Linear Repairing Codes and Side-Channel Attacks

Hervé Chabanne, Houssem Maghrebi, Emmanuel Prouff

▶ **To cite this version:**

Hervé Chabanne, Houssem Maghrebi, Emmanuel Prouff. Linear Repairing Codes and Side-Channel Attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018, 2018 (1), pp.118-141. 10.13154/tches.v2018.i1.118-141 . hal-01973360

HAL Id: hal-01973360
https://hal.science/hal-01973360

Submitted on 8 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Linear Repairing Codes and Side-Channel Attacks

Hervé Chabanne[1], Houssem Maghrebi[2], and Emmanuel Prouff[3]

[1] OT-Morpho, France
herve.chabanne@morpho.com
[2] Underwriters Laboratories***
houssem.maghrebi@ul.com
[3] ANSSI†, France
emmanuel.prouff@ssi.gouv.fr

Abstract. To strengthen the resistance of countermeasures based on secret sharing, several works have suggested to use the scheme introduced by Shamir in 1978, which proposes to use the evaluation of a random d-degree polynomial into $n \geqslant d+1$ public points to share the sensitive data. Applying the same principles used against the classical Boolean sharing, all these works have assumed that the most efficient attack strategy was to exploit the minimum number of shares required to rebuild the sensitive value; which is $d+1$ if the reconstruction is made with Lagrange's interpolation. In this paper, we highlight first an important difference between Boolean and Shamir's sharings which implies that, for some signal-to-noise ratio, it is more advantageous for the adversary to observe strictly more than $d+1$ shares. We argue that this difference is related to the existence of so-called *exact linear repairing codes*, which themselves come with reconstruction formulae that need (much) less information (counted in bits) than Lagrange's interpolation. In particular, this result implies that, contrary to what was believed, the choice of the public points in Shamir's sharing has an impact on the countermeasure strength. As another contribution, we exhibit a positive impact of the existence of linear exact repairing schemes; we indeed propose to use them to improve the state-of-the-art multiplication algorithms dedicated to Shamir's sharing. We argue that the improvement can be effective when the multiplication operation in the base field is at least two times smaller than in its sub-fields.

1 Introduction

In the late nineties, attacks called *Side-Channel Analysis* (SCA) have been exhibited against cryptosystems implemented in embedded devices. Since the seminal

*** This work has been done when the author was working at Safran Identity and Security.

† This work has been done when the author was working at Safran Identity and Security.

works [20,21], the attacks have been refined and, in particular, the initial principle has been generalized in order to exploit several instantaneous leakage points simultaneously. This led to the introduction of the *higher-order* SCA attacks [26]. To defeat the latter ones, whose practicality has been argued in several papers [23,28,36], secret sharing techniques (aka masking) are currently the most promising type of countermeasures. They can indeed be applied to get implementations with a scalable security, parametrized by the number of shares and some physical properties of the device [7,30]. The core idea of secret sharing, originally introduced in [34], is to split any sensitive variable a manipulated by the device into several (say n) shares a_i, and to process elementary operations on them while maintaining the property that any tuple of $d < n$ intermediate results is independent of any secret-dependent value. The latter property is usually called *d^{th}-order security property* and the set of all the tuples of shares allowing for the reconstruction of a is called *reconstruction set* and is denoted by $\mathcal{R}_a$ (it contains all the tuples $I \subseteq [1..n]$ such that a may be reconstruced from $(a_i)_{i\in I}$)[4]. Let $\boldsymbol{L} = (L_1, \ldots, L_n)$ denote the random variable in $\mathbb{R}^n$ associated to the noisy observation of the sharing $\boldsymbol{a} = (a_1, \ldots, a_n)$ (itself viewed as a random variable) and let MI denotes the *mutual information* operator. The core idea behind the d^{th}-order security is that the complexity of extracting information from a sharing satisfying the latter property increases linearly with $\min_{I\in\mathcal{R}_a} \left(\prod_{i\in I} \mathsf{MI}(a_i; L_i)\right)^{-\frac{1}{2}}$. For a given order d the security challenge is therefore to minimize the product of mutual information for every $I \in \mathcal{R}_a$. In the literature, the most classical sharing is the *Boolean* one in which n is chosen equal to $d+1$ and the shares a_i satisfy $a = a_1 + \cdots + a_n$, with $+$ being the Boolean (xor) addition. To recover a from a Boolean sharing, the adversary needs information on all the coordinates of $\boldsymbol{a}$ (*i.e.* $\mathcal{R}_a$ is reduced to the tuple $\{1, 2, \ldots, n\}$) and the attack complexity, quantified by the minimum number τ_{Bool} of realizations of $\boldsymbol{a}$ that must be observed to succeed with some probability β, hence satisfies:

$$\tau_{\text{Bool}} > \alpha \times \left(\prod_{i\in[1..d+1]} \mathsf{MI}(a_i; L_i) \right)^{-\frac{1}{2}} . \tag{1}$$

where α is a some constant term depending on both β and the cardinality of the definition set of the a_i [11,30].

State of the Art. The simplicity of the Boolean sharing is an advantage from an implementation complexity point of view but, on the flip side, it helps the attacker: the information on the shared data is relatively easy to rebuild from the observed shares. Starting from this remark, Prouff and Roche in [31] and Goubin and Martinelli in [14] proposed indepedently to apply Shamir's secret sharing (SSS for short) instead of Boolean sharing: the core principle of SSS is to split any sensitive variable a into $n \geqslant 2d+1$ shares a_i which correspond to the evaluation,

[4] It may be checked that, by construction and due to the d^{th}-order security, every $I \in \mathcal{R}_a$ has size at least $d+1$.

in n distinct non-zero public elements, of a random degree-d polynomial with constant term a [34]. The d^{th}-order security property comes as a direct consequence of the so-called *collusion resistance* of Shamir's sharing which essentially ensures that at least $d+1$ evaluations (aka shares a_i) must be involved to recover a. For an (n, d)-SSS sharing of a_i (namely a splitting with n shares and a degree-d polynomial), $\mathcal{R}_a$ is exactly $\{I \subseteq [1..n]; \#I > d\}$ and the reconstruction is done by polynomial interpolation. At the cost of an increase of the implementation timing complexity (compared to that obtained for Boolean sharing), the authors of [14,31] argue, with simulations and experiments, that the intrinsic complexity of SSS significantly increases the value $\min_{I \in \mathcal{R}_a; \#I=d+1} \left(\prod_{i \in I} \mathsf{MI}(a_i; L_i)\right)^{-\frac{1}{2}}$ and hence the security. This type of informational argumentation is also the cornerstone of Balasch *et al.* 's work [2], which is based on the concept of Inner Product (IP) sharing [12], or of Wang *et al.* 's work [38]. Implicitely, it assumes that the minimum of the product of mutual information is achieved for a $(d+1)$-tuple and, under this assumption, the mimimum number τ_{SSS} of realizations of the SSS sharing that must be observed to recover a should satisfiy:

$$\tau_{\text{SSS}} > \alpha \times \min_{I \subseteq [1..n]; \#I=d+1} \left(\prod_{i \in I} \mathsf{MI}(a_i; L_i) \right)^{-\frac{1}{2}} . \tag{2}$$

Contributions. The previous hypothesis about the lower bound on τ_{SSS} is motivated by the assumption that the most efficient way to attack an (n, d)-SSS sharing is to observe exactly $d+1$ shares. The underlying remarks are that (1) observing strictly less than $d+1$ shares leaks no information on the shared variable (by security property of SSS) and (2) observing strictly more than $d+1$ shares will merely provide the attacker with more noise than information since $d+1$ shares are already sufficient to rebuild a (by interpolation)[5]. One of the goals of this paper essentially aims at coming back on the second point which has been assumed, without being proved, in all previous works studying SSS in the SCA context. We actually highlight in the first part of this paper an important difference between Boolean and SSS sharings which implies that, for some signal-to-noise ratio, it is more advantageous for the adversary to observe strictly more than $d+1$ shares. This observation is illustrated in Section 3 and it is confirmed by real attack experiments on an ATMega328p architecture. For instance, Fig. 1 hereafter shows that for a noise standard deviation $\sigma \in [0.2; 0.5]$, it is more interesting for the adversary to target 4 shares instead of 3 when attacking a $(5, 2)$-SSS sharing[6].

[5] For the secure multiplication proposed in [19], recent works have shown that attacks exploiting the n^2 shares $a_i \times b_j$ of $a \times b$ may lead to more efficient attacks than targeting only the n shares of a or b, when n is greater than $O(1/t)$ and t is an upper bound of the mutual information for each share [3,11]. Our work completes these works by exploiting algebraic dependencies in Shamir's sharing.

[6] Recent works (see *e.g.* [3] and [16]) have indeed argued that several manipulations of the same share could be used to improve the efficiency of attacks exploiting only

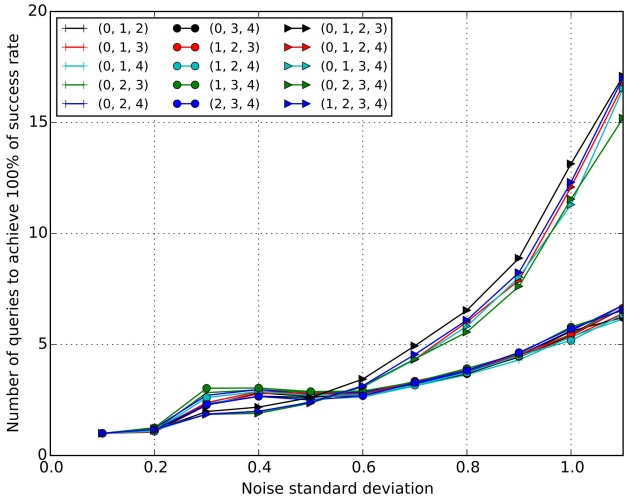

Fig. 1. For a $(5, 2)$-SSS sharing and different choice of tuples of shares, the number of observations required to achieve a 100% success rate (in y-axis) *versus* the noise standard deviation of the noise (in x-axis).

