Generalized Polynomial Decomposition for S-boxes with Application to Side-Channel Countermeasures

Dahmun Goudarzi[1,2], Matthieu Rivain[1], Damien Vergnaud[2], and Srinivas Vivek[3]

[1] CryptoExperts, Paris, France
[2] ENS, CNRS, INRIA and PSL Research University, Paris, France
[3] University of Bristol, Bristol, UK

dahmun.goudarzi@cryptoexperts.com
matthieu.rivain@cryptoexperts.com
damien.vergnaud@ens.fr
sv.venkatesh@bristol.ac.uk

Abstract. Masking is a widespread countermeasure to protect implementations of block-ciphers against side-channel attacks. Several masking schemes have been proposed in the literature that rely on the *efficient* decomposition of the underlying s-box(es). We propose a generalized decomposition method for s-boxes that encompasses several previously proposed methods while providing new trade-offs. It allows to evaluate $n\lambda$-bit to $m\lambda$-bit s-boxes for any integers $n, m, \lambda \geq 1$ by seeing it a sequence of m n-variate polynomials over $\mathbb{F}_{2^\lambda}$ and by trying to minimize the number of multiplications over $\mathbb{F}_{2^\lambda}$.

Keywords. S-box decomposition, Multiplicative complexity, Side-channel countermeasure, Masking, Software implementation, Block-cipher.

1 Introduction

Implementing cryptographic algorithms in constrained embedded devices is a challenging task. In the 1990s, Kocher *et al.* [Koc96,KJJ99] showed that one may often use the physical leakage of the underlying device during the algorithm execution (e.g., the running-time, the power consumption or the electromagnetic radiations) to recover some secret information. Side-channel analysis is the class of cryptanalytic attacks that exploit such physical emanations to hinder the strength of the underlying cryptography.

One of the most common technique to protect implementations against side-channel attacks is to mask internal secret variables. This so-called *masking* technique [GP99,CJRR99] splits every sensitive data manipulated by the algorithm (which depends on the secret key and possibly on other variables known to the attacker) into $d+1$ shares (where $d \geq 1$, the *masking order*, plays the role of a security parameter). The first d shares are usually generated uniformly at random and the last one is computed so that the combination of the $d+1$ shares for some

group law is equal to the initial value. With this technique, the attacker actually needs the whole set of $d+1$ shares to learn any information on the initial value. Since each share's observation comes with noise, the higher is the order d, the more complex is the attack [CJRR99,PR13]. This masking approach permits to achieve provable security in formal security models, notably the *probing security model* [ISW03] and the *noisy leakage model* [PR13,DDF14].

Most symmetric cryptographic algorithms manipulate binary data $x \in \{0,1\}^n$ (for some integer $n \geq 1$) and the natural group law used for masking is the Boolean XOR $\oplus$ over $\mathbb{F}_2$ (or more generally addition in the finite field $\mathbb{F}_{2^n}$ or in the vector space $\mathbb{F}_2^n$). Using this Boolean masking, each sensitive data x is thus split into $d+1$ shares $x_0, \ldots, x_d$ whose addition returns the initial value (i.e. $x = x_0 \oplus x_1 \oplus \cdots \oplus x_d$). One can then compute securely any linear function f of the sensitive value x since a sharing of $y = f(x)$ is readily obtained by computing $y_i = f(x_i)$ for $i \in \{0, \ldots, d\}$ (such that $y = y_0 \oplus y_1 \oplus \cdots \oplus y_d$). It is unfortunately not so easy to compute a sharing of $f(x)$ for non-linear functions f but solutions were proposed for multiplication (e.g. see [ISW03,RP10,BBP+16]). However, if the evaluation cost of linear function is linear in d, the best known algorithms for multiplication have $O(d^2)$ computational complexity.

In practice, iterative block cipher (such as AES) apply several time a round function to an internal state composed itself usually of a linear round key addition, of linear operations to ensure diffusion and of non-linear operations (usually called s-boxes) to ensure confusion. The main issue to provide secure implementation of block ciphers is thus to provide an efficient and secure way to mask the s-box(es). The most widely-used solution is to consider their representation as polynomial functions over finite fields $\mathbb{F}_{2^n}$ (using Lagrange's interpolation theorem) and to find an efficient way to evaluate this polynomial using a minimal number of multiplications. In this paper, we present a generalization of known methods and we obtain new interesting construction for efficiency/memory trade-offs.

