 parametrized with different orders  $n$ .

**Maximum Likelihood Attack Experiments** As a first experiment we tested the attack of Section 5.1 in order to evaluate the evolution of the rank of the correct hypothesis on a single share  $x_i$  when  $n$  varies. In 3a we plot the ranking of the correct hypothesis, in red, among all 256 byte values, for each  $n$  between 1 and 40. The results have been obtained by averaging the result among 100 repetitions of the same attack for each order. From 3a we can observe that the correct hypothesis is ranked among the first 50 values as soon as  $n > 10$ .

We also observe that we need  $n > 35$  for the correct hypothesis to be firmly ranked first by the basic attack.

During the same attack we have also evaluated the average number of shares of  $x^*$  correctly retrieved for each order  $n$ . This allowed us to obtain an estimation of the minimum order  $n$  required to successfully mount the attack. The result of such evaluation is depicted in 3b. Even if  $n = 40$  appears to be a necessary condition to directly recover all the shares, the post-processing of our attack results with a KEA algorithm [GGP<sup>+</sup>15, MOOS15] should allow to recover them for smaller  $n$  ( $n = 10$  seems to be achievable at a reasonable cost).



(a) Avg rank of the correct hypothesis vs number of shares.

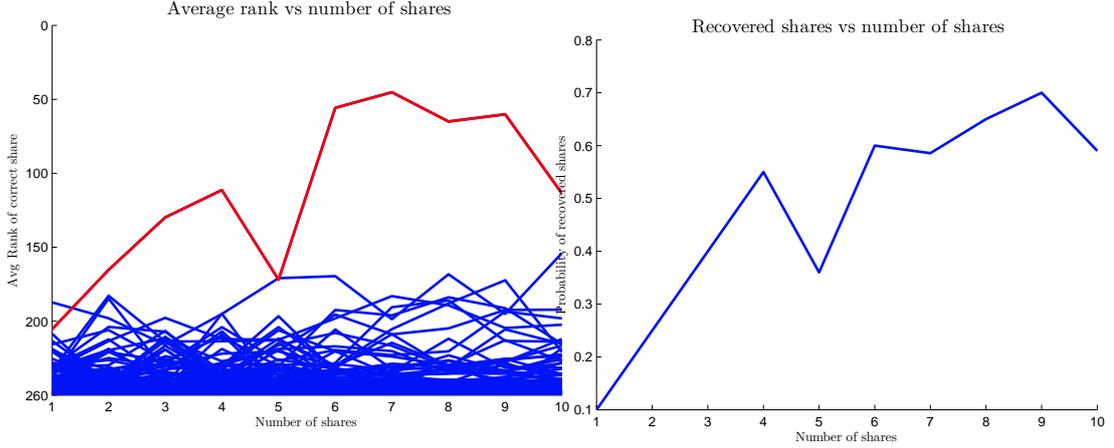
(b) Avg rate of correctly retrieved shares.

Fig. 3: Results for the attack of Section 5.1 when  $n$  varies between 1 and 40.

**Iterative Attack Experiments** We have afterwards mounted the attack described in Section 6 in order to measure the success probability with respect to the masking order. As before we provide the average ranking of the correct hypothesis (in red) for different values of the order  $n$ . Due to time constraints only 10 repetitions of the attack have been averaged. The evolution of the rank of the correct hypothesis is depicted in 4a. As before, we also provide the average number of shares correctly retrieved for each order  $n$  in 4b.

Even if we did not get results as smooth as for previous experiments, we can observe an overall better detection rate for the iterative attack if we compare to the basic one (as actually expected). Furthermore, we have averaged the results of the above attack on 10 repetitions when  $n = 20, 30, 40$ . For such order we have obtained an average number of correctly retrieved shares of 0.88, 0.93 and 0.97, respectively. We thus remark again that by using KEA post-processing on the results of our attack the correct hypothesis should be recovered with a reasonable number of shares. In particular it seems reasonable to assume that  $n = 10$  provides better results and lower costs with respect to the basic attack.

**Experiments Conclusions.** We have successfully proved the effectiveness of our attack on a real implementation using the *ATMega328* processor. We obtain better results on our experiments than those predicted by theory in 5.1 and 6 for similar SNR values. We have investigated such behavior and we conjecture that the better results are due to the use of a multi-variate attack on 11 points for the experimental attacks, where the theoretical results



(a) Avg rank of the correct hypothesis vs number of shares. (b) Avg number of correctly retrieved shares.

Fig. 4: Results for the attack of Section 6 when  $n$  varies between 1 and 10.

are computed for a mono-variate attack. The more points allows for an improvement of the global SNR which can explain the disparity between theory and practice success probabilities.

## 8 A Countermeasure against the previous Attacks

### 8.1 Description

In the following, we describe a countermeasure against the previous attack against the Rivain-Prouff algorithm. More precisely, we describe a variant of Algorithm 1, called RefSecMult, to compute an  $n$ -sharing of  $c = x^* \cdot y^*$  from  $(x_i)_{i \in [1..n]}$  and  $(y_j)_{j \in [1..n]}$ . Our new algorithm is still provably secure in the original ISW probing model, and heuristically secure against the horizontal side-channel attacks described in the previous sections.

---

#### Algorithm 3 RefSecMult

---

**Input:**  $n$ -sharings  $(x_i)_{i \in [1..n]}$  and  $(y_j)_{j \in [1..n]}$  of  $x^*$  and  $y^*$  respectively

**Output:** an  $n$ -sharing  $(c_i)_{i \in [1..n]}$  of  $x^* \cdot y^*$

```

1:  $M_{ij} \leftarrow \text{MatMult}((x_1, \dots, x_n), (y_1, \dots, y_n))$ 
2: for  $i = 1$  to  $n$  do
3:   for  $j = i + 1$  to  $n$  do
4:      $r_{i,j} \leftarrow^{\$} \mathbb{F}_{2^k}$ 
5:      $r_{j,i} \leftarrow (r_{i,j} + M_{ij}) + M_{ji}$ 
6:   end for
7: end for
8: for  $i = 1$  to  $n$  do
9:    $c_i \leftarrow M_{ii}$ 
10:  for  $j = 1$  to  $n$ ,  $j \neq i$  do  $c_i \leftarrow c_i + r_{i,j}$ 
11: end for
12: return  $(c_1, c_1, \dots, c_n)$ 

```

---

As observed in [FRR<sup>+</sup>10], the ISW and Rivain-Prouff countermeasures can be divided in two steps: a “matrix” step in which starting from the input shares  $x_i$  and  $y_j$ , one obtains a matrix  $x_i \cdot y_j$  with  $n^2$  elements, and a “compression” step in which one uses some randomness to get back to an  $n$ -sharing  $c_i$ . Namely the matrix elements  $(x_i \cdot y_j)_{1 \leq i, j \leq n}$  form an  $n^2$ -sharing

of  $x^* \cdot y^*$ :

$$x^* \cdot y^* = \left( \sum_{i=1}^n x_i \right) \cdot \left( \sum_{j=1}^n y_j \right) = \sum_{1 \leq i, j \leq n} x_i \cdot y_j \quad (19)$$

and the goal of the compression step is to securely go from such an  $n^2$ -sharing of  $x^* \cdot y^*$  to an  $n$ -sharing of  $x^* \cdot y^*$ .

Our new countermeasure (Algorithm 3) uses the same compression step as Rivain-Prouff, but with a different matrix step, called `MatMult` (Algorithm 4), so that the shares  $x_i$  and  $y_j$  are not used multiple times (as when computing the matrix elements  $x_i \cdot y_j$  in Rivain-Prouff). Eventually the `MatMult` algorithm outputs a matrix  $(M_{ij})_{1 \leq i, j \leq n}$  which is still an  $n^2$ -sharing of  $x^* \cdot y^*$ , as in (19); therefore using the same compression step as Rivain-Prouff, Algorithm 3 outputs an  $n$ -sharing of  $x^* \cdot y^*$ , as required.