Our observation implies that the complexity of attacks against SSS may be lower than what was believed in previous studies (*i.e.* (2)) since the minimum of the product of mutual information must be processed over the full reconstruction set (and not only for the tuples of size $d + 1$):

$$\tau_{\text{SSS}} > \alpha \times \min_{I \subseteq [1..n]; \#I \geqslant d+1} \left(\prod_{i \in I} \mathsf{MI}(a_i; L_i) \right)^{-\frac{1}{2}} . \tag{3}$$

We argue that this property of SSS is directly related to the existence of *linear exact repairing structures* for Reed-Solomon codes [17,39]. The latter ones can indeed be viewed as polynomial interpolation formula that optimize the amount of information which needs to be extracted from a reconstruction tuple to recover the shared value. We exhibit two important consequences of this observation:

- firstly, it implies that the choice of the public points in SSS sharing plays a role in the security (and the efficiency) of the countermeasure,
- secondly, it implies that it may be more efficient for an adversary to extract strictly less than m bits of information on $d' > d + 1$ shares, than to extract m bits of information on exactly $d+1$ shares (when Lagrange's interpolation is applied).

$d + 1$ shares to defeat a d-secure masking. But they cannot explain the important difference observed in Fig. 1.

The core idea behind the second point is that the difficulty of extracting m/t bits of information from a noisy observation of an m-bit variable a_i decreases approximately exponentially with t (see the illustration of this assertion in Appendix A). Consequently, a template attack exploiting $d' > d+1$ shares, but needing only εm bits of information, $\varepsilon \in [0:1)$, for γ of them (and exactly m bits for the remaining $d'-\gamma$) with $\gamma > d'-d-1$,[7] may be more efficient than a classical template attack exploiting m bits of information on exactly $d+1$ shares. Of course, this efficiency gain cannot be true for any amount of noise since it is known that the complexity of a side-channel attack increases exponentially with the number of exploited shares, the basis of the exponentiation being the noise standard deviation σ (assumed to be the same for all the shares). Figure 6 illustrates our argumentation in the particular case $(n,d) = (5,2)$, $d' = 4$, $\gamma = 3$ and $\varepsilon = 0.5$. It corresponds to the attack results reported in Fig. 1.

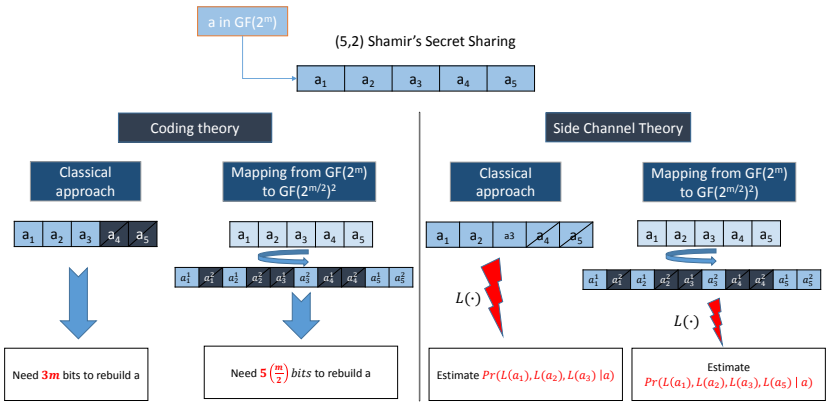

Fig. 2. Side-channel and linear repairing codes for Shamir's sharing.

In the second part of the paper, we exhibit a positive impact of the existence of linear exact repairing schemes for SSS by proposing an improvement of the state-of-the-art multiplication algorithms dedicated to this sharing. Our new proposal continues the line of studies made *e.g.* in [14] and [6] on the original scheme by Ben-Or *et al.* [4], and improves them when the multiplication operation in $\mathrm{GF}(2^m)$ is at least two times smaller than in the sub-fields.

[7] This condition implies that the amount of information (counted in bits) needed to rebuild the shared value is bounded above by $(d'-\gamma)m + \gamma\varepsilon m$ which is lower than $(d+1)m$.

2 Preliminaries on Shamir's Secret Sharing and Coding Theory

In this section we introduce the concepts involved in the subsequent sections. In particular, we recall the principles of linear exact repair schemes and we specify the recent constructions introduced in [17] to fit with our context (where we are focussing on the problem of secret reconstruction and not on the problem of decoding which is more general).

2.1 Shamir's Secret Sharing and Reed-Solomon Codes

In a seminal paper [34], Shamir proposed to split a secret $a \in \mathrm{GF}(2^m)$ into n shares such that no tuple of shares with cardinality lower than a so-called *threshold* $d < n$ depends on a. Shamir's protocol consists in associating a with a random polynomial $P_a(X) \doteq a + \sum_{i=1}^{d} u_i X^i$ of degree lower than d and with constant term $a = P_a(0)$ (this essentially amount to randomly generate the d coefficients u_i in $\mathrm{GF}(2^m)$). Then, the polynomial $P_a(X)$ is evaluated in n distinct public non-zero elements $\alpha_1, \ldots, \alpha_n$ in $\mathrm{GF}(2^m)$ to define a so-called (n, d)-*sharing* $(a_1, a_2, \cdots, a_n)$ of a such that $a_i = P_a(\alpha_i)$. To re-construct a from its sharing, polynomial interpolation is first applied to re-construct the polynomial from a subset U of at least $d + 1$ among its n evaluations a_i and then, it is evaluated in 0. Actually, using Lagrange's interpolation formula, the two steps can be combined in a single one thanks to the equality:

$$a = \sum_{a_i \in U} a_i \times \beta_i \ , \tag{4}$$

where the (public) constants β_i are defined as

$$\beta_i = \prod_{k=1, k \neq i}^{n} \frac{\alpha_k}{\alpha_i + \alpha_k} \ . \tag{5}$$

The set U is sometimes called *reconstruction set*. The vector composed of the n weights β_i is denoted by $\boldsymbol{\beta}$ and (4) is called reconstruction.

As initially observed by McEliece and Sarwate in [25], the sharing of a described above may be viewed as an *encoding* with a *Reed-Solomon* linear code. Generally speaking, a *linear code* $\mathcal{C}$ of length n and dimension k over a finite field $\mathbb{K}$ is a k-dimensional subspace of $\mathbb{K}^n$. It is denoted by $\mathcal{C}[n, k]$. A Reed-Solomon code is a particular linear code whose definition is recalled hereafter.

Definition 1 (Reed-Solomon Code). *The Reed-Solomon code* $\mathrm{RS}(\mathcal{S}, d+1) \subseteq \mathbb{K}^n$ *of dimension* $d + 1$ *over a finite field* $\mathbb{K}$ *and with evaluation subset* $\mathcal{S} = \{\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_n\}$ *of* $\mathbb{K}$ *is the subspace:*

$$\mathrm{RS}(\mathcal{S}, d+1) = \{(P(\alpha_0), P(\alpha_1), \ldots, P(\alpha_n)); P(X) \in \mathbb{K}[X] \text{ and } \deg(P) \leqslant d\} \ .$$

Reed-Solomon codes are *Maximum Distance Separable* (MDS) codes, which means that any tuple of $d+1$ symbols (that is, any tuple of $d+1$ evaluations of a polynomial $P(X) \in \mathrm{RS}(\mathcal{S}, d+1)$) can be used to recover the entire codeword (that is, $P(X)$ itself). In terms of RS codes, the sharing of a variable a with SSS is an encoding with a Reed-Solomon code $\mathrm{RS}(\{0, \alpha_1, \ldots, \alpha_n\}, d+1)$:

$$(a, a_1, \cdots, a_n) = (a, u_1, \cdots, u_d) \times G \ , \tag{6}$$

with G the matrix $(\alpha_i^j)_{i\in[0;n], j\in[0;d]}$ and $\alpha_0 \doteq 0$ (with the convention $0^0 = 1$). After denoting by G_i the ith column of G, it may indeed be checked that we have:

$$a_i = P_a(\alpha_i) = a + \sum_{j=1}^{d} u_j \alpha_i^j = (a, u_1, \ldots, u_d) \cdot G_i \ .$$

The reconstruction of a then simply corresponds to a simple *decoding* which will be the matrix representation of Lagrange's interpolation recalled in (4). For simplicity we will sometimes say that $\boldsymbol{a} = (a, a_1, \cdots, a_n)$ is a sharing of a, which will mean that a can be recovered from (a part of) $\{a_1, \cdots, a_n\}$.

2.2 Linear Exact Repairing Codes

The reconstruction of a by Lagrange's Interpolation (4) requires the full knowledge of at least $d+1$ shares $a_i \in \mathrm{GF}(2^m)$, that is $(d+1) \times m$ bits. The theory of Linear Exact Repairing Schemes aims at defining methods allowing for the reconstruction of $a \in \mathrm{GF}(2^m)$ from its sharing with strictly (much) less than $(d+1) \times m$ bits. This goal is achieved by exploiting partial information on strictly more than $d+1$ shares. If the total amount of partial information is strictly less than RB_{Lagrange}, the goal is achieved. During the last few years, there have been several publications tackling this issue in the context of RS codes (see *e.g.* [10,35]), or more generally, for MDS codes (see *e.g.* [5,9,37]). Before presenting the recent constructions of LERS for RS codes introduced in [17], let us recall hereafter the definition of a *field trace*.