1.1 Related work

The first generic method to mask any s-box at any masking order d was proposed in 2012 by Carlet, Goubin, Prouff, Quisquater and Rivain [CGP+12] (following prior work by Rivain and Prouff for the AES block cipher [RP10]). The core idea is to split into simple operations over $\mathbb{F}_{2^n}$ (namely, addition, multiplication by constant, squaring and regular multiplication of two distinct elements), the evaluation of the polynomial representation of the s-box. Among these operations, only the regular multiplication of two distinct elements is non-linear (since squaring over a characteristic 2 finite field is actually linear), and one can use the secure multiplication algorithms mentioned above [ISW03,RP10,BBP+16] to evaluate them. Since these operations have $O(d^2)$ complexity, it is interesting to propose an evaluation scheme of the polynomial with as few as possible regular multiplications. Carlet *et al.* [CGP+12] defined the *masking complexity* (also known as *multiplicative complexity* and *non-linear complexity*) of an s-box as the minimal number of such multiplications necessary to evaluate the corresponding

polynomial and they adapted known methods for polynomial evaluation based on *addition chains* (see [CGP+12] for details).

This technique was later improved by Roy and Vivek in [RV13] using *cyclotomic cosets addition chains*. They notably presented a polynomial evaluation method for the DES s-boxes that requires 7 non-linear multiplications (instead of 10 in [CGP+12]). They also presented lower-bound on the length of such a chain and showed that the multiplicative complexity of the DES s-boxes is lower bounded by 3. In 2014, Coron, Roy and Vivek [CRV14] proposed an heuristic method which may be viewed as an extension of the ideas developed in [CGP+12] and [RV13]. The so-called CRV method considers the s-box as a polynomial over $\mathbb{F}_{2^n}$ and has heuristic multiplicative complexity $O(2^{n/2}/\sqrt{n})$ instead of $O(2^{n/2})$ proven multiplicative complexity for the previous methods. They also proved a matching lower bound of $\Omega(2^{n/2}/\sqrt{n})$ on the multiplicative complexity of any generic method to evaluate n-bit s-boxes. For all the tested s-boxes their method is at least as efficient as the previous proposals and it often requires less non-linear multiplications (e.g. only 4 for the DES s-boxes).

In [GR16], Goudarzi and Rivain introduced a new method to decompose an s-box into a circuit with low multiplicative complexity. One can see their approach as a way to model the s-box as a polynomial over $\mathbb{F}_2^n$ (instead of $\mathbb{F}_{2^n}$) and it consists in applying masking at the Boolean level by bitslicing the s-boxes within a block cipher round. The proposed decomposition then relies on the one proposed by Coron *et al.* [CRV14] and extends it to efficiently deal with several coordinate functions. The schemes from [ISW03,RP10,BBP+16] can then be used to secure bitwise multiplication and the method allows to compute all the s-boxes within a cipher round at the same time.

Finally, in [PV16], Pulkus and Vivek generalized and improved Coron *et al.* technique [CRV14] by working over slightly larger fields than strictly needed (i.e. they considered the s-box as a polynomial over $\mathbb{F}_{2^t}$ instead of $\mathbb{F}_{2^n}$, where $t \geq n$) . Their technique permits notably to evaluate DES s-boxes with only 3 non-linear multiplications over $\mathbb{F}_{2^8}$ (compared to 4 over $\mathbb{F}_{2^6}$ with Coron *et al.* method [CRV14]).

1.2 Our Results

We propose a generalized decomposition method for s-boxes that unifies these previously proposed methods and provides new median case decompositions. More precisely, in our approach any $n\lambda$-bit s-box for some integers $n \geq 1$ and $\lambda \geq 1$ can be seen as a polynomial (or a vector of $m \geq 1$ polynomials) over $\mathbb{F}_{2^\lambda}^n$. We first prove a lower bound of $\Omega(2^{\lambda n/2}\sqrt{m/\lambda})$ for the complexity of any method to evaluate $n\lambda$-bit to $m\lambda$-bit s-boxes. We then describe our generalized decomposition method for which we provide concrete parameters to achieve decomposition for several triplet (n, m, λ) and for exemplary s-boxes of popular block ciphers (namely PRESENT [BKL+07], SC2000 [SYY+02], CLEFIA [SSA+07] and KHAZAD [BR00]).