---

#### Algorithm 4 `MatMult`

---

**Input:** the  $n$ -sharings  $(x_i)_{i \in [1..n]}$  and  $(y_j)_{j \in [1..n]}$  of  $x^*$  and  $y^*$  respectively

**Output:** the  $n^2$ -sharing  $(M_{ij})_{i \in [1..n], j \in [1..n]}$  of  $x^* \cdot y^*$

```

1: if  $n = 1$  then
2:    $M \leftarrow [x_1 \cdot y_1]$ 
3: else
4:    $\mathbf{X}^{(1)} \leftarrow (x_1, \dots, x_{n/2})$ ,  $\mathbf{X}^{(2)} \leftarrow (x_{n/2+1}, \dots, x_n)$ 
5:    $\mathbf{Y}^{(1)} \leftarrow (y_1, \dots, y_{n/2})$ ,  $\mathbf{Y}^{(2)} \leftarrow (y_{n/2+1}, \dots, y_n)$ 
6:    $\mathbf{M}^{(1,1)} \leftarrow \text{MatMult}(\mathbf{X}^{(1)}, \mathbf{Y}^{(1)})$ 
7:    $\mathbf{X}^{(1)} \leftarrow \text{RefreshMasks}(\mathbf{X}^{(1)})$ ,  $\mathbf{Y}^{(1)} \leftarrow \text{RefreshMasks}(\mathbf{Y}^{(1)})$ 
8:    $\mathbf{M}^{(1,2)} \leftarrow \text{MatMult}(\mathbf{X}^{(1)}, \mathbf{Y}^{(2)})$ 
9:    $\mathbf{M}^{(2,1)} \leftarrow \text{MatMult}(\mathbf{X}^{(2)}, \mathbf{Y}^{(1)})$ 
10:   $\mathbf{X}^{(2)} \leftarrow \text{RefreshMasks}(\mathbf{X}^{(2)})$ ,  $\mathbf{Y}^{(2)} \leftarrow \text{RefreshMasks}(\mathbf{Y}^{(2)})$ 
11:   $\mathbf{M}^{(2,2)} \leftarrow \text{MatMult}(\mathbf{X}^{(2)}, \mathbf{Y}^{(2)})$ 
12:   $M \leftarrow \begin{bmatrix} \mathbf{M}^{(1,1)} & \mathbf{M}^{(1,2)} \\ \mathbf{M}^{(2,1)} & \mathbf{M}^{(2,2)} \end{bmatrix}$ 
13: end if
14: return  $M$ 

```

---

As illustrated in Fig. 5, the `MatMult` algorithm is recursive and computes the  $n \times n$  matrix in four sub-matrix blocs. This is done by splitting the input shares  $x_i$  and  $y_j$  in two parts, namely  $\mathbf{X}^{(1)} = (x_1, \dots, x_{n/2})$  and  $\mathbf{X}^{(2)} = (x_{n/2+1}, \dots, x_n)$ , and similarly  $\mathbf{Y}^{(1)} = (y_1, \dots, y_{n/2})$  and  $\mathbf{Y}^{(2)} = (y_{n/2+1}, \dots, y_n)$ , and recursively processing the four sub-matrix blocs corresponding to  $\mathbf{X}^{(u)} \times \mathbf{Y}^{(v)}$  for  $1 \leq u, v \leq 2$ . To prevent the same share  $x_i$  from being used twice, each input block  $\mathbf{X}^{(u)}$  and  $\mathbf{Y}^{(v)}$  is refreshed before being used a second time, using a mask refreshing algorithm. An example of such mask refreshing, hereafter called `RefreshMasks`, can for instance be found in [DDF14]; see Algorithm 5. Since the mask refreshing does not modify the xor of the input  $n/2$ -vectors  $\mathbf{X}^{(u)}$  and  $\mathbf{Y}^{(v)}$ , each sub-matrix block  $\mathbf{M}^{(u,v)}$  is still an  $n^2/4$ -sharing of  $(\oplus \mathbf{X}^{(u)}) \cdot (\oplus \mathbf{Y}^{(v)})$ , and therefore the output matrix  $M$  is still an  $n^2$ -sharing of  $x^* \cdot y^*$ , as required. Note that without the `RefreshMasks`, we would have  $M_{ij} = x_i \cdot y_j$  as in Rivain-Prouff.

Since the `RefreshMasks` algorithm has complexity  $\mathcal{O}(n^2)$ , it is easy to see that the complexity of our `RefSecMult` algorithm is  $\mathcal{O}(n^2 \log n)$  (instead of  $\mathcal{O}(n^2)$  for the original Rivain-Prouff countermeasure in Alg. 1). Therefore for a circuit of size  $|C|$  the complexity is  $\mathcal{O}(|C| \cdot n^2 \log n)$ , instead of  $\mathcal{O}(|C| \cdot n^2)$  for Rivain-Prouff. The following lemma shows the soundness of our `RefSecMult` countermeasure.

**Lemma 1 (Soundness of `RefSecMult`).** *The `RefSecMult` algorithm, on input  $n$ -sharings  $(x_i)_{i \in [1..n]}$  and  $(y_j)_{j \in [1..n]}$  of  $x^*$  and  $y^*$  respectively, outputs an  $n$ -sharing  $(c_i)_{i \in [1..n]}$  of  $x^* \cdot y^*$ .*

---

**Algorithm 5** RefreshMasks
 

---

**Input:**  $a_1, \dots, a_n$ 
**Output:**  $c_1, \dots, c_n$  such that  $\sum_{i=1}^n c_i = \sum_{i=1}^n a_i$ 

```

1: for  $i = 1$  to  $n$  do  $c_i \leftarrow a_i$ 
2: for  $i = 1$  to  $n$  do
3:   for  $j = i + 1$  to  $n$  do
4:      $r \leftarrow \{0, 1\}^k$ 
5:      $c_i \leftarrow c_i + r$ 
6:      $c_j \leftarrow c_j + r$ 
7:   end for
8: end for
9: return  $c_1, \dots, c_n$ 

```

---

*Proof.* We prove recursively that the MatMult algorithm, taking as input  $n$ -sharings  $(x_i)_{i \in [1..n]}$  and  $(y_j)_{j \in [1..n]}$  of  $x^*$  and  $y^*$  respectively, outputs an  $n^2$ -sharing  $M_{ij}$  of  $x^* \cdot y^*$ . The lemma for RefSecMult will follow, since as in Rivain-Prouff the lines 2 to 12 of Alg. 3 transform an  $n^2$ -sharing  $M_{ij}$  of  $x^* \cdot y^*$  into an  $n$ -sharing of  $x^* \cdot y^*$ .

The property clearly holds for  $n = 1$ . Assuming that it holds for  $n/2$ , since the RefreshMasks does not change the xor of the input  $n/2$ -vectors  $\mathbf{X}^{(u)}$  and  $\mathbf{Y}^{(v)}$ , each sub-matrix block  $\mathbf{M}^{(u,v)}$  is still an  $n^2/4$ -sharing of  $(\oplus \mathbf{X}^{(u)}) \cdot (\oplus \mathbf{X}^{(v)})$ , and therefore the output matrix  $\mathbf{M}$  is still an  $n^2$ -sharing of  $x^* \cdot y^*$ , as required. This proves the lemma.  $\square$

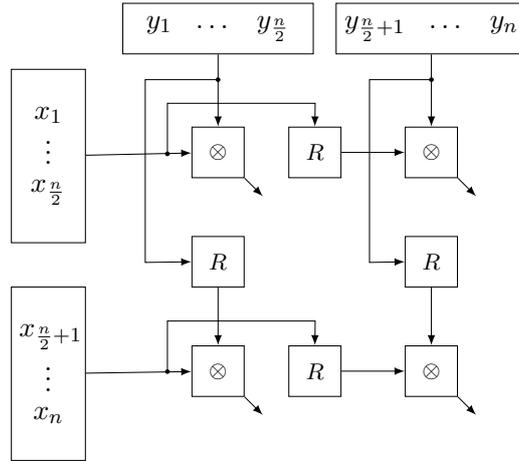


Fig. 5: The recursive MatMult algorithm, where  $R$  represents the RefreshMasks Algorithm, and  $\otimes$  represents a recursive call to the MatMult algorithm.

*Remark 2.* The description of our countermeasure requires that  $n$  is a power of two, but it is easy to modify the countermeasure to handle any value of  $n$ . Namely in Algorithm 4, for odd  $n$  it suffices to split the inputs  $x_i$  and  $y_j$  in two parts of size  $(n - 1)/2$  and  $(n + 1)/2$  respectively, instead of  $n/2$ .