Definition 2 (Field Trace). *Let q be a power of a prime integer and let t be a strictly positive integer, then the* field trace $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}$ *from $\mathbb{K} = \mathrm{GF}(q^t)$ to $\mathbb{F} = \mathrm{GF}(q)$ is defined for every $\beta \in \mathbb{K}$ by*

$$\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\beta) = \beta + \beta^q + \beta^{q^2} + \cdots + \beta^{q^{t-1}} \ .$$

In a linear repairing scheme, the reconstruction of the shared value $a \in \mathbb{K}$ involves, for each share a_i, zero or more elements of $\mathbb{F} \subseteq \mathbb{K}$ of the form $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\gamma_{i,j} a_i)$ for some well-chosen field elements $\gamma_{i,j} \in \mathbb{K}$. A linear exact repair scheme can then be described by the field elements $\gamma_{i,j}$ used for all the shares a_i, along with a (linear) repair algorithm. Minimizing the number of these elements is the main goal when designing an LERS. It is called the *Repair Bandwith* RB_{LERS} of the LERS and it is formally defined as the maximum number of sub-symbols in $\mathbb{F}$ which must be exploited to recover $P_a(\alpha^\star)$ when $\alpha^\star$ ranges

over $\mathcal{S}$. It may be observed that, in the context of SSS, we only need an efficient reconstruction scheme for $\alpha^\star = 0$ (*i.e.* for $a = P_a(0)$) and thus, we will use RB_{LERS} as an upper bound and will denote by LERS_0 a repairing scheme trying to minimize RB_{LERS} only for $\alpha^\star = 0$.

In the following section, we give, for the specific case of Reed-Solomon codes applied for secret sharing, the main outlines of the method recently proposed in [17] to construct the field elements $\gamma_{i,j}$ described previously together with an efficient repair algorithm. The presentation is completed in Appendix D by a detailed presentation of an example given in [17] and by `Sage` scripts that we made available at [1].

2.3 Explicit Constructions for Reed-Solomon Codes

In the specific case of Reed-Solomon codes $\mathrm{RS}(\{0, \alpha_1, \ldots, \alpha_n\}, d+1)$ defined over $\mathbb{K} = \mathrm{GF}(2^m)$, [17, Theorem 4] implies that the construction of a LERS_0 over $\mathbb{F} = \mathrm{GF}(2^{\frac{m}{t}})$ for some field extension t is equivalent to find a set $\{p_1, \ldots, p_t\}$ of t polynomials of degree $n-d-1$ in $\mathbb{K}$ such that:

$$\dim_{\mathbb{F}}(\{p_1(0), \ldots, p_t(0)\}) = t \tag{7}$$

and

$$RB_{\mathrm{LERS}_0} \geqslant \sum_{i=1}^{n} \dim_{\mathbb{F}}(\{p_1(\alpha_i), \ldots, p_t(\alpha_i)\}) \ .$$

Once such a family of polynomials is found with $RB_{\mathrm{LERS}_0} < t(d+1)$ (otherwise the bandwidth is worst than the trivial one), coefficients ζ_j and $\mu_{i,j}$ with $j \in [1..t]$ are built so that

$$\zeta_j = p_j(0) \tag{8}$$

and

$$\mu_{i,j} = p_j(\alpha_i) \times \beta_i \ , \tag{9}$$

where the β_i are defined as in (5).

Then, the last step of the LERS_0 design consists in computing, for any α_i, a basis $\mathcal{B}_i = \{\gamma_{i,j}\}_j$ for the smallest vector space over $\mathbb{F}$ containing $\{\mu_{i,j}; j \in [1;t]\}$ as a subset (by definition the size of the basis is at most t, and actually is frequently strictly lower, which is a core observation to understand the soundness of the decoding with an LERS scheme). With the bases in hand, the reconstruction of $a = P_a(0)$ for any polynomial P_a of degree lower than d consists in the following steps where we recall that each a_i equals $P_a(\alpha_i)$ by SSS construction:

- for any $i \in [1..n]$; compute[8] the values $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\mu_{i,j} a_i)$, with $j \in [1..t]$, from the elements in $\{\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\gamma_{i,j} a_i); \gamma_{i,j} \in \mathcal{B}_i\}$,

[8] Since $(\gamma_{j,j})_j$ forms a basis, this can be done by only processing linear combinations of the $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\gamma_{i,j} a_i)$.

– for any $j \in [1;t]$; evaluate

$$\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_j a) = \sum_{i=1}^{n} \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\mu_{i,j} a_i) \tag{10}$$

and, eventually, recover the shared value thanks to the following equality

$$a = \sum_{j=1}^{t} \nu_j \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_j a) \ , \tag{11}$$

where $\{\nu_1, \ldots, \nu_t\}$ is the dual basis of $\{\zeta_1, \ldots, \zeta_t\}$.

It may be observed that Equations (10) and (11) together give the following alternative to Lagrange reconstruction Formula (4)

$$a = \sum_{i=1}^{n} \sum_{j=1}^{t} \nu_j \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\mu_{i,j} a_i) = \sum_{i=1}^{n} \sum_{j=1}^{t} \nu_j \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(p_j(\alpha_i)\beta_i a_i) \ . \tag{12}$$

The core observation in [17] is that, surprisingly, for some a_i (and hence some $P_a(\alpha_i)$) the size of all the elements required to build the set $\{\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\mu_{\alpha,j} a_i); j \in [1;t]\}$ may be smaller than the size of a_i. This implies that the total number of bits required to recover $P(0)$ from (10) and (11), which actually equals $RB_{\mathrm{LERS}} \times \frac{m}{t}$, may be smaller than $(d+1)m$ (corresponding to a direct polynomial interpolation).

In the following section we argue that the existence of efficient LERS for Reed-Solomon codes impacts the security evaluation of Shamir's sharing in the context of side-channel analyses. In particular, we argue that the choice of the public points plays a non-negligible role in the security provided by this sharing and we exhibit (through several simulations and experiments) contexts in which state-of-the-art attacks are suboptimal.

3 Side-Channel Analysis of Shamir's Secret Sharing Scheme

We first recall that a scheme is said to be d^{th}-*order secure* if any set of at most d intermediate results during the processing reveals no information about a secret (sensitive) value. When the (n, d)-sharing of Shamir is involved to secure the implementation, the recovery of the shared values requires the knowledge of at least $d + 1$ shares among the $n = 2d + 1$ possible ones. As already discussed in the introduction, all previous works on SSS assume that the most efficient way to attack such a sharing of a is to exploit exactly $d+1$ shares. If the shared value is assumed to belong to GF(2^m), then this implies that the literature implicitly assumes that, as for Boolean sharing/masking, $(d + 1) \times m$ bits are necessary and sufficient to rebuild a. This section aims at coming back on this assumption and, more precisely, at answering the two following questions while having in mind the recent results on LERS described in previous section:

Question 1. For a given pair of sharing parameters (n, d) and a given representation of the base field $\mathrm{GF}(2^m)$, does any combination of $d+1$ shares $(a_i)_{i \in I, \#I=d+1}$ give the same amount of information an a. Actually, an attacker is spoilt for the choice of the combination of $d+1$ shares to use among the several available ones. Our goal here is to check if all $d+1$ combination of shares leak the same amount of information under the same fixed noise level for each share. The result of this investigation may bring some insights on the choice of the optimal $(d+1)$-combination of shares, if any exits, that an attacker should consider when performing his attack.

Question 2. For a given pair of sharing parameters (n, d) and a given representation of the base field $\mathrm{GF}(2^m)$, is there some $(d+2)$-combination of shares whose observation leads to a better attack than any other one with exactly $d+1$ shares? We shall see that the answer to this question depends on the amount of noise in each observation (actually we will argue that the answer is positive if and only if the signal-to-noise ratio is lower than some bound).

3.1 Preliminar Observations from Simulated Leakage Measurements

To address the questions above, we performed profiling attacks against simulated traces corresponding to the manipulation of the shares of a (n, d)-sharing for $n = 5$ and $d = 2$. This allows us to study the impact of the noise on the attacks efficiency, and to draw some conclusions. Our analyses are confirmed, in Section 3.3, by practical attack experiments against real acquisitions captured on the ChipWhisperer platform [27].

Target Implementation. It is assumed that the adversary has access to a noisy observation of the five elements $a_1, \ldots, a_5$ of the $(5, 2)$-sharing of a secret value $a \in \mathrm{GF}(2^8)$.

Leakage Model. The observation ℓ_i of each share a_i is assumed to be the sum of two *mutually independent* parts: a deterministic function f of the share a_i and a *Gaussian noise* N_i such that $\ell_i = f(a_i) + N_i$. To generate our traces, we considered two types of deterministic functions: the Hamming weight (which has been argued to be a sound approximation for many device technologies – see *e.g.* [24]) and the identity. The noises N_i are assumed to be mutually independent, to have zero mean and to have the same fixed variance σ^2 (the rationale behind this assumption is to have the same noise level for each share). It may be checked that, under our assumptions, the distribution of $\ell_i \mid a_i$ has a normal distribution with mean $f(a_i)$ and variance σ^2. Such a normal distribution is hereafter denoted by $x \in \mathbb{R} \mapsto \phi_{f(a_i),\sigma^2}(x)$.

Attack Strategy. To evaluate the security of the $(5, 2)$-sharing, we performed a higher-order template attack following the procedure described in [22]. The idea behind this choice was to directly address the most powerful adversary

who follows a *maximum likelihood approach* without modelling error. Let $\boldsymbol{\ell_j} \in \mathbb{R}^5$ denotes the j^{th} observation of the 5-tuple $(\ell_1, \ldots, \ell_5)$ and let $\ell_{1,j}$, ..., $\ell_{5,j}$ denote its coordinates, then for any hypothesis $\hat{a} \in GF(2^8)$ on a, the likelihood distinguisher[9] is defined as $\mathrm{d}_{\mathrm{ML}}(\hat{a}) = \prod_{j=1}^{N} p_{\hat{a}}(\boldsymbol{\ell_j})$, where N denotes the number of observations and where $p_{\hat{a}}$ denotes the probability density function (pdf) of the leakage which, under the Gaussian assumption, satisfies:

$$p_{\hat{a}}(\boldsymbol{\ell_j}) = \frac{1}{256^4} \sum_{a_2 \in \mathrm{GF}(2^8)} \cdots \sum_{a_5 \in \mathrm{GF}(2^8)} \prod_{i=1}^{5} \phi_{f(a_i),\sigma^2}(\ell_{i,j}) \ , \tag{13}$$

where we recall that we have $a_1 = \frac{1}{\beta_1}(\hat{a} + \sum_{i=2}^{5} a_i \times \beta_i)$ (see (4)).

