Polynomial Evaluation and Side Channel Analysis

Claude Carlet, Emmanuel Prouff

Polynomial Evaluation and Side Channel Analysis

Claude Carlet[1] and Emmanuel Prouff[2,3]

[1] LAGA, UMR 7539, CNRS, Department of Mathematics,
University of Paris XIII and University of Paris VIII
claude.carlet@univ-paris8.fr

[2] ANSSI, FRANCE
emmanuel.prouff@ssi.gouv.fr

[3] POLSYS, UMR 7606, LIP6,
Sorbonne Universities, UPMC University Paris VI

Abstract. Side Channel Analysis (SCA) is a class of attacks that exploits leakage of information from a cryptographic implementation during execution. To thwart it, masking is a common countermeasure. The principle is to randomly split every sensitive intermediate variable occurring in the computation into several shares and the number of shares, called the *masking order*, plays the role of a security parameter. The main issue while applying masking to protect a block cipher implementation is to specify an efficient scheme to secure the s-box computations. Several masking schemes, applicable for arbitrary orders, have been recently introduced. Most of them follow a similar approach originally introduced in the paper of Carlet *et al* published at FSE 2012; the s-box to protect is viewed as a polynomial and strategies are investigated which minimize the number of field multiplications which are not squarings. This paper aims at presenting all these works in a comprehensive way. The methods are discussed, their differences and similarities are identified and the remaining open problems are listed.

1 Introduction

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

Many implementations of block ciphers have been practically broken by continuous side-channel analysis — see for instance [40, 9, 45, 42] — and securing them has been a longstanding issue for the embedded systems industry. A sound

approach is to use *secret sharing* [6, 62], often called *masking* in the context of side-channel attacks. This approach consists in splitting each sensitive variable Z of the implementation (*i.e.* each variable depending on the secret key, or better for the attacker, on a small part of it, and of public data such as the plaintext) into $d+1$ shares $M_0, \ldots, M_d$, where d is called the *masking order*, such that Z can be recovered from these shares but no information can be recovered from less than $d+1$ shares. It has been shown that the complexity of mounting a successful side-channel attack against a masked implementation increases exponentially with the masking order [12, 52, 24]. Starting from this observation, the design of efficient *masking schemes* for different ciphers has become a foreground issue. When specified *at higher order* d, such a scheme aims at specifying how to update the sharing of the internal state throughout the processing while ensuring that (1) the final sharing corresponds to the expected ciphertext, and (2) the dth-order security property is satisfied. The latter property, which is equivalent to the *probing security* model introduced in [35], states that every tuple of d or less intermediate variables is independent of the secret parameter of the algorithm. When satisfied, it guarantees that no attack of order lower than or equal to d is possible.

Most block cipher structures (*e.g.* `AES` or `DES`) are iterative, meaning that they apply several times a same transformation, called *round*, to an internal state initially filled with the plaintext. The round itself is composed of a key addition, one or several linear transformation(s) and one or several non-linear transformation(s) called s-box(es). Key addition and linear transformations are easily handled as linearity enables to process each share independently. The main difficulty in designing masking schemes for block ciphers hence lies in masking the s-box(es).

During the last decade, several attempts have been done to define higher-order schemes working for any order d. The proposals [1, 2, 17, 22, 30, 61] either did unrealistic assumptions on the adversary capabilities or have been broken in subsequent papers [53, 51, 54, 21, 20]. Actually, there currently exist four masking schemes which have not been broken (and even benefit from formal security proofs):

- The first method is due to Genelle *et al* [28] and consists in mixing additive and multiplicative sharings. This scheme is primarily dedicated to the `AES` sbox and seems difficult to generalize efficiently to other s-boxes (not affinely equivalent to a power function).
- The second masking scheme is due to Prouff and Roche [55] and it relies on solutions developed in secure *multi-party computation* [3]. It is much less efficient than the other schemes (see *e.g.* [32]) but, contrary to them, remains secure even in presence of hardware glitches [41].
- The third approach has been recently proposed by Coron in [18]. The core idea is to represent the s-box by several look-up tables which are regenerated from fresh random masks and the s-box truth table, each time a new s-box processing must be done. It extends the *table re-computation technique* introduced in the original paper by Kocher *et al* [40]. The security of Coron's

scheme against higher-order SCA is formally proven under the assumption that the variable shares M_i leak independently. Its asymptotic timing complexity is quadratic in the number of shares and can be applied to any s-box. However, the RAM memory consumption to secure (at order d) an s-box with input (resp. output) dimension n (resp. m) is $m(d+1)2^n$ bits, which can quickly exceed the memory capacity of the hosted device (*e.g.* a smart card).

– The three methods recalled in previous paragraph have important limitations which impact their practicability. Actually, when the s-box to secure is not a power function and has input/output dimensions close to 8, only the following fourth approach is practical when d is greater than or equal to 3. This approach, proposed in [10], generalizes the study conducted in [57] for power functions (the latter work is itself inspired by techniques proposed for Boolean circuits by Ishai, Sahai and Wagner in [35]). The core idea is to split the s-box processing into a short sequence of field multiplications and $\mathbb{F}_2$-linear operations, and then to secure these operations independently. The complexity of the masking schemes for the multiplication and for an $\mathbb{F}_2$-linear operation[4] is $O(d^2)$ and $O(d)$ respectively. Moreover, for dimensions n greater than 6, the constant terms in these complexities are (usually) significantly greater for the multiplication than for the $\mathbb{F}_2$-linear operations. Based on this observation, the authors of [10] propose to look for operations sequences (aka s-box representations) that minimize the number of field multiplications which are not $\mathbb{F}_2$-linear[5] (this kind of multiplication shall be called *non-linear* in this paper). This led them to introduce the notion of *s-box masking complexity*, which corresponds to the minimal number of non-linear multiplications needed to evaluate the s-box. This complexity is evaluated for any power function defined in $\mathbb{F}_{2^n}$ with $n \leq 10$ (in particular, the complexity of $x \in \mathbb{F}_{2^8} \mapsto x^{254}$, which is the non-linear part of the AES s-box, is shown to be equal to 4). Tight upper bounds on the masking complexity are also given for any random s-box. The analysis in [10] has been further improved by Roy and Vivek in [59], where it is in particular shown that the masking complexity of the DES s-boxes is lower bounded by 3. The authors of [59] also present a polynomial evaluation method that requires 7 non-linear multiplications (instead of 10 in [10]). Another improvement of [10] has been proposed in [15], where it is shown that it is possible to improve the processing of the non-linear multiplications which have the particular form $x \times \mathcal{L}(x)$ with $\mathcal{L}$ being $\mathbb{F}_2$-linear. Recently, Coron, Roy and Vivek proposed an heuristic method which may be viewed as an extension of the ideas developed in [19] and [59]. For all the tested s-boxes it is at least as efficient as the previous methods and it often requires less non-linear mul-

[4] A function f is $\mathbb{F}_2$-linear if it satisfies $f(x \oplus y) = f(x) \oplus f(y)$ for any pair (x, y) of elements in its domain. This property must not be confused with $\mathbb{F}_{2^m}$-linearity of a function, where m divides n and is larger than 1, which is defined such that $f(ax \oplus by) = af(x) \oplus bf(y)$, for every $a, b \in \mathbb{F}_{2^m}$. An $\mathbb{F}_{2^m}$-linear function is $\mathbb{F}_2$-linear but the converse is false in general.

[5] A multiplication over a field of characteristic 2 is $\mathbb{F}_2$-linear if it corresponds to a squaring.

tiplications (*e.g.* 4 for the DES s-boxes). Eventually, in [11], Carlet, Prouff, Rivain and Roche continued the generalization of [10] by proposing to split the evaluation of any s-box into a short sequence of evaluations of polynomial functions with upper bound algebraic degree. It is for instance shown that the processing of any s-box of dimension $n = 8$ can be split into 11 evaluations of quadratic functions, or into 4 evaluations of cubic functions. For the latter evaluations of low degree polynomials, the authors propose several methods, among which an adaptation of CRV which is more efficient than the original one when the degree is low.

The purpose of this paper is to give an overview of the results presented in the sequence of works [10, 15, 16, 31, 59] and we also prove that the masking scheme introduced in [15] for functions in the form $x \times \mathcal{L}(x)$ can be extended to any function of algebraic degree 2. Since the work [11] was published several months after the writing of this paper, it is not fully detailed here, except in Section 3.4 where some results are discussed.

2 Securing Elementary Operations over Finite Fields

Except the masking schemes by Genelle *et al* [28] and by Coron [18] (which cannot be applied to any s-box for the first one, or has a too important RAM memory complexity for the second one), the state-of-the-art masking schemes [10, 16, 31, 55, 59] all follow the same principle: the sbox evaluation is split into a sequence of so-called *elementary operations* which are independently protected thanks to dedicated masking schemes. The set of elementary operations contains the field additions and multiplications and, for reasons that will be exposed in this section, it also includes all affine and quadratic transformations. In the following, we recall the secure masking schemes which have been introduced in the literature to process elementary operations. When defined with respect to an operation (aka transformation) f, such a scheme takes at input a $(d+1)^{\text{th}}$-order sharing of f's input(s) and returns a $(d+1)^{\text{th}}$-order sharing of its output, while ensuring that any d-tuple of intermediate results during the processing is independent of the unshared input.

2.1 Securing Multiplications in Finite Fields

Let us first start the section with a few basics on finite fields multiplications.

Basics on Multiplication Processing. Different time/memory trade-offs exist in the literature for implementing multiplications. For hardware implementations and large dimensions n, several works have been published among which the Omura-Massey method [47], the Sunar-Koc method [64, 67], the Karatsuba algorithm [36], etc. For software implementations in small dimensions (*e.g.* $n \leqslant 10$), the number of pertinent possibilities is reduced. We recall them in the following and we give their time/memory complexities (time complexities are given

in terms of number of cycles). For illustration, we also give a pseudo-code. We moreover assume that the multiplication in $\mathbb{F}_{2^n}$ corresponds to some irreducible polynomial $p(X)$ of degree n over $\mathbb{F}_2$ (*i.e.* we use the representation $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[X]/(p(X))$ or $\mathbb{F}_{2^n} \simeq \{\alpha^i; i \in [0; n-2]\} \cup \{0\}$ where α is a primitive element, root of $p(X)$).