## 8.2 Security Analysis.

**Proven security in the ISW probing model.** We prove that our RefSecMult algorithm achieves at least the same level of security of Rivain-Prouff, namely it is secure in the ISW probing model against  $t$  probes for  $n \geq t + 1$  shares. For this we use the refined security

model against probing attacks recently introduced in [BBD<sup>+</sup>15], called  $t$ -SNI security. This stronger definition of  $t$ -SNI security enables to prove that a gadget can be used in a full construction with  $n \geq t + 1$  shares, instead of  $n \geq 2t + 1$  for the weaker definition of  $t$ -NI security (corresponding to the original ISW security proof). The authors of [BBD<sup>+</sup>15] show that the ISW (and Rivain-Prouff) multiplication gadget does satisfy this stronger  $t$ -SNI security definition. They also show that with some additional mask refreshing satisfying the  $t$ -SNI property (such as `RefreshMasks`), the Rivain-Prouff countermeasure for the full AES can be made secure with  $n \geq t + 1$  shares.

The following lemma shows that our `RefSecMult` countermeasure achieves the  $t$ -SNI property; we provide the proof in Appendix B. The proof is essentially the same as in [BBD<sup>+</sup>15] for the Rivain-Prouff countermeasure; namely the compression step is the same, and for the matrix step, in the simulation we can assume that all the randoms in `RefreshMasks` are given to the adversary. The  $t$ -SNI security implies that our `RefSecMult` algorithm is also composable, with  $n \geq t + 1$  shares.

**Lemma 2 ( $t$ -SNI of `RefSecMult`).** *Let  $(x_i)_{1 \leq i \leq n}$  and  $(y_i)_{1 \leq i \leq n}$  be the input shares of the `SecMult` operation, and let  $(c_i)_{1 \leq i \leq n}$  be the output shares. For any set of  $t_1$  intermediate variables and any subset  $|\mathcal{O}| \leq t_2$  of output shares such that  $t_1 + t_2 < n$ , there exist two subsets  $I$  and  $J$  of indices with  $|I| \leq t_1$  and  $|J| \leq t_1$ , such that those  $t_1$  intermediate variables as well as the output shares  $c_{|\mathcal{O}}$  can be perfectly simulated from  $x_{|I}$  and  $y_{|J}$ .*

**Heuristic security against horizontal-DPA attacks.** We stress that the previous lemma only proves the security of our countermeasure against  $t$  probes for  $n \geq t + 1$ , so it does not prove that our countermeasure is secure against the horizontal-DPA attacks described in previous sections, since such attacks use information about  $n^2$  intermediate variables instead of only  $n - 1$ .

As illustrated in Fig. 5, the main difference between the new `RefSecMult` algorithm and the original `SecMult` algorithm (Alg. 1) is that we keep refreshing the  $x_i$  shares and the  $y_j$  shares blockwise between the processing of the finite field multiplications  $x_i \cdot y_j$ . Therefore, as opposed to what happens in `SecMult`, we never have the same  $x_i$  being multiplied by all  $y_j$ 's for  $1 \leq j \leq n$ . Therefore an attacker cannot accumulate information about a specific share  $x_i$ , which heuristically prevents the attacks described in this paper.

## 9 Mask Refreshing with Complexity $\mathcal{O}(n \cdot \log n)$

In this section we describe a new mask refreshing algorithm with complexity  $\mathcal{O}(n \cdot \log n)$  instead of  $\mathcal{O}(n^2)$  for the classical algorithm recalled in previous section. For simplicity we first assume that  $n$  is a power of 2; we refer to Appendix C for a generalization to arbitrary  $n$ . Note that a completely different mask refreshing algorithm is described in [ADF16], with complexity only  $\mathcal{O}(n)$ ; however our algorithm is simpler.

As illustrated in Fig. 6, our algorithm is defined recursively. First, a pre-processing layer  $L_I$  is applied on the  $n$  input shares  $a_i$ , corresponding to lines 5 to 9 of Algorithm 6 below. Then the `RefreshMasks` algorithm is applied recursively on the two halves of the shares. Eventually a post-processing layer  $L_O$  is applied (same as  $L_I$ ), before outputting the output shares  $d_i$  (it corresponds to Lines 12 to 16 in Alg. 6).

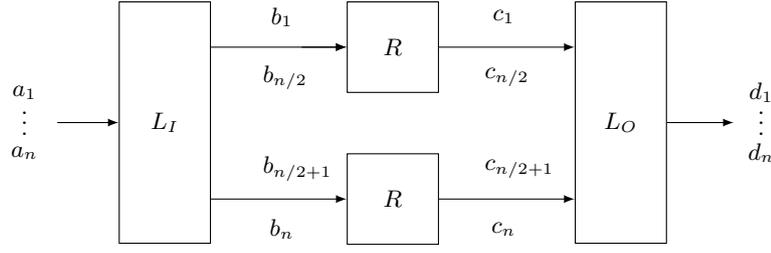


Fig. 6: Recursive definition of RefreshMasks, with the pre-processing layer  $L_I$ , the two recursive applications of RefreshMasks to the two halves of the shares, and the post-processing layer  $L_O$ .

---

### Algorithm 6 RefreshMasks

---

**Input:**  $a_1, \dots, a_n$

**Output:**  $d_1, \dots, d_n$  such that  $\bigoplus_{i=1}^n d_i = \bigoplus_{i=1}^n a_i$

```

1: if  $n = 2$  then
2:    $r \leftarrow^{\$} \{0, 1\}^k$ 
3:   return  $(a_1 \oplus r, a_2 \oplus r)$ 
4: end if
5: for  $i = 1$  to  $n/2$  do
6:    $r_i \leftarrow^{\$} \{0, 1\}^k$ 
7:    $b_i \leftarrow a_i \oplus r_i$ 
8:    $b_{n/2+i} \leftarrow a_{n/2+i} \oplus r_i$            #  $b_i \oplus b_{n/2+i} = a_i \oplus a_{n/2+i}$ 
9: end for
10:  $(c_1, \dots, c_{n/2}) \leftarrow \text{RefreshMasks}(b_1, \dots, b_{n/2})$ 
11:  $(c_{n/2+1}, \dots, c_n) \leftarrow \text{RefreshMasks}(b_{n/2+1}, \dots, b_n)$ 
12: for  $i = 1$  to  $n/2$  do
13:    $r_i \leftarrow^{\$} \{0, 1\}^k$ 
14:    $d_i \leftarrow c_i \oplus r_i$ 
15:    $d_{n/2+i} \leftarrow c_{n/2+i} \oplus r_i$        #  $d_i \oplus d_{n/2+i} = c_i \oplus c_{n/2+i}$ 
16: end for
17: return  $(d_1, \dots, d_n)$ 

```

---

**Correctness.** The correctness of Algorithm 6 is straightforward to verify recursively. Namely for all 4 blocks in Fig. 6, the xor of the outputs is the same as the xor of the inputs. Therefore globally we must have  $\bigoplus_{i=1}^n d_i = \bigoplus_{i=1}^n a_i$ .

**Complexity.** Let  $T(n)$  be the complexity of RefreshMasks. We have  $T(n) \leq 2 \cdot T(n/2) + c \cdot n$  for some constant  $c$ . One can show recursively that  $T(n) \leq 2^i \cdot T(n/2^i) + i \cdot c \cdot n$ , which implies  $T(n) = \mathcal{O}(n \cdot \log n)$ .

**Security.** The following lemma shows that our new RefreshMasks algorithm achieves the same property as the previous RefreshMasks, namely  $t$ -SNI security.

**Lemma 3 ( $t$ -SNI of RefreshMasks).** *Let  $(a_i)_{1 \leq i \leq n}$  be the input shares of the RefreshMasks algorithm, and let  $(d_i)_{1 \leq i \leq n}$  be the output shares. For any set of  $t$  intermediate variables and any subset  $|O| \leq t_O$  of output shares such that  $t + t_O < n$ , there exists a subset  $I$  of indices with  $|I| \leq t$ , such that the distribution of those  $t$  intermediate variables as well as the output shares  $d_{|O}$  can be perfectly simulated from  $a_{|I}$ .*

*Proof.* The proof is based on the two following simple lemmas.