In [22] authors have demonstrated that, for a d^{th}-order Boolean masking, (13) can be expressed as a higher-order convolution product whose evaluation complexity is $O(d)$. To continue using this efficient procedure in the context of Shamir's secret sharing, we propose hereafter to apply the simple linear change of variables $a_i \mapsto a'_i \doteq a_i \times \frac{\beta_i}{\beta_1}$ and $\hat{a} \mapsto \hat{a}' \doteq \hat{a} \times \frac{1}{\beta_1}$ so that $a_1 = \hat{a}' \oplus \bigoplus_{i=2}^{5} a'_i$, which enables us to apply the *convolution* trick to compute the score $\mathrm{d}_{\mathrm{ML}}(\hat{a} \times \beta_1^{-1})$.

Attack Results. For each possible combination of 3 shares, the left-hand side of Fig. 3 plots the number of traces (in y-axis) required to achieve a success rate of 100% according to an increasing noise standard deviation (in x-axis). We stress the fact that for each value of the standard deviation, the success rate has been averaged over 1.000 tries.

It may first be observed that the different triplet of shares that have been tested leak differently on the shared variable a (even if all the shares have been affected by the same amount of noise). For instance, for a noise standard deviation comprised in $[0.5, 1.5]$, the number of traces needed to always succeed in recovering the shared variable is around two times when exploiting the triplet of shares (a_1, a_2, a_3) instead of (a_2, a_3, a_4). To confirm this observation, we changed the public points α_i used for the sharing (see (2)). Results are plotted in the right-hand side of Fig. 3, where our previous observation is confirmed but for a different ordering of the triplets of shares. This simulations led us to answer positively to Question 1: some combination of $d+1$ shares bring to the adversary more information than others in Shamir's sharing.

To address Question 2, we performed another set of simulations to check if there exist an SNR range for which exploiting the leakage on 4 shares is more efficient than any other combination of 3 shares. In Fig. 4, we show a comparison between the template attack results when considering all combination of 3 and 4 shares. The experiences are repeated 1.000 times for each combination of shares.

[9] In practice, one often makes use of the equivalent (averaged) log-likelihood distinguisher.

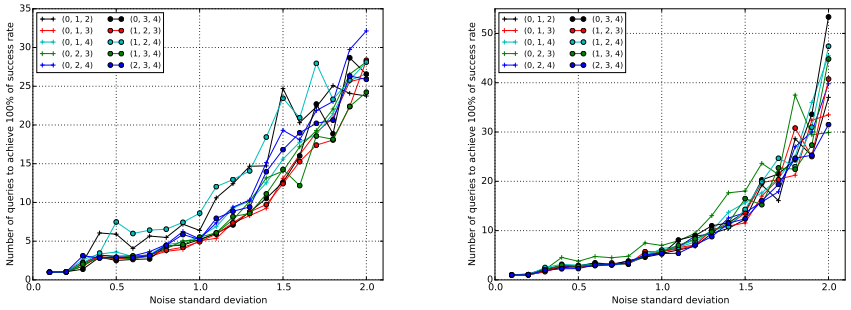

Fig. 3. Evolution of the number of queries (y-axis) to achieve a success rate of 100% according to an increasing noise standard deviation (in x-axis) for public points $\{125, 246, 119, 104, 150\} \subseteq \mathrm{GF}(2^8)$ (on the left-hand side) and $\{86, 23, 115, 107, 189\}$ (on the left-hand side).

For a specific interval of noise standard deviation (between 0.3 and 0.5), it may be observed that some combinations of 4 shares outperform all other combination of 3 shares. The same observation stands when changing the set of public points as described in the right-hand side of Fig. 4. We sum-up hereafter our main observations:

- the shares are not leaking the same amount of leakage despite the fact that the same quantity of noise has been added to them,
- exploiting the leakage of 4 shares is more efficient than any other attack strategy based on 3 shares for some values of the noise standard deviation.
- when comparing the two sides of Fig. 4, it may be observed that the optimal combination of 4 shares to recover the secret value changes with respect to the used public points. So, the amount of information leaked by the shares depends on the set of public points used to share the secret value.

3.2 Impact on the Existence of an LERS on the Efficiency of SCA against SSS

The existence of efficient LERS for some choices of public points and of field representations (aka the existence of reconstruction formulae as in (10)-(11) or in (18)) explains the observations made for previous experiments. Indeed, it implies that the statistical dependency between the value a and the tuple of shares involved in the attack can be revealed by extracting partial information on some of the shares, which renders the exploitation of the noisy observations of these shares more tolerant to error. Even if the reconstructions formulae are not directly involved in the template attacks we have performed, their existence is implicitly used by the maximum likelihood distinguisher to capture the dependency. From the designer point of view, a direct consequence is that the set of

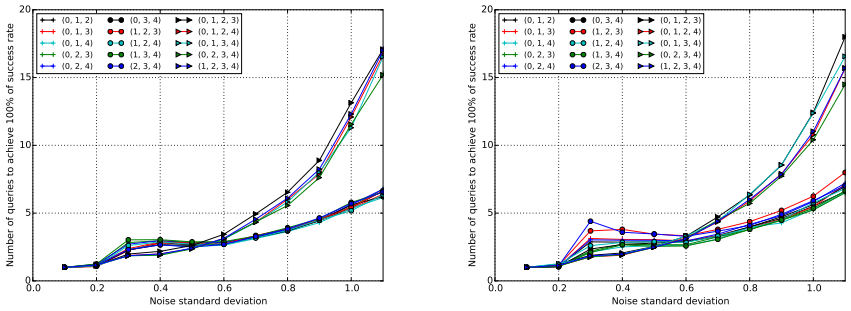

Fig. 4. Evolution of the number of queries (in y-axis) to achieve a success rate of 100% according to an increasing noise standard deviation (x-axis) for public points $\{5, 237, 175, 221, 198\} \subseteq \mathrm{GF}(2^8)$ (on the left-hand side) and $\{169, 63, 106, 49, 112\}$ (on the left-hand side).

public points and the field representation must be carefully chosen to make the construction of efficient LERS as difficult as possible. In Section 4, we show a constructive impact of LERS by proposing enhancements of the secure multiplication of data shared with SSS.

3.3 Attack results on Real Device

In order to confirm our observations on simulated traces, we conducted some practical attacks on the ChipWhisperer platform. The experimental setup is described in what follows.

We have implemented a $(5, 2)$ secret sharing on an 8-bit AVR microprocessor atxmega128d3 and we acquired power-consumption traces thanks to the ChipWhisperer-Lite (CW1173) basic board. For the leakage profiling step, we have collected 128.000 traces to estimate the mean and the variance for each share. These statistical estimation results have first been used to check that the shares observations have roughly the same variance (*i.e.* are impacted by the same level of noise) which is in-line with our simulation setup. However, the standard deviation of the noise of the acquired traces was actually outside the suitable area where the combinations of 4 shares outperform the combination of 3 shares. The estimated noise standard deviation is about 0.7. Thus, we have acquired 15.000 traces for the attack phase and we have performed the template attack by exploiting the leakage of all combinations of 4 and 3 shares. As expected, the attack results have demonstrated that any combinations of 4 shares is efficient with respect to the measures noise standard deviation. To decrease the noise level and to fit the suitable interval, we filtered the acquired traces to reach a noise standard deviation value equal to 0.5. Finally, we re-performed our higher-order template attack. The success rate of the attack for each combination of share is shown in Fig. 5.

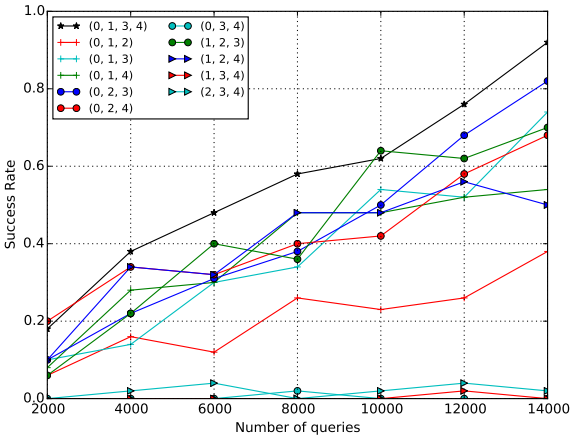

Fig. 5. Success rate of the template attack according to an increasing number of traces.

Fig. 5 shows that exploiting the leakage of the shares $(0, 1, 3, 4)$ is more suitable in terms of attack efficiency than exploiting any other tuple of 3 shares.

4 Impact of LERs to Improve Existing Multiplication Schemes for SSS

In this section we describe a constructive impact of the existence of an LERS by showing that they can be used to improve the multiplication of data shared by SSS.

4.1 Basics on the Multiplication for SSS

To define a d^{th}-order masking scheme for a block cipher implementation where each intermediate result is split with Shamir's technique, one must specify a secure method for the processing of field multiplications over GF(2^m). Most of (if not all) existing protocols start from a multiplication scheme introduced by Ben-Or *et al.* in the context of the Multi-Party Computation Theory [4]. For this protocol to work, the number of shares n per variable must be at least $2d + 1$ and for $n = 2d+1$, it is proved that it satisfies a security property encompassing the d^{th}-order SCA security [31]. We give hereafter the adaptation of [4] in the SCA context as proposed in [31,33][10].

[10] The protocol is an improved version of the protocol originally proposed by Ben-Or *et al.* [4], due to Gennaro *et al.* in [13].

Algorithm 1 Secure Multiplication For Shamir's Secret Sharing

Input: two integers n and d such that $n \geq 2d+1$, the (n,d)-sharings $(u_i)_i = (P_u(\alpha_i))_i$ and $(v_i)_i = (P_v(\alpha_i))_i$ of u and v respectively. The n distinct points α_i, the interpolation values $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_n)$.