- The most efficient multiplication method in terms of timing, and the most costly in terms of memory, is based on a complete tabulation of the processing(s). The calculation of $c = a \times b$ in $\mathbb{F}_{2^n}$ is done by reading the content of a table $multFn$ in `ROM` containing all the pre-computed results. The size of the table is $n2^{2n}$ bits and the timing of the operation is constant, around 5 cycles depending on the device architecture.

$$c = multFn[a, b]$$

- The most efficient in terms of memory, and the most costly in terms of timing, is the direct processing of the multiplication in $\mathbb{F}_{2^n}$. The memory consumption is reduced to 0 but the timing complexity is $O(n^{\log_3(2)})$ with important constants. The latter complexity is achieved thanks to Karatsuba's method. The core idea of this method is that the product $(a_hY + a_l) \times (b_hY + b_l)$, where a_h, a_l, b_h, b_l live in some ring R, say of characteristic 2, can be computed with 3 multiplications and 4 additions in R, thanks to the following processing decomposition called 2-*segment Karatsuba's method*:

$$(a_hY + a_l) \times (b_hY + b_l) = c_hY^2 + c_{hl}Y + c_l \ ,$$

  where $c_h \doteq a_h \times b_h$, $c_l \doteq a_l \times b_l$ and $c_{hl} \doteq (a_h + a_l) \times (b_h + b_l) - c_h - c_l$. With the formula above, two elements a and b of $\mathbb{F}_{2^n}$ (viewed as polynomials over $\mathbb{F}_2$) can be rewritten and multiplied using the formula:

$$(a_hX^m + a_l) \times (b_hX^m + b_l) = c_hX^{2m} + c_{hl}X^m + c_l \ , \qquad (1)$$

  where a_h, a_l, b_h, b_l, c_{hl}, c_h and c_l are polynomials of degree lower than or equal to $m = \lceil \frac{n}{2} \rceil$. The polynomials c_i are computed by applying the Karatsuba method to the polynomials a_i and b_i as single coefficients and adding coefficients of common powers of X together. Formula (1) is afterwards repeated recursively, either until getting multiplications in $\mathbb{F}_2$ only or until getting low-cost multiplications (*e.g.* because they are tabulated). We will call r-Karatsuba (or $\mathtt{K}_{n,r}$ for short), a multiplication processing where Karatsuba's method is applied r times recursively. Eventually, the reduction by the polynomial $p(X)$ can be interleaved to get the field multiplication.
- The *log-alog* method is a compromise between the two previous methods. Its memory complexity is $n2^{n+1}$ bits and its timing complexity is constant with respect to n. It assumes that the functions $log : x \in \mathbb{F}_{2^n} \mapsto i = \log_\alpha(x)$ and $alog : i \mapsto x = \alpha^i$ have been tabulated in `ROM`. The processing of $a \times b$ then simply consists in processing:

$$c = alog[(log[a] + log[b]) \bmod 2^n - 1] \ .$$

It may be observed that this addition modulo $2^n - 1$ can be processed on n-bit architecture by simply adding $log[a]$ to $log[b]$ (modulo 2^n) and by adding to the result the carry which has possibly been raised (if $log[a]+log[b] \geq 2^n$).

– Another compromise is obtained thanks to the so-called *Tower Fields* approach (see *e.g.* [34, 60]). It can be applied when n is even (*i.e.* $n = 2m$ with $m \in \mathbb{N}$) and first consists in representing $\mathbb{F}_{2^n}$ has a degree-2 extension of $\mathbb{F}_{2^m}$, allowing to perform the computations in $\mathbb{F}_{2^m} \simeq \mathbb{F}_2[X]/(p'(X))$ instead of $\mathbb{F}_{2^n}$. Concretely, the elements of $\mathbb{F}_{2^n}$ are viewed as elements of $\mathbb{F}_{2^m}[X]/(p''(X))$, where $p''(X)$ is a degree-2 polynomial irreducible over $\mathbb{F}_{2^m}$. The field isomorphism mapping an element $a \in (\mathbb{F}_2[X]/p(X))$ into the pair $(a_h, a_l) \in (\mathbb{F}_{2^m}[X]/p''(X))$ is denoted by L. Assuming that the polynomial $p''(X)$ takes the form $X^2 + X + \beta$ (which is always possible thanks to a scaling on X and normalization of the polynomial), the multiplication $a \times b$ is then executed by the following sequence of operations:

$$\begin{array}{rcl} (a_h, a_l) & \leftarrow & L(a) \\ (b_h, b_l) & \leftarrow & L(b) \\ c_l & \leftarrow & a_h \times b_h \times \beta + a_l \times b_l \\ c_h & \leftarrow & (a_h + a_l) \times (b_h + b_l) - a_l \times b_l \\ c & \leftarrow & L^{-1}(c_h, c_l) \end{array}$$

Actually, the technique can be applied to decompose the multiplications in any subfield $\mathbb{F}_{2^m}$ such that 2^r divides n and $m = \frac{n}{2^r}$. We will call r-Tower (or $\texttt{Tow}_{n,r}$ for short), a multiplication processing where the Tower Fields approach is applied downto $\mathbb{F}_{2^m}$. It may be observed that this multiplication method combines the specificity of Tower fields and Karatsuba's method (one could also use Toom-Cook's multiplication [14, 65] instead but it is only advantageous for high dimensions which is out of scope here). In the following, we assume that β is chosen such that the cost of the multiplication by β is negligible.

We sum-up hereafter the complexities of the listed multiplication methods in terms of memory consumption (`ROM` in bits), elementary field additions (`ADD`) and calls to look-up tables (`LUT`). For Karatsuba and Tower Fields approaches, we give the complexities depending on whether the the multiplications in the final subfield are performed with the log-alog method or is tabulated in `ROM`. Moreover, for simplicity reasons, complexities are given in the case where n is a power of 2.

Method	ROM (in bits)	ADD	LUT
Global Look-Up Table ($\texttt{LUT}_n$)	$n \times 2^{2n}$	0	1
Log-ALog Method ($\texttt{LAL}_n$)	$3n \times 2^n$	2	3
r-Karatsuba ($\texttt{K}_{n,r}$) + $\texttt{LUT}_{\frac{n}{2^r}}$	$\frac{n}{2^r} \times 2^{\frac{2n}{2^r}}$	$2 \times (3^r - 1)$	$3 \times \frac{3^r-1}{2}$
r-Karatsuba ($\texttt{K}_{n,r}$) + $\texttt{LAL}_{\frac{n}{2^r}}$	$3\frac{n}{2^r} \times 2^{\frac{n}{2^r}}$	$5 \times (3^r - 1)$	$9 \times \frac{3^r-1}{2}$
r-Tower ($\texttt{Tow}_{n,r}$) + $\texttt{LUT}_{\frac{n}{2^r}}$	$\frac{n}{2^r} \times 2^{\frac{2n}{2^r}} + 2n \times 2^n$	$2 \times (3^r - 1)$	$3 \times (3^r - 1)$
r-Tower ($\texttt{Tow}_{n,r}$) + $\texttt{LAL}_{\frac{n}{2^r}}$	$3\frac{n}{2^r} \times 2^{\frac{n}{2^r}} + 2n \times 2^n$	$6 \times (3^r - 1)$	$5 \times (3^r - 1)$

Table 1. Complexities of multiplication methods

We give hereafter some examples of costs (in cycles) of the elementary operations listed in Table 1 when performed on `8051` and `AVR` chip micro-controllers. For simplicity, we assume that the operations are performed whose bit-length is below that of the processor architecture:

- Addition: 1 cycle (for `8051` and `AVR`).
- Multiplication: 25 cycles (`8051`), 36 cycles (`AVR`).
- LUT call: $1 - 4$ cycle(s) (`8051`), $3 - 7$ cycles (`AVR`).

State-of-the art methods proposed to secure a finite field multiplication between two elements a and b are general and apply similarly for all the multiplication techniques previously recalled. The choice of the latter technique will however impact the practical cost of the scheme (in terms of both memory and cycles count). This explains why a security designer will often favour one of these techniques according to some pre-defined timing/memory trade-off chosen with respect to the context (application, device, cost of the random values generation, etc.). For instance, if the `ROM` memory is not a constrained resource (and/or if the dimension n is small, *e.g.* lower than 5), then the field multiplication can be tabulated and all the operations $\times$ in the hereafter schemes will simply consist in a `LUT` call (which costs around 4 cycles). At the opposite, when the `ROM` memory is a constrained resource (and/or the dimension n is between 5 and 8), then the multiplications can be performed thanks to the log-alog method (in this case each of them will cost around 25 – or 35 – cycles).

Masking Schemes for Finite Multiplications (additive sharing). When the inputs a and b are additively shared into $(a_0, a_1, \cdots, a_d)$ and $(b_0, b_1, \cdots, b_d)$ respectively, a straightforward solution consists in applying the following scheme:

Algorithm 1: Higher-Order Masking Scheme for the Multiplication (Additive Sharing)

Input : a $(d+1)^{\text{th}}$-order sharing $(a_0, a_1, \cdots, a_d)$ and $(b_0, b_1, \cdots, b_d)$ of a and b in $\mathbb{F}_{2^n}$
Output: a $(d+1)^{\text{th}}$-order sharing $(c_0, c_1, \cdots, c_d)$ of $c = a \times b$

Randomly generate $(d+1)^2$ elements $r_{ij} \in \mathbb{F}_{2^n}$ such that $\sum_{i \in [0..d]} r_{ij} = 0$ for every $j \leqslant d$
for $i = 0$ **to** d **do**
  $c_i = 0$
  for $j = 0$ **to** d **do**
    /* Construct $c_i = \sum_j a_i \times b_j + \sum_j r_{ij}$ */
    $c_i \leftarrow c_i + a_i \times b_j + r_{ij}$
return $(c_0, c_1, \cdots, c_d)$

Algorithm 1 has been proved to be d^{th}-order secure in [26]. In [35], the authors show that the number of random values r_{ij} can be reduced to $d(d-1)/2$ with no impact on the security. Initially, the scheme was presented over $\mathbb{F}_2$ and it was generalized to any finite field in [57]. The improved scheme, recalled hereafter, involves $2d(d+1)$ additions and $(d+1)^2$ multiplications in $\mathbb{F}_{2^n}$.