**Lemma 4.** Let  $a_1, a_2 \in \{0, 1\}^k$  be inputs, and let  $r \leftarrow^{\$} \{0, 1\}^k$ . Let  $\mathcal{V}$  be a subset of the variables  $\{a_1, a_2, r\}$  and let  $\mathcal{O}$  be a subset of the variables  $\{a_1 \oplus r, a_2 \oplus r\}$ . Then the variables in  $\mathcal{V} \cup \mathcal{O}$  can be perfectly simulated from  $I_1 \in \{\emptyset, \{a_1\}\}$  and  $I_2 \in \{\emptyset, \{a_2\}\}$ , with  $|I_1| \leq t_1 + |\mathcal{O}|/2$  and  $|I_2| \leq t_2 + |\mathcal{O}|/2$ , for some positive integers  $t_1, t_2$ , with  $t_1 + t_2 \leq |\mathcal{V}|$ .

*Proof.* If  $|\mathcal{O}| = 2$  or  $|\mathcal{V}| \geq 2$  then we can let  $I_1 = \{a_1\}$  and  $I_2 = \{a_2\}$ . If  $|\mathcal{O}| = 1$  and  $|\mathcal{V}| = 0$  then the variable in  $\mathcal{O}$  can be perfectly simulated by a random value. If  $|\mathcal{O}| = 1$  and  $|\mathcal{V}| = 1$ , assume without loss of generality that  $\mathcal{O} = \{a_1 \oplus r\}$ ; then if  $\mathcal{V} = \{a_1\}$  or  $\mathcal{V} = \{r\}$ , we let  $I_1 = \{a_1\}$ ; if  $\mathcal{V} = \{a_2\}$ , we let  $I_2 = \{a_2\}$ , and we can perfectly simulate  $a_1 \oplus r$  with a random value. If  $|\mathcal{O}| = 0$  then the simulation is straightforward.  $\square$

**Lemma 5.** Let  $a_1, a_2 \in \{0, 1\}^k$  be inputs, and let  $r \leftarrow^{\$} \{0, 1\}^k$ . Let  $\mathcal{V}$  be a subset of the variables  $\{a_1, a_2, r\}$  and  $\mathcal{O} \in \{\emptyset, \{a_1 \oplus r\}\}$ . Then the variables in  $\mathcal{V} \cup \mathcal{O} \cup \{a_2 \oplus r\}$  can be perfectly simulated from  $I \subset \{a_1, a_2\}$ , with  $|I| \leq 2 \cdot |\mathcal{O}| + |\mathcal{V}|$ .

*Proof.* If  $|\mathcal{O}| = 1$  or  $|\mathcal{V}| \geq 2$  we can take  $I = \{a_1, a_2\}$ . If  $|\mathcal{O}| = 0$  and  $|\mathcal{V}| = 0$ , we can simulate  $a_2 \oplus r$  with a random value. If  $|\mathcal{O}| = 0$  and  $|\mathcal{V}| = 1$ , if  $\mathcal{V} = \{a_1\}$  we let  $I = \{a_1\}$  and we can again simulate  $a_2 \oplus r$  with a random value; if  $\mathcal{V} = \{r\}$  or  $\mathcal{V} = \{a_2\}$  then we let  $I = \{a_2\}$ .  $\square$

We proceed with the proof of Lemma 3, by recurrence on the number  $n$  of input shares. The case  $n = 2$  is straightforward, and similar to Lemma 4. Namely if  $t_{\mathcal{O}} = 1$  then  $t = 0$  and thanks to the random mask  $r$  at Line 2 we can simulate the corresponding output variable with a random value; if  $t = 1$  then  $t_{\mathcal{O}} = 0$  and the simulation of the probed intermediate variable is straightforward.

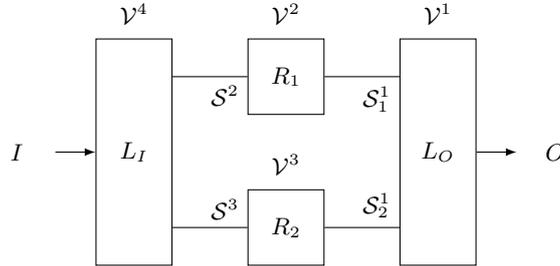


Fig. 7: RefreshMasks as composition of gadgets

We now consider the algorithm with  $n$  shares. We label the gadgets from 1 to 4 starting from right to left, with  $\mathcal{V}^i$  the corresponding probed intermediate variables (see Fig. 7). We start with the rightmost gadget  $L_O$ , with  $\mathcal{V}^1$  the corresponding set of probed intermediate variables. For simplicity we can assume that within  $\mathcal{V}^1$  only the intermediate variables  $r_i$ ,  $c_i$  and  $c_{n/2+i}$  are probed, and not the output variables  $c_i \oplus r_i$  and  $c_{n/2+i} \oplus r_i$ , since such output variables can be equivalently obtained from  $\mathcal{O}$ , for a smaller value of  $t$ , and therefore a stronger bound for  $|I|$ .

By applying Lemma 4 on each set of intermediate variables  $\{c_i, c_{n/2+i}, r_i\}$  and output variables  $\{c_i \oplus r_i, c_{n/2+i} \oplus r_i\}$  for all  $1 \leq i \leq n/2$ , and summing the inequalities, we obtain that the gadget  $L_O$  can be perfectly simulated from two sets of input indices  $\mathcal{S}_1^1 \subset \{1, \dots, n/2\}$  and  $\mathcal{S}_2^1 \subset \{n/2 + 1, \dots, n\}$ , such that

$$|\mathcal{S}_1^1| \leq t_1 + t_{\mathcal{O}}/2, \quad |\mathcal{S}_2^1| \leq t_2 + t_{\mathcal{O}}/2 \quad (20)$$

for some positive integers  $t_1, t_2$ , with  $t_1 + t_2 \leq |\mathcal{V}^1|$ .

We now consider the  $R_1$  and  $R_2$  gadgets, which are recursive applications of RefreshMasks with  $n/2$  shares. The  $t$ -SNI condition  $t + t_O < n$  for applying Lemma 3 on  $R_1$  is therefore:

$$|\mathcal{V}^2| + |\mathcal{S}_1^1| < n/2 \quad (21)$$

and when such condition is satisfied we have from Lemma 3 that the probed intermediate variables in  $\mathcal{V}^2$  and the output variables in  $\mathcal{S}_1^1$  can be simulated from a subset of input indices  $\mathcal{S}^2$  such that  $|\mathcal{S}^2| \leq |\mathcal{V}^2|$ . Similarly the  $t$ -SNI condition for gadget  $R_2$  is:

$$|\mathcal{V}^3| + |\mathcal{S}_2^1| < n/2 \quad (22)$$

When such condition is not satisfied for  $R_1$  or  $R_2$  we say that the corresponding gadget is *saturated*. In that case, to properly simulate the probed intermediate variables and output variables of the gadget, one must know *all* inputs of the gadget. As will be seen below, the proof of Lemma 3 is based on the fact that the two gadgets  $R_1$  and  $R_2$  cannot be both saturated, that is at least  $R_1$  or  $R_2$  must be non-saturated.

Namely consider the following two inequalities:

$$|\mathcal{V}^2| + t_1 + t_O/2 < n/2 \quad (23)$$

$$|\mathcal{V}^3| + t_2 + t_O/2 < n/2 \quad (24)$$

Using  $|\mathcal{S}_1^1| \leq t_1 + t_O/2$ , we have that Inequality (23) implies Condition (21) for gadget  $R_1$ ; similarly using  $|\mathcal{S}_2^1| \leq t_2 + t_O/2$ , we have that Inequality (24) implies Condition (22) for gadget  $R_2$ . Moreover, at least one of the two inequalities (23) or (24) must be satisfied, since otherwise, using  $t_1 + t_2 \leq |\mathcal{V}^1|$ :

$$n \leq |\mathcal{V}^2| + t_1 + t_O/2 + |\mathcal{V}^3| + t_2 + t_O/2 \leq |\mathcal{V}^2| + |\mathcal{V}^3| + |\mathcal{V}^1| + t_O \leq t + t_O$$

which contradicts the bound  $t + t_O < n$ . As mentioned previously, this shows that the gadgets  $R_1$  and  $R_2$  cannot be both saturated.