Output: the (n,d)-sharing $(w_i)_i \doteq (P_w(\alpha_i))_i$ of $w = u \times v$.

▷ Compute a Boolean n-sharing $(t_1, \cdots, t_n)$ of w

for $i = 1$ **to** n **do**
  $t_i \leftarrow \beta_i \times P_u(\alpha_i) \times P_v(\alpha_i)$
end for

▷ Compute a sharing $(Q_i(\alpha_j))_{j \leq n}$ for every t_i

for $i = 1$ **to** n **do**
  for $k = 1$ **to** d **do**
    $\omega_k \leftarrow \texttt{rand}(\mathrm{GF}(2^n))$
  end for
  for $j = 1$ **to** n **do**
    $Q_i(\alpha_j) \leftarrow t_i + \sum_{k=1}^{d} \omega_k \times \alpha_j^k$
  end for
end for

▷ Compute the share $w_i = P_w(\alpha_i)$ for $w = u \times v$

for $i = 1$ **to** n **do**
  $w_i \leftarrow \sum_{j=1}^{n} Q_j(\alpha_i)$
end for

The completeness of Algorithm 1 is discussed in [4]. Its d^{th}-order SCA security can be straightforwardly deduced from the proof given by Ben-Or *et al.* in [4] in the secure multi-party computation context. Eventually, for $n = 2d+1$ (which is the parameter choice which optimizes the security/efficiency overhead), the complexity of Algorithm 1 in terms of additions and multiplications is $\mathcal{O}(d^3)$.

4.2 Link with Error Correcting Codes Theory and State-Of-The-Art Improvements

Algorithm 1 may be rewritten in terms of coding theory, essentially because, as recalled in Section 2, an (n,d)-sharing with Shamir's scheme exactly corresponds to an encoding by a Reed-Solomon code with parameters $[n+1, d+1]$ [25]. Let $\circ$ denote the Hadamard product between vectors (aka the componentwise product) and let $\mathcal{C}$ denote the latter code and let $\mathcal{C}^\star$ denote the code built from $\mathcal{C}$ by taking all the vectors $\boldsymbol{\beta} \circ c \circ c'$ with $c, c' \in \mathcal{C}$ and $\boldsymbol{\beta} = (1, \beta_1, \cdots, \beta_n)$ (recall that the β_i are interpolation values specified for the Reed-Solomon code with parameters $[n+1, d+1]$). The code $\mathcal{C}^\star$ is the so-called *generalized Reed-Solomon code* defined with respect to the evaluation points $\mathcal{S}$ and the multiplier vector $\boldsymbol{\beta}$ by:

$$\mathcal{C}^\star = \{(\beta_i P(\alpha_i))_{i \in [1;n]};\ P(X) \in \mathrm{GF}(2^m)[X] \text{ and } \deg(P(X) \leqslant n\} \ . \tag{14}$$

On one hand, the tuple $(t_1, \cdots, t_n)$ in Algorithm 1 forms a sharing of $w = u \times v$ such that $c^\star = (w, t_1, \cdots, t_n) \in \mathcal{C}^\star$ and a reconstruction algorithm is simply

given by the sum of the t_i (this is a direct consequence of Lagrange's interpolation formula). On the other hand, loop 4-11 (over indices i and j) and loop 12-14 may be viewed as a *transcoding* $\mathbf{transcoding}_{\mathcal{C}^\star \to \mathcal{C}}$ that securely transforms the sharing $c^\star \in \mathcal{C}^\star$ of w into a new sharing c in $\mathcal{C}$. Eventually, it can be observed that the algorithm completeness holds because the constant term of the polynomial $P_w(X) \doteq \sum_{j=1}^{n} Q_j(X)$ associated to the codeword $\mathcal{C}$ has constant term the n-reconstruction $\sum_i \beta_i \times t_i$. Algorithm 1 involves n multiplications and the evaluations of n degree-d polynomials in n points (which makes $n^2 \times d$ multiplications for a naive implementation and $O(n \log^2 n)$ multiplications using FFT-based polynomial division[11] [8]). In [6], the authors observe that the addition of a random sharing $c^0 \doteq (0, c_1^0, \ldots, c_n^0) \in \mathcal{C}^\star$ of 0 to $c^\star$ makes it possible to reduce the number of polynomials to evaluate from n to $d+1$, without compromising the security at order d. The core idea is to precede the transcoding by a shortening of the sharing $\hat{c} \doteq (t_1 + \beta_1 \cdot c_1^0, \cdots, t_n + \beta_n \cdot c_n^0)$ into $(\sum_{i=1}^{n-d} \hat{c}_i, \hat{c}_{n-d+1}, \cdots, \hat{c}_n)$. The security essentially holds because the sharing c^0 may correspond to any polynomial of degree lower than or equal to n, which implies that at least $n-1$ shares/evaluations are required to recover all the information about the sharing (see [6] for more details about the proof). Eventually, a last improvement can be obtained by applying to loop 12-14 (in Algorithm 1) an idea initially proposed in [14]: since polynomials $Q_i(X)$ (and hence also $P_w(X)$) all have degree d, the evaluation can be made in only $d+1$ points (*e.g.* the $d+1$ first public points $\alpha_1, \ldots, \alpha_{d+1}$) and Algorithm 1 can output only the $d+1$ shares $w_1, \ldots, w_{d+1}$. When needed, the remaining evaluations/shares $w_{d+2}, \ldots, w_n$ can then be deduced by applying the following formula where the Lagrange's coefficients $\beta_{j,i} \doteq \prod_{k \neq j, k=1}^{d+1} (\alpha_i - \alpha_k)/(\alpha_j - \alpha_k)$ have been pre-computed and can be public[12]:

$$w_{d+1+i} = \sum_{j=1}^{d+1} w_j \times \beta_{j,i} \ . \tag{15}$$

This enables to exchange the $(d+1)dn$ evaluations (needed for the transcoding) to $d(d+1)^2 + (n-d-1)(d+1)$. Let c^u and c^v respectively denote the tuples of shares corresponding to the codewords in $\mathcal{C}$ associated to u and v. Bellow, we sum-up the resulting secure multiplication scheme and the cost in terms of field multiplications in the case $n = 2d+1$ (which is the optimal choice):

- [build the Sharing in $\mathcal{C}^\star$] $c^\star = c^u \circ c^v \circ \boldsymbol{\beta}$ $(4d+2$ *mult.*$)$
- [add the random sharing of 0 in $\mathcal{C}^\star$] $\hat{c} = c^\star + c^0 \circ \boldsymbol{\beta}$ $(2d+1$ *mult.*$)$
- [reduce dimension of the sharing] $\hat{c}^- = \left(\sum_{i=1}^{n-d} \hat{c}_i, \hat{c}_{n-d+1}, \cdots, \hat{c}_n\right)$
- [transcode $\hat{c}^-$ into a sharing in $\mathcal{C}$] $(d(d+1)^2 + d(d+1)$ *mult.*$)$

Performances. Clearly, the complexity of the multiplication is driven by the cost of the transcoding step which is in $O(d^3)$ while the other steps are in

[11] The constant terms being important in this complexity the naive approach is always more efficient for practical choices of n and d.

[12] As argued in [32], the processing of (15) can be simply made securely at order d without requiring additional multiplications.

$O(d)$. For efficiency reasons, *e.g.* when the cost of the field multiplication is significantly lower in $\mathrm{GF}(2^{m/2})$ than in $\mathrm{GF}(2^m)$, it is preferred to work on the vector space $\mathrm{GF}(2^{m/2})^2$ instead of $\mathrm{GF}(2^m)$. This is for instance the case in software for $m = 8$, since the field multiplication can be tabulated in GF(16) (and hence corresponds to a table access) but not in GF(256) (and hence often takes several dozens of CPU cycles – see *e.g.* [15] –). Working in $\mathrm{GF}(2^{m/2})^2$ instead of $\mathrm{GF}(2^m)$ essentially amounts to represent $\hat{c}^- \in \mathrm{GF}(2^m)^{d+1}$ as the pair $(\hat{c}_1^-, \hat{c}_2^-) \in \mathrm{GF}(2^{m/2})^{d+1} \times \mathrm{GF}(2^{m/2})^{d+1}$ and to process the transcoding over $\mathrm{GF}(2^{m/2})^2$ both for $\hat{c}_1^-$ and $\hat{c}_2^-$. This leads to exchange $d(d+1)^2 + d(d+1)$ multiplications over $\mathrm{GF}(2^m)$ for the following number of multiplications over $\mathrm{GF}(2^{m/2})$:

$$\mathtt{Comp}_{soa} = 2d(d+1)^2 + 2d(d+1) \ .$$

Before describing how the existence of efficient LERS can be used to improve the state of the art multiplication algorithms for SSS, we specify hereafter the description made in Section 2.3 for the particular case involved in the description of our improvement proposal.

4.3 LERS For Reed-Solomon Codes in the Particular case $t = 2$

Let us focus on a Reed-Solomon code $\mathrm{RS}(\mathcal{S}, d+1)$ defined over $\mathbb{K} = \mathrm{GF}(2^m)$ and with $\mathcal{S} \doteq \{0, \alpha_1, \ldots, \alpha_n\}$. When $t = 2$, the basis $\mathcal{B}_i$ of $\{\mu_{i,1}, \mu_{i,2}\}$ defined in Section 2.3 for every $i \in [1 \ldots n]$ is either of cardinal 2 if $\mu_{i,1}$ and $\mu_{i,2}$ are linearly independent over $\mathbb{F}$, or of cardinal 1 otherwise. In the first case, its two elements $\gamma_{i,1}$ and $\gamma_{i,2}$ may simply be set to $\mu_{i,1}$ and $\mu_{i,2}$ respectively. In the second case, we fix $\gamma_{i,2} = 0$ and the single basis element $\gamma_{i,1}$ can for instance be defined such that:

$$\gamma_{i,1} = \begin{cases} \mu_{i,1} & \text{if } \mu_{i,1} \neq 0, \\ \mu_{i,2} & \text{if } \mu_{i,1} = 0 \text{ and } \mu_{i,1} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The set of pairs (i, j) such that $\gamma_{i,j}$ has been assigned a non-zero value by the latter process is denoted by $\mathcal{R}$ and is called *reconstruction set*. By construction, we have $\mathcal{R} \subseteq \{(i,j); \alpha_i \in \mathcal{S} \backslash \{0\}, j \in [1;2]\}$ and $\#\mathcal{R} = RB_{\mathrm{LERS}}$. Moreover, we denote by $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) the subset of $\mathcal{R}$ containing the pairs (i, j) with $j = 1$ (resp $j = 2$). They form a partition of $\mathcal{R}$ and they are the same for any polynomial P of degree lower than or equal to d.