Algorithm 2: Improved Higher-Order Masking Scheme for the Multiplication (Additive Sharing)

Input : a $(d+1)^{\text{th}}$-order sharing $(a_0, a_1, \cdots, a_d)$ and $(b_0, b_1, \cdots, b_d)$ of a and b in $\mathbb{F}_{2^n}$
Output: a $(d+1)^{\text{th}}$-order sharing $(c_0, c_1, \cdots, c_d)$ of $c = a \times b$

Randomly generate $d(d+1)/2$ elements $r_{ij} \in \mathbb{F}_{2^n}$ indexed such that $0 \leqslant i < j \leqslant d$
for $i = 0$ **to** d **do**
  for $j = i+1$ **to** d **do**
    $r_{j,i} \leftarrow (r_{i,j} + a_i \times b_j) + a_j \times b_i$
for $i = 0$ **to** d **do**
  $c_i \leftarrow a_i \times b_i$
  for $j = 0$ **to** d, $j \neq i$ **do**
    $c_i \leftarrow c_i + r_{i,j}$
return $(c_0, c_1, \ldots, c_d)$

Masking Schemes for Finite Multiplications (polynomial sharing). When polynomial masking/sharing [62] is used, an alternative to Algorithm 2 exists which has been proposed by Ben-Or *et al* in [4]. The complexity of the latter algorithm in terms of additions and multiplications is $O(d^3)$ and its application is more complex than Algorithm 2 (see [46]). As explained in [55], it however stays secure even in presence of glitches [43] and, compared to additive sharing, it offers better resistance against unbounded adversaries (namely adversaries who can get noisy observations on all the shares of a). Before recalling Ben-Or's *et al* algorithm, let us give some basics about Shamir's polynomial sharing.

In [62] Shamir has introduced a simple and elegant way to split a secret $a \in \mathbb{F}_{2^n}$ into a well chosen number ℓ of shares such that no tuple of shares with cardinality lower than d depends on a. Shamir's protocol consists in generating a degree-d polynomial with coefficients randomly generated in $\mathbb{F}_{2^n}$, except the constant term which is always fixed to a. In other terms, Shamir proposes to associate a with a polynomial $P_a(X)$ defined such that $P_a(X) = a + \sum_{i=1}^{d} u_i X^i$,

where the u_i denote random coefficients. Then, $\ell \geqslant d$ distinct non-zero elements $\alpha_0, \ldots, \alpha_{\ell-1}$ are publicly chosen in $\mathbb{F}_{2^n}$ and the polynomial $P_a(X)$ is evaluated in the α_i to construct a so-called (ℓ, d)-*sharing* $(a_0, a_1, \cdots, a_{\ell-1})$ of a such that $a_i = P_a(\alpha_i)$ for every $i \in [0..\ell-1]$. To re-construct a from its sharing, polynomial interpolation is first applied to re-construct $P_a(X)$ from its ℓ evaluations a_i. Then, the polynomial is evaluated in 0. Those two steps indeed lead to the recovery of a since, by construction, we have $a = P_a(0)$. Actually, using Lagrange's interpolation formula, the two steps can be combined in a single one thanks to the following equation:

$$a = \sum_{i=0}^{\ell-1} a_i \cdot \beta_i \ , \tag{2}$$

where the constants β_i are defined as follows:

$$\beta_i := \prod_{k=0, k\neq i}^{\ell-1} \frac{\alpha_k}{\alpha_i + \alpha_k} \ .$$

Remark 1. The β_i can be precomputed once for all and can hence be considered as public values. They can moreover be also considered as the evaluation in 0 of the polynomials:

$$\beta_i(x) := \prod_{k=0, k\neq i}^{\ell-1} \frac{x + \alpha_k}{\alpha_i + \alpha_k} \ .$$

To securely process the multiplications of two values a and b represented by polynomial sharings, Ben-Or *et al* have introduced a protocol in the context of the Multy-Party Computation Theory [4]. For this protocol to work, the number of shares n per variable must be at least $2d + 1$ and for $\ell = 2d + 1$, it is proved that it satisfies a security property encompassing the d^{th}-order SCA security. We give hereafter the adaptation of [4] in the SCA context as proposed in [55, 58][6].

[6] The protocol is an improved version of the protocol originally proposed by Ben-Or *et al* [4], due to Gennaro *et al* in [29].

Algorithm 3: Higher-Order Masking Scheme for the Multiplication (Polynomial Sharing)

Input : two integers ℓ and d such that $\ell \geq 2d+1$, the (ℓ, d)-sharings $(a_i)_i = (P_a(\alpha_i))_i$ and $(b_i)_i = (P_b(\alpha_i))_i$ of a and b respectively.
Output: the (ℓ, d)-sharing $(P_c(\alpha_i))_i$ of $c = a \cdot b$.
Public : the ℓ distinct points α_i, the interpolation values $(\beta_0, \cdots, \beta_{\ell-1})$

for $i = 0$ **to** $\ell - 1$ **do**
  $w_i \leftarrow P_a(\alpha_i) \cdot P_b(\alpha_i)$

/* Compute a sharing $(Q_i(\alpha_j))_{j \leq d}$ of w_i with $Q_i(X) = w_i + \sum_{j=1}^{d} a_j \cdot X^j$ */
for $i = 0$ **to** $\ell - 1$ **do**
  for $j = 1$ **to** d **do**
    $a_j \leftarrow \mathtt{rand}(\mathbb{F}_{2^n})$
  for $j = 0$ **to** $\ell - 1$ **do**
    $Q_i(\alpha_j) \leftarrow w_i + \sum_{k=1}^{d} a_k \cdot \alpha_j^k$

/* Compute the share $c_i = P_c(\alpha_i)$ for $c = a \cdot b$ */
for $i = 0$ **to** $\ell - 1$ **do**
  $c_i \leftarrow \sum_{j=0}^{\ell-1} Q_j(\alpha_i) \cdot \beta_j$

return $(c_i)_i$

The completeness of Algorithm 3 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 $\ell = 2d+1$ (which is the parameter choice which optimizes the security/efficiency overhead), the complexity of Algorithm 3 in terms of additions and multiplications is $\mathcal{O}(d^3)$. In [21], it is reduced to $O(d^2 \log d)$, essentially by computing polynomial evaluations with a Discrete Fourier Transform as proposed in [68] instead of a naive evaluation[7]. In [33], Grosso and Standaert apply a classical technique from multi-party computation, called *packet secret sharing* and introduced by Franklin and Yung [27], which essentially consists in sharing several secrets with the same polynomial. This technique is of interest when several multiplications, say t, between secrets must be secured. In such a case, the achieved complexity is $O((t+d)^3)$ instead of $O(td^3)$, which implies a complexity improvement if d is greater than $t(t^{\frac{1}{3}} - 1)^{-1}$.

Masking Schemes for Finite Multiplications (sharing by linear codes). As initially observed by Massey [44], there is an equivalence between the existence of linear sharing schemes and the existence of linear codes with certain parameters. Indeed, the set $\{(a, a_0, a_1, \cdots, a_{\ell-1}) \in \mathbb{F}_{2^n}^{\ell+1}\}$ defined by the linear sharings of the elements $a \in \mathbb{F}_{2^n}$ into $(a_0, \cdots, a_{\ell-1}) \in \mathbb{F}_{2^n}^{\ell+1}$ is a subspace of $\mathbb{F}_{2^n}^{\ell+1}$ (aka a linear code). Reciprocally, from any linear $[\ell+1, k, d]$-code[8] C such that the corresponding dual code $C^\perp$ has a distance $d^\perp$ satisfying $d^\perp \geqslant 2$, one can define a linear ℓ-sharing over $\mathbb{F}_{2^n}$. If G denotes the generator matrix of C in *systematic* form (*i.e.* $G = [I_{\ell+1} \mid M]$ where $I_{\ell+1}$ is the

[7] such improvement was already known in the context of multi-party computation [23].

[8] where $\ell+1$ corresponds to the code length and where k (resp. d) denotes its dimension (resp. distance).

$(\ell+1)$-dimensional identity matrix over $\mathbb{F}_{2^n}$), then the sharing $(a_0, a_1, \cdots, a_{\ell-1})$ of a is built from a $(\ell-1)$-tuple of random values $(r_0, r_1, \cdots, r_{\ell-1})$ such that $(a, a_0, a_1, \cdots, a_{\ell-1}) = (a, r_0, \cdots, r_{\ell-1}) \times G$. The reconstruction of a from its sharing $(a_0, \cdots, a_{\ell-1})$ is obtained by processing the product scalar between the latter vector and a so-called *reconstruction vector* $(\beta_0, \cdots, \beta_{\ell-1})$ whose definition depends on the linear code (*e.g.* it corresponds to the first row of the inverse of the Vandermonde (ℓ, ℓ)-matrix $(\alpha_i^{j+1})_{0 \leqslant i,j < \ell}$ in the case of Shamir's polynomial sharing (2)). It can moreover be proved that the sharing defined in such a way defeats any side channel attack of order lower than or equal to $d^{\perp} - 2$ [13]. The thesis of Renner [56] is dedicated to this subject: for all the studied linear sharings (deduced from linear codes) the proposed multiplication schemes have complexity $O(d^3)$. In the particular case of Shamir's polynomial sharing, new methods are however proposed that enable to decrease the constant terms in this complexity and to get interesting practical timing complexity improvements (compared to the methods proposed in [4, 29]).

2.2 Securing Affine Transformations

For $\mathbb{F}_2$-affine transformations, defining a higher-order masking scheme is straightforward. If $(a_0, \cdots, a_d) \in \mathbb{F}_{2^n}^{d+1}$ denotes the *additive sharing* of an intermediate variable $a \in \mathbb{F}_{2^n}$ (*i.e.* the a_i are randomly generated such that $a = \sum_{i \in [0..d]} a_i$) and $\mathcal{A}$ denotes the affine transformation to securely apply on a, then the following simple scheme may be involved. It essentially applies the affine transformation $\mathcal{A}$ to each share of a:

Algorithm 4: Higher-Order Masking Scheme for Affine Transformation (Additive Sharing)

Input : a $(d+1)^{\text{th}}$-order sharing $(a_0, a_1, \cdots, a_d)$ of a, an affine transformation $\mathcal{A}$
Output: a $(d+1)^{\text{th}}$-order sharing $(c_0, c_1, \cdots, c_d)$ of $c = \mathcal{A}(a)$

```
for i = 0 to d do
    c_i ← A(a_i)
if d is odd then
    c_0 ← c_0 + A(0)
return (c_0, c_1, ..., c_d)
```

The same scheme can be straightforwardly extended to any group law and any function $\mathcal{A}$ which is affine for the latter law (see *e.g.* [26]). It can moreover be extended when the sharing is no longer additive with respect to the group law but is more generally based on a linear code [56]. For instance, if the linear code corresponds to Shamir's polynomial sharing (in this case the code is a Reed-Solomon one), then the a_i (resp. the c_i) correspond to the evaluation in $d+1$ public points α_i of a random degree-d polynomial $P_a(X)$ (resp. $[P_a \circ \mathcal{A}](X)$) with constant term a (resp. c). Namely, the input shares are defined such that $a_i = P_a(\alpha_i)$ and, by construction, the output shares satisfy $c_i = [P_a \circ \mathcal{A}](\alpha_i)$ (see for instance [55]).