If both inequalities (23) and (24) are satisfied, then both gadgets  $R_1$  and  $R_2$  are non-saturated, and by recursively applying Lemma 3 on both gadgets, we get  $|\mathcal{S}^2| \leq |\mathcal{V}^2|$  and  $|\mathcal{S}^3| \leq |\mathcal{V}^3|$ . One can therefore let  $I = \mathcal{S}^2 \cup \mathcal{S}^3$  and simulate the  $L_I$  gadget as in the real circuit<sup>3</sup>; we then have as required:

$$|I| \leq |\mathcal{S}^2| + |\mathcal{S}^3| \leq |\mathcal{V}^2| + |\mathcal{V}^3| \leq t$$

Assume now wlog that (23) is satisfied and (24) is not. Then  $R_1$  is non-saturated and we can apply Lemma 3 on  $R_1$ , which gives as previously  $|\mathcal{S}^2| \leq |\mathcal{V}^2|$ . For  $R_2$  we cannot necessarily apply Lemma 3, so we must take  $\mathcal{S}^3 = \{n/2 + 1, \dots, n\}$ , which means that all inputs of  $R_2$  must be simulated. We now consider the  $L_I$  gadget. We can assume wlog that all intermediate variables probed in  $\mathcal{V}^4$  are of the form  $a_i$ ,  $a_{n/2+i}$  and  $r_i$ , since all output variables  $a_i \oplus r_i$  and  $a_{n/2+i} \oplus r_i$  can be probed in  $\mathcal{V}^2$  and  $\mathcal{V}^3$ . By applying Lemma 5 for all  $1 \leq i \leq n/2$  on each set of intermediate variables  $\{a_i, a_{n/2+i}, r_i\}$  and output variable  $a_i \oplus r_i$ , where all output variables  $a_{n/2+i} \oplus r_i$  must be simulated (since  $R_2$  is saturated), and by summing the inequalities, we construct  $I \subset \{1, \dots, n\}$  such that:

$$|I| \leq 2 \cdot |\mathcal{S}^2| + |\mathcal{V}^4| \leq 2 \cdot |\mathcal{V}^2| + |\mathcal{V}^4| \quad (25)$$

It remains to show that  $|I| \leq t$ . Since by assumption (23) is satisfied and (24) is not, we have:

$$|\mathcal{V}^2| + t_1 + t_O/2 < n/2 \leq |\mathcal{V}^3| + t_2 + t_O/2$$

<sup>3</sup> When both  $R_1$  and  $R_2$  are non-saturated, the  $L_I$  gadget is actually useless.

which gives using  $t_2 \leq |\mathcal{V}^1|$ :

$$|\mathcal{V}^2| \leq |\mathcal{V}^3| + t_2 \leq |\mathcal{V}^3| + |\mathcal{V}^1|$$

Then from (25) we obtain:

$$|I| \leq 2 \cdot |\mathcal{V}^2| + |\mathcal{V}^4| \leq |\mathcal{V}^2| + |\mathcal{V}^3| + |\mathcal{V}^1| + |\mathcal{V}^4| \leq t$$

as required, which terminates the proof of Lemma 3.  $\square$

*Remark 3.* One can argue that our construction is somewhat minimal. Namely if we remove the pre-processing layer  $L_I$ , then the adversary can let  $\mathcal{O} = \{1, \dots, n/4\} \cup \{n/2+1, \dots, 3n/4\}$ , which implies that the first  $n/4$  output variables of  $R_1$  must be simulated; the adversary can then probe the remaining  $n/4$  output variables of  $R_1$ , which implies that  $R_1$  is now saturated. Then without  $L_I$  one must have  $|I| = n/2$ ; with  $|\mathcal{V}| = n/4$  this contradicts the bound  $|I| \leq |\mathcal{V}|$ , while with  $|\mathcal{O}| = n/2$  the condition  $|\mathcal{V}| + |\mathcal{O}| < n$  is still satisfied.

Similarly, if we remove the post-processing layer  $L_O$ , then the adversary can saturate  $R_1$  by letting  $\mathcal{O} = \{1, \dots, n/2\}$ . Then all inputs of  $R_1$  must be known, and for each intermediate probe at the input of  $R_2$ , one must add two additional indices in  $I$ , which contradicts the bound  $|I| \leq t$ .

*Remark 4.* Using the above RefreshMasks algorithm for our countermeasure of Section 8, its complexity becomes  $\mathcal{O}(n^2)$  instead of  $\mathcal{O}(n^2 \cdot \log n)$ , hence the same complexity as the original ISW countermeasure.

**Acknowledgments.** We are very grateful to the anonymous CHES reviewers for pointing a flaw in a previous version of our countermeasure in Section 8.

## References

- [ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with  $\mathcal{O}(1/\log n)$  leakage rate. In *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*, pages 586–615, 2016.
- [BBD<sup>+</sup>15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional verification of higher-order masking: Application to a verifying masking compiler. Cryptology ePrint Archive, Report 2015/506, 2015. <http://eprint.iacr.org/>.
- [BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlich. Inner product masking revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 486–510. Springer, 2015.
- [BJPW13] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and vertical side-channel attacks against secure RSA implementations. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco, CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.
- [Bla79] G.R. Blakely. Safeguarding cryptographic keys. In *National Comp. Conf.*, volume 48, pages 313–317, New York, June 1979. AFIPS Press.
- [CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.
- [CFG<sup>+</sup>10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihang Qing, and Javier López, editors, *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2010.

- [CGP<sup>+</sup>12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.
- [CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 398–412, 1999.
- [Cor14] Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.
- [CP00] M.A. Carreira-Perpinan. Mode-finding for mixtures of Gaussian distributions Carreira-Perpinan. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 22(11):1318–1323, November 2000.
- [CRV15] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. *J. Cryptographic Engineering*, 5(2):73–83, 2015.
- [DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 423–440, 2014.
- [DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 401–429, 2015.
- [FRR<sup>+</sup>10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 135–156, 2010.
- [GGP<sup>+</sup>15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015.
- [GHR15] Sylvain Guilley, Annelie Heuser, and Olivier Rioul. A key to success - success exponents for side-channel distinguishers. In Alex Biryukov and Vipul Goyal, editors, *Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings*, volume 9462 of *Lecture Notes in Computer Science*, pages 270–290. Springer, 2015.
- [GPQ11] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In *CHES*, pages 240–255, 2011.
- [GPS14] Vincent Grosso, Emmanuel Prouff, and François-Xavier Standaert. Efficient masked s-boxes processing - A step forward -. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2014.
- [ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 463–481, 2003.
- [MOOS15] Daniel P. Martin, Jonathan F. O’Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015.
- [ORSW12] Yossef Oren, Mathieu Renaud, François-Xavier Standaert, and Avishai Wool. Algebraic side-channel attacks beyond the hamming weight leakage model. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th Inter-*

- national Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2012.
- [PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 63–78, 2011.
- [PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 142–159, 2013.
- [RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 413–427, 2010.
- [Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.
- [SVO<sup>+</sup>10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlich, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.
- [VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

## A Mutual Information Approximation

In this section, we develop the mutual information between the Hamming weight of uniform random variable  $X$  defined over  $\mathbb{F}_{2^k}$  and a noisy observation  $L$  of this Hamming weight, the noise being modeled by a Gaussian random variable with 0 mean and standard deviation  $\sigma$ . By definition, we have  $L = \text{HW}(X) + B$ , with  $B \sim \mathcal{N}(0, \sigma^2)$ . By definition of the entropy, we have:

$$\mathbb{H}(X|L) = \mathbb{H}(X, L) - \mathbb{H}(L) . \quad (26)$$

Given  $X$ , the distribution of  $L$  is Gaussian with standard deviation  $\sigma$ . The differential entropy corresponding to a Gaussian distribution with standard deviation  $\sigma$  is  $\ln(\sigma\sqrt{2\pi e})$ , whereas the entropy of a uniform random variable over  $\mathbb{F}_{2^k}$  is  $k$ . We therefore get:

$$\mathbb{H}(X, L) = \mathbb{H}(X) + \mathbb{H}(B) = k + \ln(\sigma\sqrt{2\pi e}) \quad (27)$$

The distribution of  $L$  is the sum of two distributions. We can model the distribution of  $\text{HW}(X)$  as Gaussian with standard deviation  $\sigma_k = \sqrt{k}/2$ ; this is true for large  $k$ . The sum of two independent and normally distributed random variables has a normal distribution. Moreover, its mean and variance are simply obtained by summing those of the added variables. In our context, this gives:

$$\mathbb{H}(L) \simeq \ln \left( \sqrt{(\sigma_k^2 + \sigma^2)2\pi e} \right) \simeq \ln \left( \sqrt{(k/4 + \sigma^2)2\pi e} \right) \quad (28)$$