Depending on the s-box, our generalized method allows one to choose the parameters n, m and λ to obtain the best possible s-box decomposition in terms of

multiplications over $\mathbb{F}_{2^\lambda}$. In particular, for 8×8 s-boxes, the CRV decomposition method [CRV14] ($n = 1$, $m = 1$, $\lambda = 8$) and the bitslice decomposition method [GR16] ($n = 8$, $m = 8$, $\lambda = 1$) are special cases of this generalized decomposition method. The implementation results provided in Section 6 (8×8 s-boxes on a 32-bit ARM architecture) show that our method is comparable with [CRV14] while being more space efficient. It is therefore a good alternative to prior techniques and can be effectively implemented in software on devices with limited resources.

In the full version of this paper, we generalize the method further by exploring the problem of decomposing arbitrary (n, m)-bit s-boxes over an arbitrary field $\mathbb{F}_{2^\lambda}$. Namely we do not require that λ divides the s-box input and output bit-lengths. This allows us to also integrate, in addition to [CRV14,GR16], the method of [PV16] that considers decomposition when $\lambda \geq n$.

2 Preliminaries

2.1 Notations and notions

Let λ be a positive integer. Then $\mathbb{F}_{2^\lambda}$ denotes the finite field with 2^λ elements. Let $\mathcal{F}_{\lambda,n}$ be the set of functions from $\mathbb{F}_{2^\lambda}^n$ to $\mathbb{F}_{2^\lambda}$. Using Lagrange's interpolation theorem, any function $f \in \mathcal{F}_{\lambda,n}$ can be seen as a multivariate polynomial over $\mathbb{F}_{2^\lambda}[x_1, x_2, \ldots, x_n]/(x_1^{2^\lambda} - x_1, x_2^{2^\lambda} - x_2, \ldots, x_n^{2^\lambda} - x_n)$:

$$f(x) = \sum_{u \in [0, 2^\lambda - 1]^n} a_u \, x^u \ , \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$, $x^u = x_1^{u_1} \cdot x_2^{u_2} \cdot \ldots \cdot x_n^{u_n}$, and $a_u \in \mathbb{F}_{2^\lambda}$ for every $u = (u_1, \ldots, u_n) \in [0, 2^\lambda - 1]^n$.

The *multiplicative complexity* of a function in $\mathcal{F}_{\lambda,n}$ (also called the non-linear complexity) is defined as the minimal number of $\mathbb{F}_{2^\lambda}$-multiplications required to evaluate it.

2.2 S-box characterization

In the following, an s-box S is characterized with respect to 3 parameters: the number of input elements n; the number of output elements m; and the bit-size of the elements λ. In other words, an s-box with λn input bits and λm outputs bits is represented as follows:

$$S(x) = (f_1(x), f_2(x), \ldots, f_m(x)), \tag{2}$$

where functions $f_1, f_2, \ldots, f_m \in \mathcal{F}_{\lambda,n}$ are called the *coordinate functions* of S.

As mentioned in the introduction, Roy and Vivek presented in [RV13] lower-bound on the length of cyclotomic coset addition chains and used it to derive a logarithmic lower bound on the multiplicative complexity of an s-box (i.e. on the minimal number of such multiplications necessary to evaluate the corresponding

polynomial). Coron *et al.* [CRV14] improved this lower bound and showed that the non-linear complexity of any generic method to evaluate n-bit s-boxes when seen as a polynomial defined over $\mathbb{F}_{2^n}$ is in $\Omega(2^{n/2}/\sqrt{n})$.