2.3 Securing Quadratic Transformations

In [15], Coron *et al* have recently shown that multiplications of the form $a \times \mathcal{L}(a)$, with $\mathcal{L}$ being $\mathbb{F}_2$-linear, can be securely evaluated more efficiently than standard multiplications when n is small enough to allow for the tabulation of univariate transformation in $\mathbb{F}_{2^n}$ (*i.e.* when $n \leqslant 10$ for nowadays devices). This scheme is recalled hereafter where the operation $a \mapsto a \times \mathcal{L}(a)$ is denoted by $\mathcal{Q}(a)$.

Algorithm 5: Higher-Order Masking Scheme for Multiplication in the form $a \times \mathcal{L}(a)$ (Additive Sharing)

Input : the $(d+1)^{\text{th}}$-order sharing $(a_0, a_1, \cdots, a_d)$ of a in $\mathbb{F}_{2^n}$
Output: a $(d+1)^{\text{th}}$-order sharing $(c_0, c_1, \cdots, c_d)$ of $c = a \times \mathcal{L}(a)$

Randomly generate $d(d+1)^2/2$ elements $r_{ij} \in \mathbb{F}_{2^n}$ indexed such that $0 \leqslant i < j \leqslant d$
Randomly generate $d(d+1)^2/2$ elements $r'_{ij} \in \mathbb{F}_{2^n}$ indexed such that $0 \leqslant i < j \leqslant d$
for $i = 0$ **to** d **do**
    for $j = i+1$ **to** d **do**
        $r_{j,i} \leftarrow r_{i,j} + \mathcal{Q}(a_i + r'_{i,j}) + \mathcal{Q}(a_j + r'_{i,j}) + \mathcal{Q}((a_i + r'_{i,j}) + a_j) + \mathcal{Q}(r'_{i,j})$
for $i = 0$ **to** d **do**
    $c_i \leftarrow \mathcal{Q}(a_i)$
    for $j = 0$ **to** d, $j \neq i$ **do**
        $c_i \leftarrow c_i + r_{i,j}$
return $(c_0, c_1, \ldots, c_d)$

Algorithm 5 can actually be extended to any quadratic function (instead of only the quadratic functions $a \in \mathbb{F}_{2^n} \mapsto a \times \mathcal{L}(a)$). Let $\mathcal{Q}$ be any quadratic function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$. The bivariate function $\varphi^{(2)}(a_0, a_1) = \mathcal{Q}(a_0 + a_1) + \mathcal{Q}(a_0) + \mathcal{Q}(a_1) + \mathcal{Q}(0)$ is bilinear (this is a necessary and sufficient condition for h to be quadratic), symmetric and null when a_0, a_1 are linearly dependent over $\mathbb{F}_2$ (that is, when $a_0 = 0$ or $a_1 = 0$ or $a_0 = a_1$). The equality $\mathcal{Q}(a_0 + a_1) = \varphi^{(2)}(a_0, a_1) + \mathcal{Q}(a_0) + \mathcal{Q}(a_1) + \mathcal{Q}(0)$ can be iterated: it can be easily proven by induction on $d \geq 1$ that for every $(a_0, a_1, \ldots, a_d) \in {\mathbb{F}_{2^n}}^{d+1}$, we have;

$$\mathcal{Q}\left(\sum_{i=0}^{d} a_i\right) = \sum_{0 \leq i < j \leq d} \varphi^{(2)}(a_i, a_j) + \sum_{i=0}^{d} \mathcal{Q}(a_i) + (d \ [\text{mod } 2])\, \mathcal{Q}(0) \ . \quad (3)$$

Note that this formula, which has been extended from the quadratic case to any algebraic degree s in [11] (see Section 3.4, is also valid for $d = 0$. Moreover, for every $a_i, a_j, r'_{i,j}$ in $\mathbb{F}_{2^n}$ we have

$$\varphi^{(2)}(a_i, a_j) = \mathcal{Q}(a_i + a_j + r'_{i,j}) + \mathcal{Q}(a_i + r'_{i,j}) + \mathcal{Q}(a_j + r'_{i,j}) + \mathcal{Q}(r'_{i,j}) \ , \quad (4)$$

since $\varphi^{(2)}(a_0, a_1) + \mathcal{Q}(a_0 + a_1 + r) + \mathcal{Q}(a_0 + r) + \mathcal{Q}(a_1 + r) + \mathcal{Q}(r) = \varphi^{(2)}(a_0, a_1) + \varphi^{(2)}(a_0, r) + \varphi^{(2)}(a_0, a_1 + r)$ is null ($\varphi^{(2)}$ being bilinear). Hence, the same calculations as above can be made by injecting $r'_{i,j}$ into each processing of $\varphi^{(2)}(a_0, a_1)$ and we have then:

$$\mathcal{Q}\left(\sum_{i=0}^{d} a_i\right) = \sum_{0\leq i<j\leq d} \mathcal{Q}(a_i + a_j + r'_{i,j}) + \mathcal{Q}(a_i + r'_{i,j}) + \mathcal{Q}(a_j + r'_{i,j}) + \mathcal{Q}(r'_{i,j}) + \sum_{i=0}^{d} \mathcal{Q}(a_i) + (d \ [\mathrm{mod}\ 2])\, \mathcal{Q}(0) \ . \quad (5)$$

From (5) we deduce that a quadratic function $\mathcal{Q}$ can be securely evaluated for any d by processing the following sequence of operations:

Algorithm 6: Higher-Order Masking Scheme for Quadratic Vectorial Function

Input : a $(d+1)^{\text{th}}$-order sharing $(a_0, a_1, \cdots, a_d)$ of a in $\mathbb{F}_2^n$
Output: a $(d+1)^{\text{th}}$-order sharing $(c_0, c_1, \cdots, c_d)$ of $c = \mathcal{Q}(a)$

1 Randomly generate $d(d+1)/2$ elements $r_{ij} \in \mathbb{F}_{2^n}$ indexed such that $0 \leqslant i < j \leqslant d$
2 Randomly generate $d(d+1)/2$ elements $r'_{ij} \in \mathbb{F}_{2^n}$ indexed such that $0 \leqslant i < j \leqslant d$
3 **for** $i = 0$ **to** d **do**
4 **for** $j = i+1$ **to** d **do**
  /* process $r_{j,i} = \varphi_h^{(2)}(a_i, a_j) + r_{i,j}$ */
5 $r_{j,i} \leftarrow r_{i,j} + \mathcal{Q}(a_i + r'_{i,j}) + \mathcal{Q}(a_j + r'_{i,j}) + \mathcal{Q}((a_i + r'_{i,j}) + a_j) + \mathcal{Q}(r'_{i,j})$
6 **for** $i = 0$ **to** d **do**
7 $c_i \leftarrow \mathcal{Q}(a_i)$
8 **for** $j = 0$ **to** d, $j \neq i$ **do**
9 $c_i \leftarrow c_i + r_{i,j}$
10 $c_0 \leftarrow c_0 + (d \bmod 2)\mathcal{Q}(0)$
11 **return** $(c_0, c_1, \ldots, c_d)$

Except the addition of the constant term at Step 7, Algorithm 6 is exactly the same as Algorithm 5. It involves $5d(d+1)$ additions and $2d(d+1)$ calls to the transformation $\mathcal{Q}$. In order to satisfy the d^{th}-order security, the sequence of operations at Step 5 must be done from left to right.

2.4 Conclusion About Elementary Masking Schemes

We sum-up hereafter the complexities of Algorithms[9] 2, 3, 4 and 6:

Scheme	Additions	Multilpications	LUT Calls
Scheme for Multiplications (Algo. 2)	$2d(d+1)$	$(d+1)^2$	0
Scheme for Multiplications (Algo. 3)	$4d^3 + 8d^2 + 3d$	$4d^3 + 4d^2 + 5d + 2$	0
Scheme for Affine Transformations	0	0	$d+1$
Scheme for Quadratic Transformations	$5d(d+1)$	0	$(2d+1)(d+1)$

[9] The improvement of Algorithm 3 proposed in [56] involves $d^3 + 9d^2 + 5d$ additions and $d^3 + 8d^2 + 9d + 2$ multiplications, which leads to an improvement when $d \geqslant 3$ (see [56]).

In [31], Grosso *et al* experimentally validated for $n = 8$ the advantage of using Algorithm 5 instead of Algorithm 2 to securely process multiplications in the form $a \times \mathcal{L}(a)$. For $d \in \{1, 2, 3\}$, they indeed implemented both approaches in `C` and in Assembly on ATMEGA644p. We recall their results hereafter:

Table 2. Costs comparison (in cycles) between Algorithms 2 and 5 over $\mathbb{F}_{2^n}[8]$.

Operation	`C` $d = 1$	`C` $d = 2$	`C` $d = 3$	[Assembly] $d = 1$
Algorithm 2	146	430	802	136
Algorithm 5	61	152	344	54

3 Securing Polynomial Evaluation

3.1 On the Notion of Masking Complexity

The core idea of the secure polynomial evaluations proposed in the literature is to split the processing into a sequence of field multiplications and $\mathbb{F}_2$-linear operations, and then to secure both operations independently thanks to the methods recalled in previous section. Taking into account that the complexity of masking schemes for $\mathbb{F}_2$-linear operations is linear in d, whereas that for multiplications is at least quadratic, the proposed techniques try to minimize the number of field multiplications which are not $\mathbb{F}_2$-linear[10] (this kind of multiplication shall be called *non-linear* in this paper). This strategy led the authors of [10] to introduce the notion of *polynomial masking complexity*, which corresponds to the minimal number of non-linear multiplications needed to evaluate a given polynomial (aka s-box). Computing this masking complexity for any given function is today a challenge. Following a brute force approach, the authors of [10] exhibited the masking complexity for all monomials in $\mathbb{F}_{2^n}$ with $n \leqslant 8$. Since the complexity is the same for all powers in the same cyclotomic class, results are grouped by classes. We recall the following table from [10]:

Determining the masking complexity of a monomial $x^\alpha \in \mathbb{F}_{2^n}[x]$ amounts to find the shortest 2-*addition chain* for α, with the supplementary assumption that multiplications by 2 are for free. The notion of q-addition chain has been introduced in [38] and studied *e.g.* in [66]. The general problem (without the assumption that multiplications by q are for free) is known to be a NP-hard problem. In [59], the authors argue that the notion of *cyclotomic class addition chain* (CC-addition chain for short) is more accurate to refer to the processing of x^α from cyclotomic class elements. A CC-addition chain for a non-zero element $\alpha \in \mathbb{Z}/(n-1)\mathbb{Z}$ is a collection of cyclotomic classes $(C_{a_i})_{0 \leq i \leq r}$ such that $C_{a_0} = 1$ and $C_{a_r} = C_\alpha$, and for every $i \in [1..r]$ there exist $(j, k) \in [0..r]^2$, $\beta_j \in C_{a_j}$ and

[10] A multiplication over a field of characteristic 2 corresponding to a squaring is $\mathbb{F}_2$-linear.