Combining (26)-(28) leads to:

$$\begin{aligned} \mathbb{H}(X | L) &\simeq k + \ln(\sigma\sqrt{2\pi e}) - \ln \left( \sqrt{(k/4 + \sigma^2)2\pi e} \right) \simeq k + \frac{1}{2} \ln \left( \frac{\sigma^2}{k/4 + \sigma^2} \right) \\ &\simeq k + \frac{1}{2} \ln \left( \frac{1}{1 + k/(4\sigma^2)} \right) \simeq k - \frac{k}{8\sigma^2} , \end{aligned}$$

after approximating  $\ln(1-U)$  by  $-U$ , and  $k/(k+4\sigma^2)$  by  $k/4\sigma^2$  (which is true when  $\sigma^2 \gg k$ ). Eventually, the amount of information  $I(X;L) \doteq \mathbb{H}(X) - \mathbb{H}(X | L)$  given by the noisy Hamming weight leakage can be approximated for  $\sigma^2 \gg k$  by:

$$I(X;L) \simeq \frac{k}{8\sigma^2}$$

## B Proof of Lemma 2

Our proof is essentially the same as in [BBD<sup>+</sup>15]. We construct two sets  $I$  and  $J$  corresponding to the input shares of  $x^*$  and  $y^*$  respectively. We denote by  $M_{ij}$  the result of the subroutine  $\text{MatMult}((x_1, \dots, x_n), (y_1, \dots, y_n))$ . From the definition of  $\text{MatMult}$  and  $\text{RefreshMasks}$ , it is easy to see that each  $M_{ij}$  can be perfectly simulated from  $x_i$  and  $y_j$ ; more generally any internal variable within  $\text{MatMult}$  can be perfectly simulated from  $x_i$  and/or  $y_j$  for some  $i$  and  $j$ ; for this it suffices to simulate the randoms in  $\text{RefreshMasks}$  exactly as they are generated in  $\text{RefreshMasks}$ .

We divide the internal probes in 4 groups. The four groups are processed separately and sequentially, that is we start with all probes in Group 1, and finish with all probes in Group 4.

- Group 1: If  $M_{ii}$  is probed, add  $i$  to  $I$  and  $J$ .
- Group 2: If  $r_{i,j}$  or  $c_{i,j}$  is probed (for any  $i \neq j$ ), add  $i$  to  $I$  and  $J$ .

Note that after the processing of Group 1 and 2 probes, we have  $I = J$ ; we denote by  $U$  the common value of  $I$  and  $J$  after the processing of Group 1 and 2 probes.

- Group 3: If  $M_{ij} \oplus r_{i,j}$  is probed: if  $i \in U$  or  $j \in U$ , add  $\{i, j\}$  to both  $I$  and  $J$ .
- Group 4: If  $M_{ij}$  is probed (for any  $i \neq j$ ), then add  $i$  to  $I$  and  $j$  to  $J$ . If some probe in  $\text{MatMult}$  requires the knowledge of  $x_i$  and/or  $y_j$ , add  $i$  to  $I$  and/or  $j$  to  $J$ .

We have  $|I| \leq t_1$  and  $|J| \leq t_1$ , since for every probe we add at most one index in  $I$  and  $J$ . The simulation of probed variables in groups 1 and 4 is straightforward. Note that for  $i < j$ , the variable  $r_{ij}$  is used in all partial sums  $c_{ik}$  for  $k \geq j$ ; moreover  $r_{ij}$  is used in  $r_{ij} \oplus M_{ij}$ , which is used in  $r_{ji}$ , which is used in all partial sums  $c_{jk}$  for  $k \geq i$ . Therefore if  $i \notin U$ , then  $r_{ij}$  is not probed and does not enter in the computation of any probed  $c_{ik}$ ; symmetrically if  $j \notin U$ , then  $r_{ji}$  is not probed and does not enter in the computation of any probed  $c_{jk}$ .

For any pair  $i < j$ , we can now distinguish 4 cases:

- Case 1:  $\{i, j\} \in U$ . In that case, we can perfectly simulate all variables  $r_{ij}$ ,  $M_{ij}$ ,  $M_{ij} \oplus r_{ij}$ ,  $M_{ji}$  and  $r_{ji}$ . In particular, we let  $r_{ij} \leftarrow \mathbb{F}_{2^k}$ , as in the real circuit.
- Case 2:  $i \in U$  and  $j \notin U$ . In that case we simulate  $r_{ij} \leftarrow \mathbb{F}_{2^k}$ , as in the real circuit. If  $M_{ij} \oplus r_{i,j}$  is probed (Group 3), we can also simulate it since  $i \in U$  and  $j \in J$  by definition of the processing of Group 3 variables.
- Case 3:  $i \notin U$  and  $j \in U$ . In that case  $r_{ij}$  has not been probed, nor any variable  $c_{ik}$ , since otherwise  $i \in U$ . Therefore  $r_{ij}$  is not used in the computation of any probed variable (except  $r_{ji}$ , and possibly  $M_{ij} \oplus r_{i,j}$ ). Therefore we can simulate  $r_{ji} \leftarrow \mathbb{F}_{2^k}$ ; moreover if  $M_{ij} \oplus r_{i,j}$  is probed, we can perfectly simulate it using  $M_{ij} \oplus r_{i,j} = M_{ji} \oplus r_{ji}$ , since  $j \in U$  and  $i \in J$  by definition of the processing of Group 3 variables.
- Case 4:  $i \notin U$  and  $j \notin U$ . If  $M_{ij} \oplus r_{i,j}$  is probed, since  $r_{ij}$  is not probed and does not enter into the computation of any other probed variable, we can perfectly simulate such probe with a random value.

From cases 1, 2 and 3, we obtain that for any  $i \neq j$ , we can perfectly simulate any variable  $r_{ij}$  such that  $i \in U$ . This implies that we can also perfectly simulate all partial sums  $c_{ik}$  when  $i \in U$ , including the output variables  $c_i$ . Finally, all probed variables are perfectly simulated.

We now consider the simulation of the output variables  $c_i$ . We must show how to simulate  $c_i$  for all  $i \in \mathcal{O}$ , where  $\mathcal{O}$  is an arbitrary subset of  $[1, n]$  such that  $t_1 + |\mathcal{O}| < n$ . For  $i \in U$ , such variables are already perfectly simulated, as explained above. We now consider the output variables  $c_i$  with  $i \notin U$ . We construct a subset of indices  $V$  as follows: for any probed Group 3 variable  $M_{ij} \oplus r_{ij}$  such that  $i \notin U$  and  $j \notin U$  (this corresponds to Case 4), we put  $j$  in  $V$  if  $i \in \mathcal{O}$ , otherwise we put  $i$  in  $V$ . Since we have only considered Group 3 probes, we must have  $|U| + |V| \leq t_1$ , which implies  $|U| + |V| + |\mathcal{O}| < n$ . Therefore there exists an index  $j^* \in [1, n]$  such that  $j^* \notin U \cup V \cup \mathcal{O}$ . For any  $i \in \mathcal{O}$ , we can write:

$$c_i = M_{ii} \oplus \bigoplus_{j \neq i} r_{ij} = r_{i,j^*} \oplus \left( M_{ii} \oplus \bigoplus_{j \neq i, j^*} r_{ij} \right)$$

We claim that neither  $r_{i,j^*}$  nor  $r_{j^*,i}$  do enter into the computation of any probed variable or other  $c_{i'}$  for  $i' \in \mathcal{O}$ . Namely  $i \notin U$  so neither  $r_{i,j^*}$  nor any partial sum  $c_{ik}$  was probed; similarly  $j^* \notin U$  so neither  $r_{j^*,i}$  nor any partial sum  $c_{j^*,k}$  was probed, and the output  $c_{j^*}$  does not have to be simulated since by definition  $j^* \notin \mathcal{O}$ . Finally if  $i < j^*$  then  $M_{i,j^*} \oplus r_{i,j^*}$  was not probed since otherwise  $j^* \in V$  (since  $i \in \mathcal{O}$ ); similarly if  $j^* < i$  then  $M_{j^*,i} \oplus r_{j^*,i}$  was not probed since otherwise we would have  $j^* \in V$  since  $j^* \notin \mathcal{O}$ . Therefore, since neither  $r_{i,j^*}$  nor  $r_{j^*,i}$  are used elsewhere, we can perfectly simulate  $c_i$  by generating a random value. This proves the Lemma.