Eventually, for the case $t = 2$, we may conclude that for any polynomial P_a of degree lower than or equal to d:

Fact 1 (memory) To reconstruct $a = P_a(0)$ thanks to the Formula (12), it is sufficient to store the elements $w_{i,j} = \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\gamma_{i,j} a_i) \in \mathbb{F}$ with the $(i, j) \in \mathcal{R}$. Hence, the overall scheme leads to replace the storage of the $(d+1) \times m$ bits needed for the classical reconstruction with Lagrange Interpolation, by the storage of $RB_{\mathrm{LERS}} \times \frac{m}{2}$ bits.

Fact 2 (processing) The reconstruction of $a = P_a(0)$ can always be done thanks to the following formula which is deduced from (9)-(10):

$$a = \nu_1\Big(\sum_{(i,1)\in\mathcal{R}_1} w_{i,1}\Big) + \nu_2\Big(\sum_{(i,2)\in\mathcal{R}_2} w_{i,2} + \sum_{(i,1)\in\mathcal{R}_1,\#\mathcal{B}_i=1} \frac{\mu_{i,2}}{\mu_{i,1}} w_{i,1}\Big) \ . \quad (16)$$

where $\{\nu_1, \nu_2\}$ is the dual basis of $\{\zeta_1, \zeta_2\}$. Note that the coefficients $\mu_{i,2}/\mu_{i,1}$ belong to $\mathbb{F}$ by construction.

Fact 3 (security) The elements $w_{i,j}$ form a sharing of a and this sharing still satisfies the d^{th}-order security property since, by construction, at least $d+1$ shares are necessary to rebuild w.

Remark 1. For the new multiplication protocol exhibited in Section 4.4, we will prefer to use the following equation which is equivalent to (16):

$$P(0) = \nu_1\Big(\underbrace{\sum_{(i,1)\in\mathcal{R}_1} \tau_i \times w_{i,1}}_{\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_1 w)}\Big) + \nu_2\Big(\underbrace{\sum_{(i,2)\in\mathcal{R}_2} w_{i,2}}_{\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_2 w)}\Big) \ . \quad (17)$$

where τ_i is constant with respect to $P(X)$ and is defined such that:

$$\tau_i = \begin{cases} 1 + \frac{\nu_2\mu_{i,2}}{\nu_1\mu_{i,1}} & \text{if } \#\mathcal{B}_i = 1 \\ 1 & \text{otherwise.} \end{cases} \quad (18)$$

In [17, Theorem 10] it is proved that for Reed-Solomon codes $\mathrm{RS}(\mathcal{S}, d+1)$ with $\#\mathcal{S} = n$ and $t = 2$, there always exits an LERS with the bandwidth RB_{LERS} being at most $3n/2n$ (which leads to a reconstruction of a with strictly less than $\frac{3n}{4}m$ bits instead of $(d+1)m$).

As an illustration of the concepts presented in this section, the example given in [17] is detailed in Appendix D. It corresponds to the case of a code $\mathrm{RS}(\mathcal{S}, 9+1)$ over the field $\mathbb{K} \simeq \mathrm{GF}(256)$ with $\mathcal{S} = \{g^0, \ldots, g^{13}\}$ where g is a primitive element of $\mathbb{K}$. This example is completed by `Sage` scripts available at [1].

4.4 Improvement From LERS

To simplify the presentation, we apply our proposal in the particular case where the linear exact repairing codes recalled in Sec. 2.3 are applied for $\mathbb{K} = \mathrm{GF}(2^m)$, $\mathbb{F} = \mathrm{GF}(2^{m/2})$ and $t = 2$. For this parameters, we assume that an LERS exists for a Reed-Solomon code with evaluation set $\mathcal{S} = \{\alpha_1, \ldots, \alpha_{n'}\}$ and order $2d+1$ (this implies that n' is greater than $n = 2d+1$). We assume that the latter LERS has repair bandwidth $RB_{\mathrm{LERS}} < 2(2d+1)$ and we denote by $(\zeta_1, \zeta_2\}$ the basis defined in (8) and by (ν_1, ν_2) the corresponding dual basis over $\mathrm{GF}(2^{m/2})$. For our description we assume that coefficients $\mu_{\alpha_i,j}$ and $\gamma_{i,j}$, and also the reconstruction set $\mathcal{R}$, have been defined as specified in Sec. 4.3. Eventually, to properly define the evaluation of polynomials in $\mathrm{GF}(2^{\frac{m}{2}})[X]$ in the public elements $\alpha_i \in \mathcal{S}$, we shall assume that they belong to $\mathbb{F} = \mathrm{GF}(2^{m/2})$ and, when needed, that their

$\frac{m}{2}$-bit representation is (artificially) extended to an m-bit representation (in $\mathrm{GF}(2^m)$) by simply left-padding with 0 (*e.g.* the element $3 \in \mathrm{GF}(2^{\frac{m}{2}})$ becomes 03 in $\mathrm{GF}(2^m)$).[13]

For our new proposal, we assume that the two values u and v whose multiplication has to be secured are respectively represented by the $(n', d+1)$-polynomial sharings $(P_u(\alpha_i))_{\alpha_i \in \mathcal{S}}$ and $(P_v(\alpha_i))_{\alpha_i \in \mathcal{S}}$ (we hence have $u = P_u(0)$ and $v = P_v(0)$). Note that contrary to the classical SSS sharing, we need n' to be strictly greater than $2d+1$ in order to get an efficient LERS.

Sharing of $w = uv$ in $\mathcal{C}^\star$. As in Algorithm 1, we start by building a sharing $(w_1, \ldots, w_{n'})$ of $w = uv$ in the product code $\mathcal{C}^\star$ by simply defining w_i, for $i \in [1; n']$, such that:

$$w_i = P_u(\alpha_i) \times P_v(\alpha_i) \doteq P_w(\alpha_i)$$

By construction, it may be observed that $\mathcal{C}^\star$ is a sub-code of the Reed-Solomon code with parameters $[n'+1, 2d+1]$; this implies that w can be rebuild from RB_{LERS} coordinates $w_{i,j} \in \mathrm{GF}(2^{m/2})$ of the w_i's, by applying Formula (16) (instead of $2d+1$ elements $w_i \in \mathrm{GF}(2^m)$ when classical Lagrange's interpolation is applied). From the coefficients $\gamma_{i,j}$ associated to our LERS for the code $\mathrm{RS}(n'+1, 2d+1)$ (hence with (i,j) in the reconstruction set $\mathcal{R}$), we deduce the new shares $w_{i,j} \in \mathrm{GF}(2^{m/2})$ from the the shares $w_i \in \mathrm{GF}(2^m)$ by processing[14]:

$$w_{i,j} = \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\gamma_{i,j} w_i) \ . \tag{19}$$

We recall that $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively denote $\{(i,j) \in \mathcal{R}, j = 1\}$ and $\{(i,j) \in \mathcal{R}, j = 2\}$. The set $\{w_{i,j}; (i,j) \in \mathcal{R}_1\}$ may hence be viewed as a sharing of the first coordinate of $P(0)$ in $\mathrm{GF}(2^{m/2})^2$ (aka $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_1 P(0))$ with the notations in Section 2.3), while the set $\{w_{i,j}; (i,j) \in \mathcal{R}_2\}$ is the sharing of the second coordinate (aka $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_2 P(0))$). By construction, the sharing of w defined by evaluating (19) for $(i,j) \in \mathcal{R}$ has size RB_{LERS}.

Now, applying the same ideas as described in previous section:

Reduce dimension of the sharings. We generate two independent Boolean sharings of $\mathbf{0}$ in $\mathrm{GF}(2^{m/2})$ that we respectively denote by $\{c^0_{i,j}; (i,j) \in \mathcal{R}_1\}$ and $\{c^0_{i,j}; (i,j) \in \mathcal{R}_2\}$. Then, we build the two following new sharings of $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_1 P(0))$ and $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_2 P(0))$ respectively:

$$\hat{\boldsymbol{c}}_1 = \{\hat{\boldsymbol{c}}_{1,1}, \hat{\boldsymbol{c}}_{1,2}, \ldots, \hat{\boldsymbol{c}}_{1,\#\mathcal{R}_1}\} = \{\tau_i \times w_{i,j} + c^0_{i,j}; (i,j) \in \mathcal{R}_1\}$$

[13] This introduces the restriction $\#\mathcal{S} < 2^{\frac{m}{2}} - 1$ which has no important impact in practice.