Table 3. Cyclotomic classes for $n \in \{4, 6, 8\}$ w.r.t. the masking complexity k

k	Cyclotomic classes C_i of elements x^i in $\mathbb{F}_{2^n}$ for $n \in \{4, 6, 8\}$
	$n = 4$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8\}$
1	$C_3 = \{3, 6, 12, 9\}$, $C_5 = \{5, 10\}$
2	$C_7 = \{7, 14, 13, 11\}$
	$n = 6$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8, 16, 32\}$
1	$C_3 = \{3, 6, 12, 24, 48, 33\}$, $C_5 = \{5, 10, 20, 40, 17, 34\}$ $C_9 = \{9, 18, 36\}$
2	$C_7 = \{7, 14, 28, 56, 49, 35\}$, $C_{11} = \{11, 22, 44, 25, 50, 37\}$ $C_{13} = \{13, 26, 52, 41, 19, 38\}$, $C_{15} = \{15, 30, 29, 27, 23\}$ $C_{21} = \{21, 42\}$, $C_{27} = \{27, 54, 45\}$
3	$C_{23} = \{23, 46, 29, 58, 53, 43\}$, $C_{31} = \{31, 62, 61, 59, 55, 47\}$
	$n = 8$
0	$C_0 = \{0\}$, $C_1 = \{1, 2, 4, 8, 16, 32, 64, 128\}$
1	$C_3 = \{3, 6, 12, 24, 48, 96, 192, 129\}$, $C_5 = \{5, 10, 20, 40, 80, 160, 65, 130\}$, $C_9 = \{9, 18, 36, 72, 144, 33, 66, 132\}$, $C_{17} = \{17, 34, 68, 136\}$
2	$C_7 = \{7, 14, 28, 56, 112, 224, 193, 131\}$ $C_{11} = \{11, 22, 44, 88, 176, 97, 194, 133\}$, $C_{13} = \{13, 26, 52, 104, 208, 161, 67, 134\}$ $C_{15} = \{15, 30, 60, 120, 240, 225, 195, 135\}$, $C_{19} = \{19, 38, 76, 152, 49, 98, 196, 137\}$ $C_{21} = \{21, 42, 84, 168, 81, 162, 69, 138\}$, $C_{25} = \{25, 50, 100, 200, 145, 35, 70, 140\}$ $C_{27} = \{27, 54, 108, 216, 177, 99, 198, 141\}$, $C_{37} = \{37, 74, 148, 41, 82, 164, 73, 146\}$ $C_{45} = \{45, 90, 180, 105, 210, 165, 75, 150\}$, $C_{51} = \{51, 102, 204, 153\}$, $C_{85} = \{85, 170\}$
3	$C_{23} = \{23, 46, 92, 184, 113, 226, 197, 139\}$ $C_{29} = \{29, 58, 116, 232, 209, 163, 71, 142\}$, $C_{31} = \{31, 62, 124, 248, 241, 227, 199, 143\}$ $C_{39} = \{39, 78, 156, 57, 114, 228, 201, 147\}$, $C_{43} = \{43, 86, 172, 89, 178, 101, 202, 149\}$ $C_{47} = \{47, 94, 188, 121, 242, 229, 203, 151\}$ $C_{53} = \{53, 106, 212, 169, 83, 166, 77, 154\}$ $C_{55} = \{55, 110, 220, 185, 115, 230, 205, 155\}$ $C_{59} = \{59, 118, 236, 217, 179, 103, 206, 157\}$ $C_{61} = \{61, 122, 244, 233, 211, 167, 79, 158\}$ $C_{63} = \{63, 126, 252, 249, 243, 231, 207, 159\}$ $C_{87} = \{87, 174, 93, 186, 117, 234, 213, 171\}$, $C_{91} = \{91, 182, 109, 218, 181, 107, 214, 173\}$ $C_{95} = \{95, 190, 125, 250, 245, 235, 215, 175\}$ $C_{111} = \{111, 222, 189, 123, 246, 237, 219, 183\}$ $C_{119} = \{119, 238, 221, 187\}$
4	$C_{127} = \{127, 254, 253, 251, 247, 239, 223, 191\}$

$\beta_k \in C_{a_k}$ such that $\beta_i \equiv \beta_j + \beta_k \bmod 2^n - 1$. The value r is called the size of the CC-addition chain. The masking complexity of x^α corresponds to the shortest CC-addition chain of α.

In some contexts, it may be pertinent to evaluate a monomial defined over $\mathbb{F}_{2^n}$ thanks to operations defined over subfields $\mathbb{F}_{2^{n/2^r}}$ of $\mathbb{F}_{2^n}$ (*e.g.* applying the Tower Fields approach recalled in Section 2.1). This strategy increases the overall number of multiplications. However, operating in $\mathbb{F}_{2^{n/2^r}}$ instead of $\mathbb{F}_{2^n}$ may have a significant practical impact on the processing cost (in terms of `CPU` cycles number). For instance, according to Table 1, the number of cycles required to process multiplications in $\mathbb{F}_{2^4}$ with $\texttt{LUT}_{n=4}$ is 4 (in `8051`) if 256 bytes of `ROM` are available. Since the multiplication in $\mathbb{F}_{2^8}$ cannot be tabulated (it would require 256^2 bytes of `ROM`), the best timing/memory trade-off is achieved with $\texttt{LAL}_{n=8}$ method and leads to a cost of around 25 cycles in `8051` architecture. Eventually, we get a multiplication over $\mathbb{F}_{2^4}$ which is around 6 times faster than a multiplication over $\mathbb{F}_{2^8}$. Hence exchanging the latter operation by 6 or less multiplications in $\mathbb{F}_{2^4}$ leads to a practical efficiency gain. This strategy has been followed by Kim *et al* in [37] for the evaluation of the monomial x^{254} in $\mathbb{F}_{2^8}$ (which is affinely equivalent to the `AES` sbox) and led to a practical improvement compared to the approach in [19].

3.2 Masking Complexity of Polynomials

When the polynomial representation is not reduced to a single monomial, the notions of CC-addition chain can be straightforwardly extended. Actually, the notion of polynomial chain is given in [38, Section 4.6.4] and the shortest size of such a chain (when only non-linear multiplications are counted) exactly corresponds to the masking complexity.

For $n \leq 8$, Table 3 can of course be used to deduce an upper-bound of the masking complexity of any polynomial defined over $\mathbb{F}_{2^n}$, by summing the masking complexities of its monomials. However, as we will see hereafter, the achieved bounds are far from being tight since the evaluation of a polynomial can be performed more efficiently than simply evaluating each of its monomials separately. Actually, the authors of [10] present two polynomial evaluation methods which aim at minimizing the number of required non-linear multiplications. They have been afterwards improved in [59] and recently in [16]. We recall these works afterwards.

In [10], the authors propose two solutions to securely evaluate a polynomial $P(x) \in \mathbb{F}_{2^n}[x]$.

The *cyclotomic method* consists in rewriting $P(x)$ in the form:

$$P(x) = u_0 + \sum_{i=1}^{q} L_i(x^{\alpha_i}) + u_{2^n-1}x^{2^n-1} \ , \tag{6}$$

where q is a positive integer and $(L_i)_{i \leqslant q}$ is a family of linearized polynomials[11]. Since the transformations $x \in \mathbb{F}_{2^n} \mapsto x^{2^j}$ are $\mathbb{F}_2$-linear, their masking complexity is null. This implies that the masking complexity of $\sum_{i=1}^{q} L_i(x^{\alpha_i})$ equals the number of non-linear multiplications required to evaluate all the monomials x^{α_i}. It is shown in [10], that the latter number is bounded above by the number of cyclotomic classes in $\mathbb{F}_{2^n}$ minus 2, which led to the following proposition:

Proposition 1. [10] *Let n be a positive integer. For every $\mathrm{P}(x) \in \mathbb{F}_{2^n}[x]$, the masking complexity of $P(x)$, denoted $\mathcal{MC}(P)$, satisfies:*

$$\mathcal{MC}(P) \leqslant \sum_{\delta | (2^n - 1)} \frac{\varphi(\delta)}{\mu(\delta)} - 1 \ ,$$

where $\mu(\delta)$ denotes the multiplicative order of 2 modulo δ and $\varphi(\cdot)$ denotes the Euler totient function.

Remark 2. The proposition is a direct implication of the fact that the number of cyclotomic classes in $\mathbb{F}_{2^n}$ is $\sum_{\delta | (2^n-1)} \frac{\varphi(\delta)}{\mu(\delta)}$, which is bounded below by $(2^n - 1)/n$.

Remark 3. It is proved in [59] that the masking complexity is invariant w.r.t. field representation.

Proposition 1 has been afterwards completed in [59] with the following result giving a lower bound on the masking complexity.

Proposition 2. [59] *Let n be a positive integer. For every polynomial $\mathrm{P}(x) = \sum_{i=0}^{2^n-1} u_i x^i$ in $\mathbb{F}_{2^n}[x]$, the masking complexity of $P(x)$ satisfies:*

$$\max_{\substack{0 < i < 2^n - 1 \\ u_i \neq 0}} \ m_n(i) \leqslant \mathcal{MC}(P) \ ,$$

where, for every $i \in [1..2^n - 2]$, $m_n(i)$ denotes the size of the shortest cyclotomic-class addition chain.

Remark 4. It may be observed that the masking complexity of the monomial x^i exactly corresponds to $m_n(i)$ [10, 59]. The authors of [59] recall that the $m_n(i)$ is itself bounded above by $\lceil \log_2(\mathtt{HW}(i)) \rceil$. They moreover show that techniques proposed by Brauer in [8] may be applied to prove that $m_n(i)$ is bounded below by $\frac{\log_2(i)}{\log_2 \log_2(i)} \times (1 + O(1))$ when i (and thus n) tends towards infinity.