## C Generalization of Mask Refreshing for Arbitrary $n$

In this section we describe the generalization of our new mask refreshing algorithm from Section 9 to arbitrary  $n$ . We note that to ensure the  $t$ -SNI property such generalization is not completely straightforward; for example if at lines 16 and 17 of Algorithm 7 below we would replace  $\lfloor n/2 \rfloor$  by  $\lceil n/2 \rceil$ , then for  $n = 3$  we would have  $d_3 = a_3$  and the algorithm could not be  $t$ -SNI.

---

**Algorithm 7** RefreshMasks
 

---

**Input:**  $a_1, \dots, a_n$   
**Output:**  $d_1, \dots, d_n$  such that  $\bigoplus_{i=1}^n d_i = \bigoplus_{i=1}^n a_i$

- 1: **if**  $n = 1$  **then**
- 2:   **return**  $(a_1)$
- 3: **end if**
- 4: **if**  $n = 2$  **then**
- 5:    $r \leftarrow^{\$} \{0, 1\}^k$
- 6:   **return**  $(a_1 \oplus r, a_2 \oplus r)$
- 7: **end if**
- 8: **for**  $i = 1$  **to**  $\lfloor n/2 \rfloor$  **do**
- 9:    $r_i \leftarrow^{\$} \{0, 1\}^k$
- 10:    $b_i \leftarrow a_i \oplus r_i$
- 11:    $b_{\lfloor n/2 \rfloor + i} \leftarrow a_{\lfloor n/2 \rfloor + i} \oplus r_i$                        $\# b_i \oplus b_{\lfloor n/2 \rfloor + i} = a_i \oplus a_{\lfloor n/2 \rfloor + i}$
- 12: **end for**
- 13: **if**  $n \bmod 2 = 1$  **then**
- 14:    $b_n \leftarrow a_n$
- 15: **end if**
- 16:  $(c_1, \dots, c_{\lfloor n/2 \rfloor}) \leftarrow \text{RefreshMasks}(b_1, \dots, b_{\lfloor n/2 \rfloor})$
- 17:  $(c_{\lfloor n/2 \rfloor + 1}, \dots, c_n) \leftarrow \text{RefreshMasks}(b_{\lfloor n/2 \rfloor + 1}, \dots, b_n)$
- 18: **for**  $i = 1$  **to**  $\lfloor n/2 \rfloor$  **do**
- 19:    $r_i \leftarrow^{\$} \{0, 1\}^k$
- 20:    $d_i \leftarrow c_i \oplus r_i$
- 21:    $d_{\lfloor n/2 \rfloor + i} \leftarrow c_{\lfloor n/2 \rfloor + i} \oplus r_i$                        $\# d_i \oplus d_{\lfloor n/2 \rfloor + i} = c_i \oplus c_{\lfloor n/2 \rfloor + i}$
- 22: **end for**
- 23: **if**  $n \bmod 2 = 1$  **then**
- 24:    $d_n \leftarrow c_n$
- 25: **end if**
- 26: **return**  $(d_1, \dots, d_n)$

---

The following Lemma shows that our new RefreshMasks algorithm is still  $t$ -SNI for arbitrary  $n$ .

**Lemma 6** ( *$t$ -SNI of RefreshMasks*). *Let  $(a_i)_{1 \leq i \leq n}$  be the input shares of the RefreshMasks operation, and let  $(d_i)_{1 \leq i \leq n}$  be the output shares. For any set of  $t$  intermediate variables and any subset  $|O| \leq t_O$  of output shares such that  $t + t_O < n$ , there exists a subset  $I$  of indices with  $|I| \leq t$ , such that the distribution of those  $t$  intermediate variables as well as the output shares  $d_{|O|}$  can be perfectly simulated from  $a_{|I|}$ .*

*Proof.* We show how to adapt the proof of Lemma 3 to arbitrary  $n$ . The case  $n = 2$  is the same as previously. We now consider the algorithm with  $n \geq 3$  shares.

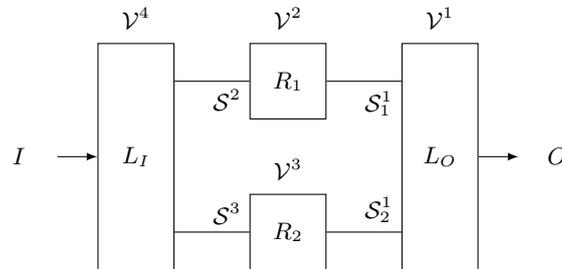


Fig. 8: RefreshMasks as composition of gadgets

As previously we start with the  $L_O$  gadget. Note that for even  $n$  the  $L_O$  gadget is the same as in the proof of Lemma 3, but for odd  $n$  there is a special wire corresponding to

$d_n = c_n$  We show that for the input indices  $\mathcal{S}_1^1$  and  $\mathcal{S}_2^1$  required to simulate  $L_O$ , we always have:

$$|\mathcal{S}_1^1| \leq t_1 + \lceil t_O/2 \rceil, \quad |\mathcal{S}_2^1| \leq t_2 + \lceil t_O/2 \rceil \quad (29)$$

for some positive integers  $t_1, t_2$ , with  $t_1 + t_2 \leq |\mathcal{V}^1|$ ; this is an adaptation of (20) for arbitrary  $n$ . As previously we assume that within  $\mathcal{V}^1$  only the intermediate variables  $r_i, c_i$  and  $c_{\lfloor n/2 \rfloor + i}$  are probed, and not the output variables  $c_i \oplus r_i$  and  $c_{\lfloor n/2 \rfloor + i} \oplus r_i$ , since such output variables can be equivalently obtained from  $\mathcal{O}$ ; similarly if  $n \bmod 2 = 1$ , we assume that  $\mathcal{V}^1$  does not contain the variable  $d_n = c_n$ .

We first consider the  $L_O$  gadget without the wire  $d_n = c_n$  when  $n \bmod 2 = 1$ , which we denote by  $L'_O$ ; we denote by  $\mathcal{O}'$  the corresponding set of output variables that must be simulated, and  $t'_O = |\mathcal{O}'|$ . For  $n = 0 \bmod 2$  we have  $L'_O = L_O$  and  $t'_O = t_O$ . We can then apply Lemma 4 as previously for all  $1 \leq i \leq \lfloor n/2 \rfloor$  on each set of intermediate variables  $\{c_i, c_{\lfloor n/2 \rfloor + i}, r_i\}$  and output variables  $\{c_i \oplus r_i, c_{\lfloor n/2 \rfloor + i} \oplus r_i\}$ ; summing the inequalities, we obtain that the gadget  $L'_O$  can be perfectly simulated from two sets of input indices  $\mathcal{S}_1^1 \subset \{1, \dots, \lfloor n/2 \rfloor\}$  and  $\mathcal{S}'^1_2 \subset \{\lfloor n/2 \rfloor + 1, \dots, 2\lfloor n/2 \rfloor\}$ , such that  $|\mathcal{S}_1^1| \leq t_1 + t'_O/2$  and  $|\mathcal{S}'^1_2| \leq t_2 + t'_O/2$  for some positive integers  $t_1, t_2$ , with  $t_1 + t_2 \leq |\mathcal{V}^1|$ . Since  $|\mathcal{S}_1^1|$  and  $|\mathcal{S}'^1_2|$  are integers, we can use the bounds:

$$|\mathcal{S}_1^1| \leq t_1 + \lceil t'_O/2 \rceil, \quad |\mathcal{S}'^1_2| \leq t_2 + \lceil t'_O/2 \rceil \quad (30)$$

We now consider the full gadget  $L_O$ . When  $n = 0 \bmod 2$ , we have  $L'_O = L_O$ , so we can take  $\mathcal{S}_2^1 = \mathcal{S}'^1_2$ ; then with  $t'_O = t_O$ , from (30) we obtain the bounds in (29). When  $n \bmod 2 = 1$ , we must consider the additional wire  $d_n = c_n$ . We distinguish two cases. If  $n \notin \mathcal{O}$ , then we have  $t'_O = t_O$ , and we can take  $\mathcal{S}_2^1 = \mathcal{S}'^1_2$ ; therefore we obtain again (29). Finally if  $n \in \mathcal{O}$ , we have  $t'_O = t_O - 1$ , and we can take  $\mathcal{S}_2^1 = \mathcal{S}'^1_2 \cup \{n\}$ . This gives from (30):

$$|\mathcal{S}_2^1| \leq |\mathcal{S}'^1_2| + 1 \leq t_2 + \lceil (t_O - 1)/2 \rceil + 1 \leq t_2 + \lceil t_O/2 \rceil$$

which gives again (29).