[14] Note that this processing is for free if the field representation of $\mathrm{GF}(2^m)$ has been carefully chosen.

and

$$\hat{\boldsymbol{c}}_2 = \{\hat{\boldsymbol{c}}_{2,1}, \hat{\boldsymbol{c}}_{2,2}, \ldots, \hat{\boldsymbol{c}}_{2,\#\mathcal{R}_2}\} = \{w_{i,j} + c^0_{i,j}; (i,j) \in \mathcal{R}_2\} \ ,$$

where τ_i is defined as in (18). Then, if $\#\mathcal{R}_1$ and/or $\#\mathcal{R}_2$ are greater than $d+1$, we shorten the sharings as described in previous section. For $j = 1, 2$, we get:

$$\hat{\boldsymbol{c}}^-_j = \big(\sum_{i=1}^{\#\mathcal{R}_j - d} \hat{c}_{j,i}, \hat{c}_{j,\#\mathcal{R}_j - d+1}, \cdots, \hat{c}_{j,\#\mathcal{R}_j} \big) \ .$$

Transcodings in $\mathcal{C}$ over GF($2^{\frac{m}{2}}$). Eventually, we conclude the processing as in previous section by transcoding $\hat{\boldsymbol{c}}^-_1$ and $\hat{\boldsymbol{c}}^-_2$ into two new sharings in the Reed-Solomon code with parameters $[n'+1, 2d+1]$ over GF($2^{m/2}$) which themselves form a sharing of w in the Reed-Solomon code with parameters $[n'+1, 2d+1]$ over GF(2^m) (with basis (ν_1, ν_2)).

Completeness. By construction, it may be checked that we have:

$$\sum_i \hat{c}^-_{1,i} = \sum_{i=1}^{\#\mathcal{R}_1 - d} \hat{c}_{1,i} + \hat{c}_{1,\#\mathcal{R}_1 - d+1} + \cdots + \hat{c}_{1,\#\mathcal{R}_1} \tag{20}$$

$$= \sum_{(i,j)\in\mathcal{R}_1} \tau_i \times w_{i,j} + \sum_{(i,j)\in\mathcal{R}_1} c^0_{i,j} = \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\zeta_1 w) \ , \tag{21}$$

which comes at a direct consequence of (17) and the fact that the $c^0_{i,j}$ forms a Boolean sharing of $\mathbf{0}$. Since the same holds for the sharing $\hat{\boldsymbol{c}}^-_2$, we deduce that the input of the first (resp. second) transcoding is a sharing of the first (resp. second) coordinate of w in GF($2^{\frac{m}{2}}$). If one denote by $P_{w,1}(X)$ (resp. $P_{w,2}(X)$) the random degree-d polynomial in GF($2^{\frac{m}{2}}$) corresponding to the sharing at input of the first (resp. second) transcoding, then it may be checked that $P_w(X) \doteq \nu_1 P_{w,1}(X) + \nu_2 P_{w,1}(X)$ is a random degree-d polynomial over GF(2^m) and $(d+1, d+1)$-SSS sharing is given by its evaluation $P_w(X)(\alpha_i) = \nu_1 P_{w,1}(\alpha_i) + \nu_2 P_{w,2}(\alpha_i)$ in the public elements $\alpha_i \in \mathcal{S}$.

Extension to $t > 2$. The construction can be directly generalized to any extension order $t = 2^k$ (*i.e.* under the assumption that there exists an efficient LERS for the Reed-Solomon code $\mathcal{C} = \mathrm{RS}(\mathcal{S}, 2d+1)$ with $\mathbb{K} = \mathrm{GF}(2^m)$ and $\mathbb{F} = \mathrm{GF}(2^{m/t})$). This will lead to the construction of t sharings $\hat{c}^-_1, \hat{c}^-_2, \ldots, \hat{c}^-_t$ instead of 2, and to t transcodings in the code $\mathcal{C}$ over GF($2^{m/t}$). The corresponding reconstruction sets are denoted by $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_t$.

Efficiency. A core observation here is that the size of each sharing $\hat{\boldsymbol{c}}^-_j$ is $\min(\#\mathcal{R}_j, d+1)$. Hence, the complexity of the transcoding step is:

$$\mathtt{Comp}_{\mathrm{new}} = d(d+1)\big(\sum_{j=1}^{t} \min(\#\mathcal{R}_j, d+1)\big) + t(n'-d-1)(d+1)$$

which is, in the least favourable case, comparable to the complexity $\texttt{Comp}_{\text{soa}}$ of the state-of-the-art schemes and which is better, for almost all choices of d and n, if one of the $\mathcal{R}_j$ has cardinality lower than $d+1$.

5 Conclusion

In this paper, we highlighted an important difference between Boolean and Shamir's sharings which implies that, for some signal-to-noise ratio, it is more advantageous for the adversary to observe strictly more than $d+1$ shares. We argue that this difference is related to the existence of so-called exact linear repairing codes, which themselves come with reconstruction formulae that need (much) less information (counted in bits) than Lagrange's interpolation. In particular, this result implies that, contrary to what was believed, the choice of the public points in Shamir's sharing has an impact on the countermeasure strength. In the second part of the paper, we then exhibited a positive impact of the existence of linear exact repairing schemes; we indeed proposed to use them to improve the state-of-the-art multiplication algorithms dedicated to Shamir's sharing. We think that this work opens promising avenues on the design of LERS taking into account the constraints of the SCA contexts.

References

1. Anonymous. Sage Scripts for Examples of Exact Linear Repairing Codes, 2017. Available at `https://cloud.sagemath.com/projects/bae91571-60b3-4815-b911-aa3090b81a47/files/`. Login: `Ches2017sis@gmail.com` and password: ches2017.
2. J. Balasch, S. Faust, and B. Gierlichs. Inner Product Masking Revisited. In E. Oswald and M. 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.
3. A. Battistello, J. Coron, E. Prouff, and R. Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.
4. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1988. ACM.
5. V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li. Optimal repair of MDS codes in distributed storage via subspace interference alignment. *CoRR*, abs/1106.1250, 2011.
6. G. Castagnos, S. Renner, and G. Zémor. High-order masking by using coding theory and its application to AES. In M. Stam, editor, *Cryptography and Coding*

- *14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*, pages 193–212. Springer, 2013.
7. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In *Second AES Candidate Conference – AES 2*, Mar. 1999.
8. J.-S. Coron, E. Prouff, and T. Roche. On the use of shamir's secret sharing against side-channel analysis. In S. Mangard, editor, *CARDIS*, volume 7771 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2012.
9. A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Trans. Inf. Theor.*, 56(9):4539–4551, Sept. 2010.
10. A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.
11. A. Duc, S. Faust, and F. Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In E. Oswald and M. 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 401–429. Springer, 2015.
12. S. Dziembowski and S. Faust. Leakage-resilient circuits without computational assumptions. In R. Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 230–247. Springer, 2012.
13. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified vss and fact-track multiparty computations with applications to threshold cryptography. In *PODC*, pages 101–111, 1998.
14. L. Goubin and A. Martinelli. Protecting aes with shamir's secret sharing scheme. In Preneel and Takagi [29], pages 79–94.
15. V. Grosso, E. Prouff, and F. Standaert. Efficient masked s-boxes processing - A step forward -. In D. Pointcheval and D. 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.
16. V. Grosso and F. Standaert. Masking Proofs are Tight (and How to Exploit it in Security Evaluations). *IACR Cryptology ePrint Archive*, 2017:116, 2017.
17. V. Guruswami and M. Wootters. Repairing reed-solomon codes. In D. Wichs and Y. Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 216–226. ACM, 2016.
18. W. Huang and J. Bruck. Secret Sharing with Optimal Decoding and Repair Bandwidth. Unpblished. Available at `http://www.paradise.caltech.edu/papers/etr135.pdf`., 2016.
19. Y. Ishai, A. Sahai, and D. Wagner. Circuits: Securing Hardware against Probing Attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
20. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
21. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

22. V. Lomné, E. Prouff, M. Rivain, T. Roche, and A. Thillard. How to estimate the success rate of higher-order side-channel attacks. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2014.
23. V. Lomné, E. Prouff, and T. Roche. Behind the Scene of Side Channel Attacks. In *ASIACRYPT 2013*, pages 506–525. Springer, 2013.
24. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, 2007.
25. R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Commun. ACM*, 24(9):583–584, 1981.
26. T. Messerges. Using Second-order Power Analysis to Attack DPA Resistant software. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
27. C. O'Flynn and Z. D. Chen. Chipwhisperer: An open-source platform for hardware embedded security research. Cryptology ePrint Archive, Report 2014/204, 2014. `http://eprint.iacr.org/2014/204`.
28. E. Oswald, S. Mangard, C. Herbst, and S. Tillich. Practical Second-order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
29. B. Preneel and T. Takagi, editors. *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
30. E. Prouff and M. Rivain. Higher-Order Side Channel Security and Mask Refreshing. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013 - 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.
31. E. Prouff and T. Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In Preneel and Takagi [29], pages 63–78.
32. S. Renner. *Protection des algorithmes cryptographiques embarqués*. PhD thesis, Bordeaux, 2013. Available at `https://tel.archives-ouvertes.fr/tel-01149061/document`.
33. T. Roche and E. Prouff. Higher-order glitch free implementation of the AES using secure multi-party computation protocols - extended version. *J. Cryptographic Engineering*, 2(2):111–127, 2012.
34. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
35. K. Shanmugam, D. S. Papailiopoulos, A. G. Dimakis, and G. Caire. A repair framework for scalar MDS codes. *CoRR*, abs/1312.2135, 2013.
36. F.-X. Standaert, N. Veyrat-Charvillon, E. Oswald, B. Gierlichs, M. Medwed, M. Kasper, and S. Mangard. The world is not enough: Another look on second-order dpa. Cryptology ePrint Archive, Report 2010/180, 2010. `http://eprint.iacr.org/`.
37. C. Suh and K. Ramchandran. On the existence of optimal exact-repair MDS codes for distributed storage. *CoRR*, abs/1004.4663, 2010.
38. W. Wang, F. Standaert, Y. Yu, S. Pu, J. Liu, Z. Guo, and D. Gu. Inner product masking for bitslice ciphers and security order amplification for linear leakages. In K. Lemke-Rust and M. Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November*

7-9, 2016, Revised Selected Papers, volume 10146 of *Lecture Notes in Computer Science*, pages 174–191. Springer, 2016.
39. M. Ye and A. Barg. Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 1202–1206. IEEE, 2016.