Thanks to Proposition 2, Roy and Vivek argue that the masking complexity of the `DES` s-boxes is lower bounded by 3, whereas the s-box of `AES` is bounded below by 4 (actually the bound is tight with the representation introduced in [57]). Proposition 2 has been further improved by Coron, Roy and Vivek in [16] where the following new lower bound has been exhibited by adapting a technique initially introduced by Paterson and Stockmeyer [49].

[11] *i.e.* a linear combination of monomials in the form x^{2^j} with $j < n$

Proposition 3. *For every positive integer n, there exists a polynomial $P(x) \in \mathbb{F}_{2^n}[x]$ with masking complexity satisfying:*

$$\sqrt{\frac{2^n}{n}} - 2 \leqslant \mathcal{MC}(P) \ . \tag{7}$$

For the polynomials $P(x) = \sum_{i=0}^{2^n-1} u_i x^i$ whose masking complexity is bounded above by $\sqrt{\frac{2^n}{n}} - 2$, Proposition 3 improves the lower-bound $\max_{i,u_i \neq} \lceil \log_2(\mathtt{HW}(i)) \rceil$ givenin Proposition 2 when n is greater than or equal to 9.

The *Knuth-Eve method* proposed in [10] is actually a direct application of Knuth-Eve algorithm [25, 39] which is based on a recursive use of the following lemma.

Lemma 1. *Let n and t be two positive integers and let $P(x)$ be a polynomial of degree t over $\mathbb{F}_{2^n}[x]$. There exist two polynomials $P_1(x)$ and $P_2(x)$ of degrees bounded above by $\lfloor t/2 \rfloor$ over $\mathbb{F}_{2^n}[x]$ such that:*

$$P(x) = P_1(x^2) \oplus P_2(x^2)x \ . \tag{8}$$

Applying Lemma 1 to the polynomial $P(x)$ gives $P(x) = P_1(x^2) + P_2(x^2)x$, where $P_1(x)$ and $P_2(x)$ are two polynomials of degrees bounded above by $2^{n-1} - 1$. The authors of [10] deduce that $P(x)$ can be computed after computation of all monomials $(x^{2j})_{j \leq 2^{n-1}-1}$ with a single multiplication by x. Then, applying Lemma 1 again to the polynomials $P_1(x)$ and $P_2(x)$ both of degree bounded above by $2^{n-1} - 1$ leads to two new pairs of polynomials $(P_{11}(x), P_{12}(x))$ and $(P_{21}(x), P_{22}(x))$ such that $P_1(x^2) = P_{11}(x^4) + P_{12}(x^4)x^2$ and $P_2(x^2) = P_{21}(x^4) + P_{22}(x^4)x^2$. The degree of the new polynomials is bounded above by $2^{n-2} - 1$. Eventually, applying Lemma 1 recursively r times gives an evaluation of $P(x)$ involving evaluations in x^{2^r} of polynomials of degree bounded above by $2^{n-r} - 1$ plus $2^r - 1 = \sum_{i=0}^{r-1} 2^i$ multiplications by powers of x in the form x^{2^i} with $i \leq 2^{r-1}$. This observation leads to the following proposition.

Proposition 4. *Let n be a positive integer. For every $P(x) \in \mathbb{F}_{2^n}[x]$, the masking complexity of $P(x)$ satisfies:*

$$\mathcal{MC}(P) \leqslant \min_{0 \leq r \leq n} (2^{n-r-1} + 2^r) - 2 = \begin{cases} \frac{3}{2}2^{n/2} - 2 & \text{if } n \text{ is even} \\ 2^{(n+1)/2} - 2 & \text{if } n \text{ is odd} \end{cases} \ . \tag{9}$$

Roy-Vivek's method has been introduced in [59]. It follows an approach very close to that of Paterson and Stockmeyer in [49] and it essentially consists in expressing $P(x)$ as a function of several lower degree polynomials, each of degree at most k for some fixed k. In its most simple version, the method assumes that the degree of $P(x)$ equals $k(2t-1)$ and it starts by dividing $P(x)$ by x^{kt}. The remainder $R_0(x)$ has degree at most $kt - 1$, whereas the quotient $Q_0(x)$ has degree $k(t-1)$. Adding the term $x^{k(t-1)}$ to $R_0(x)$ and dividing the sum by

$Q_0(x)$ leads to $R_0(x) - x^{k(t-1)} = C_0(x) \times Q_0(x) + S_0(x)$ where $C_0(x)$ and $S_0(x)$ have degree at most k and $k(t-1)-1$ respectively. We then get:

$$P(x) = (x^{kt} + C_0(x)) \times Q_0(x) + x^{k(t-1)} + S_0(x) \ .$$

The method is then applied recursively to the polynomials $Q_0(x)$ and $x^{k(t-1)} + S_0(x)$ (both of degree $k(t-1)$). Namely, they are both divided by $x^{\frac{kt}{2}}$ leading to:

$$Q_0(x) = (x^{\frac{kt}{2}} + C_1(x)) \times Q_1(X) + x^{k(\frac{t}{2}-1)} + S_1(x)$$

and

$$x^{k(t-1)} + S_0(x) = (x^{\frac{kt}{2}} + C_2(x)) \times Q_2(x) + x^{k(\frac{t}{2}-1)} + S_2(x) \ ,$$

where, for $i \in \{1, 2\}$, the polynomials $C_i(x)$ have degree at most k, the polynomials $Q_i(x)$ have degree $k(\frac{t}{2}-1)$ and the polynomials $S_i(x)$ have degree strictly lower than $k(\frac{t}{2}-1)$). Repeating the procedure $\lceil \log_2(t) \rceil$ times eventually splits $P(x)$ as a combination of polynomials of degree upper bounded by k and of monomials in the cyclotomic class of x^k. For a polynomial $P(x)$ representing an s-box from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^m}$, the number of non-linear multiplications needed with Roy-Vivek's method is around $k \times (2^m - 1)$ (assuming that the polynomial representation is dense). It involves around $(k+1) \times (2^m - 1)$ additions and $k/2 + \log_k \deg(P)$ squarings.

Roy-Vivek's method enables to process the `DES` s-boxes with 7 non-linear multiplications which is smaller than the numbers 10 and 11 respectively needed with the cyclotomic and Knuth-Eve's methods. Actually, for `CAMELIA`, `CLEFIA`, `PRESENT` and `SERPENT` s-boxes, Roy-Vivek's method is at least as efficient as the latter methods, and often performs more efficiently.

Coron-Roy-Vivek's method has been recently proposed in [16] and may be viewed as an extension of [59]. It first consists in building an union $\mathcal{C}$ of some cyclotomic classes $\mathcal{C}_i$ of elements in $\mathbb{Z}/(2^n-1)\mathbb{Z}$. The number of non-linear multiplications required to build $\mathcal{C}$ is denoted by μ. The set of monomials x^j with j in $\mathcal{C}$ spans a subspace of $\mathbb{F}_{2^n}[x]$ which is denoted by $\mathcal{P}$. The second step of Coron's method consists in finding a t-variate polynomial $R \in \mathbb{F}_{2^n}[x_1, \cdots, x_t]$ such that:

$$P(x) = R\left(P_1(x), \cdots, P_t(x)\right) \ , \tag{10}$$

and $\mathcal{MC}(R) + \mu$ is as small as possible. To ease the search of the polynomial R, Coron suggests to limit the search to some polynomials and to apply the following heuristic approach:

1. build the union set $\mathcal{C}$ such that all the powers of P's monomials are in $\mathcal{C}+\mathcal{C}$.
2. Choose/fix a set of r polynomials $P_1(x)$, ..., $P_r(x)$ in $\mathcal{P}$ and search $r+1$ polynomials $P_{r+1}(x)$, ..., $P_{2r+1}(x)$ such that:

$$P(x) = \sum_{i=1}^{r} P_i(x) \times P_{r+i}(x) + P_{2r+1}(x) \ . \tag{11}$$

To find the $r+1$ polynomials $P_{r+i}(x)$, with $i \in [1..r+1]$, Coron suggests to solve the linear system of $n2^n$ Boolean equations implied by the evaluation of Equation (11) in every $x \in \mathbb{F}_{2^n}$. Let ℓ denote the size of $\mathcal{C}$. The number of unknown values in the system is bounded above by $\min(r,\ell) \times \ell + \ell$. Hence, the condition $2^n \leqslant \ell \times (1 + \min(r,\ell))$ ensures that the method outputs at least one solution.

In [16], it is pointed out that the method is heuristic and that there is currently no proof that it leads to a solution. In practice however, it is observed that the method always leads to a solution in the cases considered by the author. Its complexity (in terms of the number of non-linear multiplications) in those cases is $O(\sqrt{2^n/n})$, which is asymptotically better than the complexity of Knuth-Eve's method that equals $O(\sqrt{2^n})$ (due to Inequality (9)). Moreover, a comparison of Coron's complexity with Inequality (7) shows that it is asymptotically optimal.

The method is applied for the first `DES` s-box and leads to an evaluation with only 4 non-linear multiplications, implying that the masking complexity of this sbox is at most 4 (and at least 3 due to Proposition 2). The method is also applied to the sboxes of CLEFIA [63], PRESENT [7] leading to a complexity of 10 and 2 respectively (which improves all previous methods).

3.3 The Extended Masking Complexity

As recalled in previous section, the secure processing of monomials in the form x^{1+2^s} (which corresponds to Algorithm 5) is more efficient than that of any other power functions which are not in the cyclotomic class of x. Based on this observation, the authors of [31] followed an approach close to [10] in order to exhibit a new processing of power functions where calculi of the form $x \mapsto x \times x^{2^s}$ are no longer considered as nonlinear multiplications but as a third type of operations. Namely, for every power function $x \mapsto x^\alpha$, [31] presents new operations' sequences which first minimize the number of non-linear multiplications (which are neither $\mathbb{F}_2$-affine nor in the form $x \mapsto x \times x^{2^s}$) (referred as Type II operation), and then minimize the number of processings in the form $x \mapsto x \times x^{2^s}$ (referred as Type III operation). As observed by the authors themselves, this amounts to output, for each exponent α, the shortest cyclotomic class addition chain with the supplementary constraint that multiplications by 2^s, for any integer s, or additions in the form $v + 2^s v$ are for free. The length corresponding to this type of addition chain is referred to as *extended length* in [31]. It is defined as a pair (k_1, k_2) where k_1 refers to the number of Type III operations and k_2 refers to the number of Type II operations. The results obtained in [31] are recalled in Table 4.