We now consider the  $R_1$  and  $R_2$  gadgets. For  $R_1$  the  $t$ -SNI condition is:

$$|\mathcal{V}^2| + |\mathcal{S}_1^1| < \lfloor n/2 \rfloor \quad (31)$$

and when such condition is satisfied we have that the probed intermediate variables in  $\mathcal{V}^2$  and the output variables in  $\mathcal{S}_1^1$  can be simulated from a subset of input indices  $\mathcal{S}^2$  such that  $|\mathcal{S}^2| \leq |\mathcal{V}^2|$ . Similarly for gadget  $R_2$  the  $t$ -SNI condition is:

$$|\mathcal{V}^3| + |\mathcal{S}_2^1| < \lfloor n/2 \rfloor \quad (32)$$

Note that when  $n = 3$  we can still apply Condition (31) on  $R_1$ ; in that cases it requires  $|\mathcal{V}^2| = |\mathcal{S}_1^1| = 0$ , and then  $|\mathcal{S}^2| = 0$ .

Consider now the following two inequalities:

$$|\mathcal{V}^2| + t_1 + \lceil t_O/2 \rceil < \lfloor n/2 \rfloor \quad (33)$$

$$|\mathcal{V}^3| + t_2 + \lceil t_O/2 \rceil < \lfloor n/2 \rfloor \quad (34)$$

As previously, we have using (29) that Inequality (33) implies Condition (31) for gadget  $R_1$ , and similarly Inequality (34) implies Condition (32) for  $R_2$ . And as previously, at least one of the two inequalities (33) or (34) must be satisfied, since otherwise, using  $\lfloor x/2 \rfloor + \lceil x/2 \rceil = x$  for any  $x \in \mathbb{Z}$ , we obtain using  $t_1 + t_2 \leq |\mathcal{V}^1|$ :

$$n \leq |\mathcal{V}^2| + t_1 + \lceil t_O/2 \rceil + |\mathcal{V}^3| + t_2 + \lceil t_O/2 \rceil \leq |\mathcal{V}^2| + |\mathcal{V}^3| + |\mathcal{V}^1| + t_O \leq t + t_O$$

which contradicts the bound  $t + t_O < n$ . This again shows that the two gadgets  $R_1$  and  $R_2$  cannot be both saturated.

If both inequalities (33) and (34) are satisfied, then both gadgets  $R_1$  and  $R_2$  are non-saturated, and by recursively applying Lemma 3 on both gadgets, we get  $|\mathcal{S}^2| \leq |\mathcal{V}^2|$  and  $|\mathcal{S}^3| \leq |\mathcal{V}^3|$ . One can therefore let  $I = \mathcal{S}^2 \cup \mathcal{S}^3$  and simulate the  $L_I$  gadget as in the real circuit; we then have as required:

$$|I| \leq |\mathcal{S}^2| + |\mathcal{S}^3| \leq |\mathcal{V}^2| + |\mathcal{V}^3| \leq t$$

Assume now that (33) is satisfied and (34) is not. Then  $R_1$  is non-saturated and we can apply Lemma 3 on  $R_1$ , which gives as previously  $|\mathcal{S}^2| \leq |\mathcal{V}^2|$ . For  $R_2$  we take  $\mathcal{S}^3 = \{\lfloor n/2 \rfloor + 1, \dots, n\}$ , which means that all inputs of  $R_2$  must be simulated. We now consider the  $L_I$  gadget. By applying Lemma 5 for all  $1 \leq i \leq \lfloor n/2 \rfloor$  on each set of intermediate variables  $\{a_i, a_{\lfloor n/2 \rfloor + i}, r_i\}$  and output variable  $a_i \oplus r_i$ , where all output variables  $a_{\lfloor n/2 \rfloor + i} \oplus r_i$  must be simulated, and by summing the inequalities, we construct  $I \subset \{1, \dots, n\}$  such that:

$$|I| \leq 2 \cdot |\mathcal{S}^2| + |\mathcal{V}^4| + (n \bmod 2) \leq 2 \cdot |\mathcal{V}^2| + |\mathcal{V}^4| + (n \bmod 2) \quad (35)$$

where the additional term  $(n \bmod 2)$  comes from the wire  $b_n = a_n$  when  $n \bmod 2 = 1$ ; namely we must have  $n \in I$  since  $n \in \mathcal{S}^3$ .

It remains to show that  $|I| \leq t$ . Since by assumption (33) is satisfied and (34) is not, we have:

$$\begin{aligned} |\mathcal{V}^2| + t_1 + \lfloor t_O/2 \rfloor &< \lfloor n/2 \rfloor \\ |\mathcal{V}^3| + t_2 + \lceil t_O/2 \rceil &\geq \lceil n/2 \rceil = \lfloor n/2 \rfloor + (n \bmod 2) \end{aligned}$$

which gives:

$$|\mathcal{V}^2| + t_1 + \lfloor t_O/2 \rfloor + (n \bmod 2) < |\mathcal{V}^3| + t_2 + \lceil t_O/2 \rceil$$

which implies:

$$|\mathcal{V}^2| + (n \bmod 2) < |\mathcal{V}^3| + t_2 + (t_O \bmod 2)$$

and therefore:

$$|\mathcal{V}^2| + (n \bmod 2) \leq |\mathcal{V}^3| + t_2 \leq |\mathcal{V}^3| + |\mathcal{V}^1|$$

Then from (35) we obtain:

$$|I| \leq |\mathcal{V}^2| + |\mathcal{V}^3| + |\mathcal{V}^1| + |\mathcal{V}^4| \leq t$$

as required.

Assume conversely that (34) is satisfied and (33) is not. Then  $R_2$  is non-saturated and we can apply Lemma 3, which gives as previously  $|\mathcal{S}^3| \leq |\mathcal{V}^3|$ . For  $R_1$  we take  $\mathcal{S}^2 = \{1, \dots, \lfloor n/2 \rfloor\}$ , which means that all inputs of  $R_1$  must be simulated. We now consider the  $L_I$  gadget. By applying Lemma 5 for all  $1 \leq i \leq \lfloor n/2 \rfloor$  on each set of intermediate variables  $\{a_i, a_{\lfloor n/2 \rfloor + i}, r_i\}$  and output variable  $a_{\lfloor n/2 \rfloor + i} \oplus r_i$ , where all output variables  $a_i \oplus r_i$  must be simulated, and by summing the inequalities, we construct  $I \subset \{1, \dots, n\}$  such that:

$$|I| \leq 2 \cdot |\mathcal{S}^3| + |\mathcal{V}^4| \leq 2 \cdot |\mathcal{V}^3| + |\mathcal{V}^4| \quad (36)$$

Namely when  $n \bmod 2 = 1$ , if  $n \in \mathcal{S}^3$  we can put  $n$  in  $I$ , and therefore the bound (36) still applies.

Since by assumption (34) is satisfied and (33) is not, we have:

$$\begin{aligned} |\mathcal{V}^2| + t_1 + \lfloor t_O/2 \rfloor &\geq \lfloor n/2 \rfloor \\ |\mathcal{V}^3| + t_2 + \lceil t_O/2 \rceil &< \lceil n/2 \rceil = \lfloor n/2 \rfloor + (n \bmod 2) \end{aligned}$$

which gives:

$$|\mathcal{V}^3| + t_2 + \lceil t_0/2 \rceil < |\mathcal{V}^2| + t_1 + \lfloor t_0/2 \rfloor + (n \bmod 2)$$

which implies:

$$|\mathcal{V}^3| < |\mathcal{V}^2| + t_1 + (n \bmod 2)$$

and therefore:

$$|\mathcal{V}^3| \leq |\mathcal{V}^2| + t_1 \leq |\mathcal{V}^2| + |\mathcal{V}^1|$$

Then from (36) we obtain as previously:

$$|I| \leq 2 \cdot |\mathcal{V}^3| + |\mathcal{V}^4| \leq |\mathcal{V}^3| + |\mathcal{V}^2| + |\mathcal{V}^1| + |\mathcal{V}^4| \leq t$$

as required. This terminates the proof of Lemma 6. □