A Information Extraction from Noisy Observations of 8-bit Values

In the following figure, we plot the number of queries to achieve a success rate of 100% when performing a template attack which tries to recover, from noisy observations, either the full value of the manipulated data a or its 4 least significant bits. The observations ℓ have been simulated to satisfy $\ell = f(a) + b$ where b is drawn from the normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma \in [1, 5]$ and where the deterministic part $f(a)$ has been defined such that $f(a) \doteq \sum_{i=1}^{8} \alpha_i a[i]$ with the α_i generated at random from the normal distribution $\mathcal{N}(1, 0.2^2)$ and kept fixed for all the attacks[15].

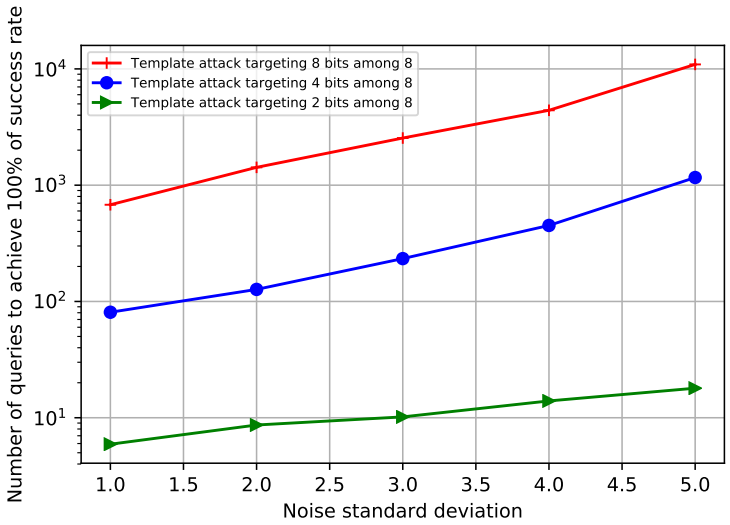

Fig. 6. Number of queries (in y-axis) to achieve a success rate of 100% for the extraction of either 8 bits or 4 bits or 2 bits with respect to the noise standard deviation (in x-axis).

[15] The idea was to simulate a leakage close to the Hamming weight model but with the weights associated to the bit-coordinates of a differing with an average amplitude in $[-0.2, +0.2]$.

B Bounds for the Bandwidth of Exact Linear Repairing codes

When t is sufficiently large, RB_{LERS} is bounded below by $(t \times RL)/(RL - d)$, where RL denotes the *repair locality* and corresponds to the maximum number of $\alpha_i \in \mathcal{S}$ that are required to recover $P(\alpha^\star)$ whatever $\alpha^\star \in \mathcal{S}$. For very large values of t, several schemes have been proposed for which the lower bound is achieved (see *e.g.* [5,37]).

When t is small, it is clear that the lower bound above cannot be achieved since we have $RB_{\text{LERS}} \geqslant d + t$. It must however be observed that this latter bound is much better than the naive one $t \times (d+1)$. Explicit constructions have recently been proposed that lead to good RB_{LERS} in practice (*e.g.* [17], [18] and [39]).

C Formal Definition of an Exact Linear Repairing Schemes

Definition 3 (Linear Exact Repair Scheme). *Let $\mathcal{S} = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ be an evaluation subset of a finite field $\mathbb{K}$ and let $\text{RS}(\mathcal{S}, d+1) \subseteq \mathbb{K}^n$ be the corresponding Reed-Solomon code. A* linear exact repair scheme *for this code over a subfield $\mathbb{F}$ of $\mathbb{K}$ is composed of the following steps:*

- *for each $\alpha^\star \in \mathcal{S}$, and for each $\alpha \in \mathcal{S} \backslash \{\alpha^\star\}$, a set of queries $Q_\alpha(\alpha^\star) \subseteq \mathbb{K}$,*
- *for each $\alpha \in \mathcal{S}$ and any polynomial $P(X) \in \mathbb{K}[X]$ with degree lower than $d+1$, a linear reconstruction algorithm that computes*

$$P(\alpha^\star) = \sum_{i=1}^{t} \lambda_i \times \nu_i \ ,$$

for coefficients $\lambda_i \in \mathbb{F}$ and a basis $\{\nu_1, \nu_2, \cdots, \nu_t\}$ for $\mathbb{K}$ over $\mathbb{F}$, so that the coefficients λ_i are $\mathbb{F}$-linear combinations of the queries in

$$\bigcup_{\alpha \in \mathcal{S}/\{\alpha^\star\}} \{\text{tr}_{\mathbb{K}/\mathbb{F}}(\gamma \times P(\alpha); \gamma \in Q_\alpha(\alpha^\star)\} \ .$$

The *repair bandwith* RB_{LERS} of an LERS is the maximum number of sub-symbols which must be returned by the nodes to recover $P(\alpha^\star)$. It is defined by:

$$RB_{\text{LERS}} = \max_{\alpha^\star \in \mathcal{S}} \sum_{\alpha \in \mathcal{S}/\{\alpha^\star\}} \#Q_\alpha(\alpha^\star) \ .$$

D Example

Let us consider the example given in [17]. Let latter example considers the code $\text{RS}(\mathcal{S}, 9+1)$ over the field $\mathbb{K} \simeq \text{GF}(256)$ with $\mathcal{S} = \{g^0, \ldots, g^{13}\}$ where g is a

primitive element of $\mathbb{K}$ (which implies $n = 14$ and $d = 9$ for the notations used in previous section). It is assumed that $\alpha^\star = g^0 = 1$ and $\alpha_i = g^i$ for $i \in [1; 13]$ and $t = 2$ (hence $\mathbb{F} \simeq \mathrm{GF}(16)$). To build the family of two polynomials satisfying (7) and (8), [17] proposes a pretty restrictive (but yet effective) approach: they randomly generate two polynomials $p_1(X)$ and $p_2(X)$ of degree $n - d - 1 = 3$ with roots[16] in $\mathcal{S}$ and test (1) if the space spanned by $p_1(\alpha^\star)$ and $p_2(\alpha^\star)$ has full rank and (2) if the sum of the dimensions of the spaces spanned by the pairs $(p_1(\alpha_i), p_2(\alpha_i))$ when i ranges in $[1; 13]$ is lower than some threshold judged as good (at least smaller than 20 in order to get a reconstruction better than the simple interpolation which needs 10 bytes, or identically 20 nibbles, corresponding to 10 polynomial evaluations). In the paper, the authors fix the threshold to 16.

For this example, we have $\zeta_1 = p_1(1)$ and $\zeta_2 = p_2(1)$. For the field representation $\mathrm{GF}(256) \simeq \mathrm{GF}(2)[X]/(X^8 + X^4 + X^3 + X^2 + 1)$, we found that the 3-degree polynomials $p_1(X) = X^3 + 38X^2 + 200X + 29$ and $p_2(X) = X^3 + 105X^2 + 213X + 58$ enables to achieve our fixed threshold (16).

	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mu_{\alpha_i,1}$	0	0	76	68	0	238	57	157	220	80	115	204	131
$\mu_{\alpha_i,2}$	248	21	120	0	127	0	211	56	0	171	33	147	45

By using the coefficients ζ_1, ζ_2 and $(\mu_{\alpha_i,j})_{i\in[1;13],j\in[1;2]}$ as specified in Formula (10) and (11), it is possible to reconstruct $P(1)$ for any polynomial $P(X)$ of degree lower than or equal to 9 with only 64 bits of information[17] on the 14 shares $P(\alpha_i)$ (instead of the 80 bits required by using Lagrange's Interpolation Formula). Indeed, from the table above and Equations (10) and (11), we get that:

- the 4-bit field elements $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\mu_{\alpha_i,j} P(\alpha))$ corresponding to $\mu_{\alpha_i,j} = 0$ do not need to be stored,
- for all pairs of 4-bit field elements $\mu_{\alpha_i,1}$ and $\mu_{\alpha_i,2}$ which are linearly dependent over GF(16) (these are the elements with light-gray background), it is only necessary to save a single trace evaluation, say $\mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\mu_{\alpha,1} P(\alpha))$, and the second one can be deduced by simply multiplying it by the value $\tau_i \in \mathrm{GF}(16)$ such that $\tau_i \mu_{\alpha_i,1} = \mu_{\alpha_i,2}$ (which necessarily exists since they linearly dependent over GF(16)).

With the notations defined in Section 4.3, a reconstruction set for the code $\mathrm{RS}(\mathcal{S}, 9 + 1)$ is given by:

$$\mathcal{R}_1 = \{(3,1), (4,1), (6,1), (7,1), (8,1), (9,1), (10,1), (11,1), (12,1), (13,1)\}$$

and

$$\mathcal{R}_2 = \{(1,2), (2,2), (5,2), (8,2), (11,2), (13,2)\}$$

[16] The intuition is that having polynomials such that at least $p_1(\alpha_i) = 0$ or $p_2(\alpha_i) = 0$ for some $i \in [1; 13]$ automatically reduces the dimension of the corresponding set $\{p_j(\alpha_i) : j \in [1; 2]\}$, which helps to satisfy (8).

[17] always the same whatever $P(X)$

and $\mathcal{R} = \mathcal{R} \cup \mathcal{R}_2$.

Eventually, we may conclude that for any polynomial $P(X)$ of degree lower than or equal to 9:

Fact 1 (memory) To reconstruct $w = P(1)$, it is sufficient to store the 4-bit elements $w_{i,j} = \mathrm{tr}_{\mathbb{K}/\mathbb{F}}(\gamma_{i,j} P(\alpha))$ with $(i,j) \in \mathcal{R}$. Since $\#\mathcal{R} = RB_{\mathrm{LERS}} = 16$, the overall scheme leads to save $10 * 4 = 40$ bits for the storage of the sharing of $P(1)$ in $\mathrm{RS}(\mathcal{S}, 9+1)$ (and the set $\mathcal{R}$ is the same for any $P(X)$).

Fact 2 (processing) The reconstruction of $P(1)$ can always be done by evaluating (16) at the cost of $RB_{\mathrm{LERS}} = 16$ additions over GF(16) and $2 + 4$ multiplications by a constant scalar over GF(16).