Remark 5. Costs given in Table 4 have been obtained by first minimizing the global number of Type-II and Type-III operations, and then by minimizing the number of Type-III multiplications. It can be noticed that other minimization strategies could be applied. For instance, if the goal is to minimize the number of Type-III multiplications, then it can be checked that x^{254} can be evaluated

Table 4. Cyclotomic classes for $n \in \{4, 6, 8\}$ w.r.t. the masking complexity k

(k_1, k_2)	Cyclotomic classes C_α of elements x^α in $\mathbb{F}_{2^n}$ for $n \in \{4, 6, 8\}$
	$n = 4$
$(0, 0)$	$C_0 = \{0\}$,$C_1 = \{1, 2, 4, 8\}$
$(0, 1)$	$C_3 = \{3, 6, 12, 9\}$, $C_5 = \{5, 10\}$
$(1, 1)$	$C_7 = \{7, 14, 13, 11\}$
	$n = 6$
$(0, 0)$	$C_0 = \{0\}$,$C_1 = \{1, 2, 4, 8, 16, 32\}$
$(0, 1)$	$C_3 = \{3, 6, 12, 24, 48, 33\}$, $C_5 = \{5, 10, 20, 40, 17, 34\}$, $C_9 = \{9, 18, 36\}$
$(0, 2)$	$C_{11} = \{11, 22, 44, 25, 50, 37\}$,$C_{15} = \{15, 30, 60, 57, 51, 39\}$ $C_{27} = \{27, 54, 45\}$
$(1, 1)$	$C_7 = \{7, 14, 28, 56, 49, 35\}$ $C_{13} = \{13, 26, 52, 41, 19, 38\}$ $C_{21} = \{21, 42\}$ $C_{31} = \{31, 62, 61, 59, 55, 47, \}$
$(1, 2)$	$C_{23} = \{23, 46, 29, 58, 53, 43\}$
	$n = 8$
$(0, 0)$	$C_0 = \{0\}$,$C_1 = \{1, 2, 4, 8, 16, 32, 64, 128\}$
$(0, 1)$	$C_3 = \{3, 6, 12, 24, 48, 96, 192, 129\}$, $C_5 = \{5, 10, 20, 40, 80, 160, 65, 130\}$ $C_9 = \{9, 18, 36, 72, 144, 33, 66, 132\}$, $C_{17} = \{17, 34, 68, 136\}$
$(0, 2)$	$C_{15} = \{15, 30, 60, 120, 240, 225, 195, 135\}$ $C_{21} = \{21, 42, 84, 168, 81, 162, 69, 138\}$ $C_{25} = \{25, 50, 100, 200, 145, 35, 70, 140\}$ $C_{27} = \{27, 54, 108, 216, 177, 99, 198, 141\}$ $C_{45} = \{45, 90, 180, 105, 210, 165, 75, 150\}$ $C_{51} = \{51, 102, 204, 153\}$, $C_{85} = \{85, 170\}$
$(0, 3)$	$C_{63} = \{63, 126, 252, 249, 243, 231, 207, 159\}$ $C_{95} = \{95, 190, 125, 250, 245, 235, 215, 175\}$ $C_{111} = \{111, 222, 189, 123, 246, 237, 219, 183\}$
$(0, 4)$	$C_{39} = \{39, 78, 156, 57, 114, 228, 201, 147\}$ $C_{55} = \{55, 110, 220, 185, 115, 230, 205, 155\}$ $C_{87} = \{87, 174, 93, 186, 117, 234, 213, 171\}$
$(1, 1)$	$C_7 = \{7, 14, 28, 56, 112, 224, 193, 131\}$ $C_{11} = \{11, 22, 44, 88, 176, 97, 194, 133\}$ $C_{13} = \{13, 26, 52, 104, 208, 161, 67, 134\}$ $C_{19} = \{19, 38, 76, 152, 49, 98, 196, 137\}$ $C_{37} = \{37, 74, 148, 41, 82, 164, 73, 146\}$
$(1, 2)$	$C_{23} = \{23, 46, 92, 184, 113, 226, 197, 139\}$ $C_{29} = \{29, 58, 116, 232, 209, 163, 71, 142\}$ $C_{31} = \{31, 62, 124, 248, 241, 227, 199, 143\}$ $C_{43} = \{43, 86, 172, 89, 178, 101, 202, 149\}$ $C_{47} = \{47, 94, 188, 121, 242, 229, 203, 151\}$ $C_{53} = \{53, 106, 212, 169, 83, 166, 77, 154\}$ $C_{59} = \{59, 118, 236, 217, 179, 103, 206, 157\}$ $C_{61} = \{61, 122, 244, 233, 211, 167, 79, 158\}$ $C_{91} = \{91, 182, 109, 218, 181, 107, 214, 173\}$ $C_{119} = \{119, 238, 221, 187\}$
$(1, 3)$	$C_{127} = \{127, 254, 253, 251, 247, 239, 223, 191\}$

without such operation: first process x^{63}, then $(x+x^{63})^3 = x^{189}+x^{127}+x^{65}+x^3$, end eventually process x^{189}, x^{65} and x^3, and subtract them to $(x+x^{63})^3$ to get $x^{254} = (x^{127})^2$ (which gives a processing without Type-III operations and 9 Type-II operations).

For the exponentiation $x \mapsto x^{254}$ (the non-linear part of the `AES` S-box), the extended addition chain is $(1, 2, 5, 25, 125, 127, 254)$ whose extended length is $1+3$. This sequence requires only 1 operation of Type II (to get x^{127}), 2 linear operations (aka Type I operations) (to get and x^2 and x^{254}) and 3 operations of Type II (to get x^5, x^{25} and x^{125})). It may moreover be observed that the sequence involves the same operation $y \mapsto y^{1+2^2}$ each time, which reduces the memory required to implement the solution.

The interest of using extended addition chains instead of addition chains has been experimentally validated by Grosso *et al* [31] for the particular case of the exponentiation $x \mapsto x^{254}$ over $\mathbb{F}_{2^n}[8]$ and the first `DES` sbox. We recall their implementation results hereafter[12]:

Table 5. Secure `AES` S-box for ATMEGA644p.

Solution	[C] $d=1$	[C] $d=2$	[C] $d=3$
Addition-Chain Method [58]	753	1999	3702
Extended Addition-Chain Method [31]	488	1227	2319

Table 6. Secure `DES` S-box for ATMEGA644p.

Solution	C $d=1$	C $d=2$	C $d=3$
Cyclotomic Method [10]	2001	4646	8182
Cyclotomic Method with Type-III operation [31]	1623	3574	7413

3.4 Some recent progresses made during this paper was reviewed

The present paper was written in November 2014. The publication process makes it published approximately one year later. Meanwhile, important advances have been made, that we wish to briefly present. In [11], it has been introduced a new method for masking s-boxes, which decomposes them by means of functions of low algebraic degree, and masks each such low degree function. The decomposition step starts by deriving a family of generators: $\begin{cases} G_1(x) = F_1(x) \\ G_i(x) = F_i\big(G_{i-1}(x)\big) \end{cases}$ where the F_i are random polynomials of algebraic degree s. Then it randomly

[12] Implementations have been done in `C` and compiled for ATMEGA644p microcontroller thanks to the compiler avr_gcc with optimisation flag -o2.

generates t polynomials $Q_i = \sum_{j=1}^{r} L_j \circ G_j$, where the L_j are linearized polynomials. Eventually, it searches for t polynomials P_i of algebraic degree s and for $r+1$ linearized polynomials L_i such that:

$$P(x) = \sum_{i=1}^{t} P_i\big(Q_i(x)\big) + \sum_{i=1}^{r} L_i\big(G_i(x)\big) + L_0(x) \ .$$

As in the CRV method, the search of polynomials P_i and L_i amounts to solve a system of linear equations over $\mathbb{F}_{2^n}$.
For masking a function F of algebraic degree at most s, the method uses that for such function, the mapping

$$\varphi_F^{(s)}(a_1, a_2, \ldots, a_s) = \sum_{I \subseteq \{1,\ldots,s\}} F\Big(\sum_{i \in I} a_i\Big)$$

is multilinear (this is characteristic of functions of algebraic degree at most s). Then, it is proved that, for every $d \geq s$:

$$F\Big(\sum_{i=1}^{d} a_i\Big) = \sum_{1 \leq i_1 < \cdots < i_s \leq d} \varphi_F^{(s)}(a_{i_1}, \ldots, a_{i_s}) + \sum_{j=0}^{s-1} \eta_{d,s}(j) \sum_{\substack{I \subseteq \{1,\ldots d\} \\ |I|=j}} F\Big(\sum_{i \in I} a_i\Big) \ ,$$

where $\eta_{d,s}(j) = \binom{d-j-1}{s-j-1} \bmod 2$ for every $j \leq s-1$, and this allows proving that:

$$F\Big(\sum_{i=1}^{d} a_i\Big) = \sum_{j=0}^{s} \mu_{d,s}(j) \sum_{\substack{I \subseteq \{1,\ldots,d\} \\ |I|=j}} F\Big(\sum_{i \in I} a_i\Big) \ ,$$

where $\mu_{d,s}(j) = \binom{d-j-1}{s-j} \bmod 2$ for every $j \leq s$.
This reduces the complexity of the d-masking of a degree s function to several s-maskings.

4 Conclusion and Perspectives

In this paper we have recalled the main techniques proposed in the literature to evaluate functions over finite fields while defeating higher-order side channel attacks. All of them start by splitting the evaluation into a sequence of elementary operations which are afterwards independently protected with bespoke schemes that operate on shared values and output a sharing of the result. A section has been dedicated to the presentation and analysis of these schemes. Essentially, they allow for the secure processing of any field multiplication or quadratic function (or more general low degree function). Their construction differs depending on the underlying data sharing (*e.g.* additive, polynomial or, more generally, based on a linear code). When additive sharing is used, the complexity of the secure processing of affine transformations is linear in the security order d and it

is quadratic for the secure processing of multiplications or quadratic functions. If data are represented by polynomial (aka Shamir's) sharing, the complexity of the scheme for affine functions stays linear in d but the complexity of the scheme dedicated to the multiplication becomes cubic. As argued in [55, 33], polynomial sharing (and the dedicated schemes) may however be preferred to additive sharing since it provides better resistance against unbounded side-channel attacks[13]. For practical values of d (*e.g.* $d \leqslant 10$), the timing complexity of the recalled schemes in terms of `CPU` cycles strongly depends on the cost of the underlying field multiplication. The latter one itself depends on the field dimension n and the memory constraints. We recalled different multiplication implementation strategies which offer various timing/memory trade-offs. The choice among them depends on the context constraints.

Since the complexity of the schemes dedicated to the secure processing of multiplications is quadratic, several polynomial evaluation strategies published in the literature essentially try to split the evaluation into a sequence of elementary operations including a minimal number of multiplications. We recalled these strategies which are: the Cyclotomic method [10], the Knuth-Eve method [39] and the Coron-Roy-Vivek's method [16]. This approach has raised the need to introduce a new notion, called masking complexity of a polynomial, which corresponds to the minimum number of non-linear multiplications required to evaluate the polynomial on any field element. Computing this complexity for any polynomial seems to be a difficult problem but we recalled several results published in [10, 16, 59] which enable to have lower and upper bounds. Among the three evaluation methods presented in the three latter papers, the one by Coron *et al* seems to be the most efficient one in general, *i.e.* when d and n are not too small and the polynomial has no particular structure (but the very recent Carlet-Prouff-Rivain-Roche CPRR method further improves the efficiency). It involves only $O(\sqrt{2^n/n})$ non-linear multiplications which can be proved to be asymptotically optimal. Despite its qualities, Coron *et al* 's method is heuristic, as well as the more recent CPRR method, and no formal rules today exist to parametrize the main steps (the construction of the union of cyclotomic classes and the choice of the fixed polynomials). Dealing with this issue seems to be an interesting open avenue for further research. Moreover, several ways could be investigated to improve Coron-Roy-Vivek's approach. For instance, it can be studied whether the cost of the building of the classes of cyclotomic classes (which is the first step of the method) could not be reduced. Another interesting subject of further research could be to study whether Coron-Roy-Vivek's approach cannot be advantageously combined with the idea extended masking complexity in which quadratic functions are viewed as a a class of elementary functions whose complexity is between that of affine transformations and that of non-linear multiplications.

[13] these attacks assume that the adversary is not limited to the observation of d intermediate results during the evaluation but can observe any family of intermediate results.

References

1. M.-L. Akkar and L. Goubin. A Generic Protection against High-order Differential Power Analysis. In T. Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 192–205. Springer, 2003.
2. J. Balasch, S. Faust, B. Gierlichs, and I. Verbauwhede. Theory and practice of a leakage resilient masking scheme. In X. Wang and K. Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 758–775. Springer, 2012.
3. M. Bellare, S. Goldwasser, and D. Micciancio. "pseudo-random" number generation within cryptographic algorithms: the DSS case. In B. Kalisky Jr., editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 277–291. Springer, 1997.
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. G. Bertoni and J.-S. Coron, editors. *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*. Springer, 2013.
6. G. Blakely. Safeguarding cryptographic keys. In *National Comp. Conf.*, volume 48, pages 313–317, New York, June 1979. AFIPS Press.
7. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Paillier and Verbauwhede [48], pages 450–466.
8. A. Brauer. On Addtion Chains. *Bull. Amer. MAth. Soc.*, 45, 1939.
9. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
10. C. Carlet, L. Goubin, E. Prouff, M. Quisquater, and M. Rivain. Higher-order masking schemes for s-boxes. In A. Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.
11. C. Carlet, E. Prouff, M. Rivain, and T. Roche. Algebraic decomposition for probing security. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 742–763. Springer, 2015.
12. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Wiener [69], pages 398–412.
13. H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan. Secure computation from random error correcting codes. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 291–310. Springer, 2007.
14. A. Cook, Stephen. *On the minimum computation time of functions*. PhD thesis, Harvard University, Cambridge, MA, USA, 1966. Available at `http://cr.yp.to/bib/entries.html#1966/cook`.
15. J. Coron, E. Prouff, M. Rivain, and T. Roche. Higher-order side channel security and mask refreshing. In S. Moriai, editor, *Fast Software Encryption - 20th*

International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.
16. J. Coron, A. Roy, and S. Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In L. Batina and M. Robshaw, editors, *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 170–187. Springer, 2014.
17. J.-S. Coron. A New DPA Countermeasure Based on Permutation Tables. In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008*, volume 5229 of *Lecture Notes in Computer Science*, pages 278–292. Springer, 2008.
18. J.-S. Coron. Higher order masking of look-up tables. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.
19. J.-S. Coron, C. Giraud, E. Prouff, S. Renner, M. Rivain, and P. K. Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In W. Schindler and S. A. Huss, editors, *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.
20. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In Paillier and Verbauwhede [48], pages 28–44.
21. 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.
22. N. Courtois and L. Goubin. An Algebraic Masking Method to Protect AES against Power Attacks. In D. Won and S. Kim, editors, *Information Security and Cryptology – ICISC 2005*, volume 3935 of *Lecture Notes in Computer Science*, pages 199–209. Springer, 2006.
23. I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, 2008.
24. A. Duc, S. Dziembowski, and S. Faust. Unifying Leakage Models: from Probing Attacks to Noisy Leakage. In P. Q. Nguyen and E. Oswald, editors, *Eurocrypt*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.
25. J. Eve. The evaluation of polynomials. *Comm. ACM*, 6(1):17–?21, 1964.
26. S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, 2010.
27. M. K. Franklin and M. Yung. Communication complexity of secure computation (extended abstract). In S. R. Kosaraju, M. Fellows, A. Wigderson, and J. A. Ellis, editors, *STOC*, pages 699–710. ACM, 1992.
28. L. Genelle, E. Prouff, and M. Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In Preneel and Takagi [50], pages 240–255.
29. 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.
30. L. Goubin and A. Martinelli. Protecting aes with shamir's secret sharing scheme. In Preneel and Takagi [50], pages 79–94.

31. 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.
32. V. Grosso, F.-X. Standaert, and S. Faust. Masking vs. multiparty computation: How large is the gap for aes? In Bertoni and Coron [5], pages 400–416.
33. V. Grosso, F.-X. Standaert, and S. Faust. Masking vs. multiparty computation: how large is the gap for aes? *J. Cryptographic Engineering*, 4(1):47–57, 2014.
34. S. Gueron, O. Parzanchevsky, and O. Zuk. Masked Inversion in GF(2^n) Using Mixed Field Representations and its Efficient Implementation for AES. In N. Nedjah and L. M. Mourelle, editors, *Embedded Cryptographic Hardware: Methodologies and Architectures*, pages 213–228. Nova Science Publishers, 2004.
35. Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
36. A. Karatsuba and Y. Ofman. Multiplication of many-digital numbers by automatic computers. *Proceedings of the USSR Academy of Sciences*, 145:293–294, 1962. Translation in the academic journal Physics-Doklady, 7 (1963), pp. 595596.
37. H. Kim, S. Hong, and J. Lim. A fast and provably secure higher-order masking of aes s-box. In Preneel and Takagi [50], pages 95–107.
38. D. Knuth. *The Art of Computer Programming*, volume 2. Addison Wesley, third edition, 1988.
39. D. E. Knuth. Evaluation of polynomials by computers. *Comm. ACM*, 5(12), 1962.
40. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In Wiener [69], pages 388–397.
41. S. Mangard, T. Popp, and B. M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
42. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
43. S. Mangard and K. Schramm. Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2006.
44. J. Massey. Minimal Codewords and Secret Sharings. *Sixth Joint Sweedish-Russian Workshop on Information Theory*, pages 246–249, 1993.
45. 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.
46. A. Moradi and O. Mischke. How Far Should Theory Be From Practice? - Evaluation of a Countermeasure. In E. Prouff and P. Schaumont, editors, *CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2012.
47. J. Omura and J. Massey. Computational method and apparatus for finite field arithmetic. Technical report, Omnet Associates, May 1986. Patent Number 4,587,627.

48. P. Paillier and I. Verbauwhede, editors. *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*. Springer, 2007.
49. M. Paterson and L. J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, pages 60–66, 1973.
50. 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.
51. E. Prouff and R. P. McEvoy. First-Order Side-Channel Attacks on the Permutation Tables Countermeasure. In C. Clavier and K. Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2009.
52. 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.
53. E. Prouff, M. Rivain, and T. Roche. On the practical security of a leakage resilient masking scheme. In J. Benaloh, editor, *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2014.
54. E. Prouff and T. Roche. Attack on a higher-order masking of the aes based on homographic functions. In G. Gong and K. C. Gupta, editors, *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 262–281. Springer, 2010.
55. E. Prouff and T. Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In Preneel and Takagi [50], pages 63–78.
56. S. Renner. *Protection des Algorithmes Cryptographiques Embarqués*. PhD thesis, University of Bordeaux, 2014. Available at `http://www.math.u-bordeaux1.fr/~srenner/Thesis_Soline_Renner.pdf`.
57. M. Rivain and E. Prouff. Provably secure higher-order masking of aes. In S. Mangard and F.-X. Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
58. 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.
59. A. Roy and S. Vivek. Analysis and improvement of the generic higher-order masking scheme of fse 2012. In Bertoni and Coron [5], pages 417–434.
60. A. Rudra, P. K. Bubey, C. S. Jutla, V. Kumar, J. Rao, and P. Rohatgi. Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In Ç. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 171–184. Springer, 2001.
61. K. Schramm and C. Paar. Higher Order Masking of the AES. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.
62. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
63. T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-bit blockcipher clefia (extended abstract). In A. Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.
64. B. Sunar and Çetin Kaya Koç. An efficient optimal normal basis type ii multiplier. *IEEE Trans. Computers*, 50(1):83–87, 2001.

65. A. L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963. Available at `http://www.de.ufpe.br/toom/articles/engmat/MULT-E.PDF`.
66. J. von zur Gathen. Efficient and optimal exponentiation in finite fields. *Computational Complexity*, 1:360–394, 1991.
67. J. von zur Gathen, A. Shokrollahi, and J. Shokrollahi. Efficient multiplication using type 2 optimal normal bases. In C. Carlet and B. Sunar, editors, *WAIFI*, volume 4547 of *Lecture Notes in Computer Science*, pages 55–68. Springer, 2007.
68. Y. Wang and X. Zhu. A fast algorithm for the Fourier transform over finite fields and its VLSI implementation. *IEEE Journal on Selected Areas in Communications*, 6(3):572–577, Apr. 1988.
69. M. Wiener, editor. *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.