In the following section, we generalize their approach and provide a new lower bound on the multiplicative complexity of a sequence of n-variate polynomials over $\mathbb{F}_{2^\lambda}$. Following [RV13], we define the multiplicative complexity notion for such a sequence as follows:

Definition 1 (Polynomial chain). *Let $\lambda \geq 1$, $n \geq 1$ and $m \geq 1$ be three integers and let $f_1, \ldots, f_m \in \mathbb{F}_{2^\lambda}[x_1, \ldots, x_n]$ be a sequence of n-variate polynomials over $\mathbb{F}_{2^\lambda}$. A* polynomial chain $\boldsymbol{\pi}$ *for $(f_1, \ldots, f_m)$ is a sequence $\boldsymbol{\pi} = (\pi_i)_{i \in \{-n, \ldots, \ell\}}$ and a list $(i_1, \ldots, i_m) \in \{-n, \ldots, \ell\}^m$ with*

$$\pi_{-n} = x_n,\ \pi_{1-n} = x_{n-1},\ \ \ldots,\ \pi_{-1} = x_1,\ \pi_0 = 1,$$

$$\pi_{i_j} = f_j(x_1, \ldots, x_n) \bmod (x_1^{2^\lambda} + x_1, \ldots, x_n^{2^\lambda} + x_n),\ \forall j \in \{1, \ldots, m\},$$

and such that for every $i \in \{1, \ldots, \ell\}$, one of the following condition holds:

1. *there exist j and k in $\{-n, \ldots, i-1\}$ such that $\pi_i = \pi_j \cdot \pi_k$;*
2. *there exist j and k in $\{-n, \ldots, i-1\}$ such that $\pi_i = \pi_j + \pi_k$;*
3. *there exists j in $\{-n, \ldots, i-1\}$ such that $\pi_i = \pi_j^2$;*
4. *there exists j in $\{-n, \ldots, i-1\}$ and $\alpha \in \mathbb{F}_{2^\lambda}$ such that $\pi_i = \alpha \cdot \pi_j$.*

Given such a polynomial chain $\boldsymbol{\pi}$ for $(f_1, \ldots, f_m)$, the multiplicative complexity *of $\boldsymbol{\pi}$ is the number of times the first condition holds in the whole chain $\boldsymbol{\pi}$. The* multiplicative complexity of $(f_1, \ldots, f_m)$ *over $\mathbb{F}_{2^\lambda}$, denoted $\mathcal{M}(f_1, \ldots, f_m)$ is the minimal multiplicative complexity over all polynomial chains for $(f_1, \ldots, f_m)$.*

Remark 1. The multiplicative complexity is similar to the classical circuit complexity notion in which we do not count the linear operations over $\mathbb{F}_{2^\lambda}$ (namely addition, scalar multiplication and squaring operations). For any sequence of n-variate polynomials $f_1, \ldots, f_m \in \mathbb{F}_{2^\lambda}[x_1, \ldots, x_n]$ we obviously have:

$$\mathcal{M}(f_1, \ldots, f_m) \leq \mathcal{M}(f_1) + \cdots + \mathcal{M}(f_m)$$

3 Multiplicative Complexity Lower Bound

In the next section, we will provide a heuristic method which given a sequence of n-variate polynomials over $\mathbb{F}_{2^\lambda}$ provides an evaluation scheme (or a circuit) with "small" multiplicative complexity. Following, Coron *et al.* [CRV14], Proposition 1 provides a $\Omega(2^{n\lambda/2}\sqrt{m/\lambda})$ lower bound on this multiplicative complexity. As in [CRV14], the proof is a simple combinatorial argument inspired by [PS73].

Proposition 1. *Let $\lambda \geq 1$, $n \geq 1$ and $m \geq 1$ be three integers. There exists $f_1, \ldots, f_m \in \mathbb{F}_{2^\lambda}[x_1, \ldots, x_n]$ a sequence of n-variate polynomials over $\mathbb{F}_{2^\lambda}$ such that $\mathcal{M}(f_1, \ldots, f_m) \geq \sqrt{\frac{m2^{n\lambda}}{\lambda}} - (2n + m - 1)$.*

Proof. We consider a sequence of n-variate polynomials $f_1, \ldots, f_m$ in the algebra $\mathbb{F}_{2^\lambda}[x_1, \ldots, x_n]$ with multiplicative complexity $\mathcal{M}(f_1, \ldots, f_m) = r$ for some integer $r \geq 1$. If we consider only the non-linear operations in a polynomial chain of minimal multiplicative complexity $\boldsymbol{\pi} = (\pi_i)_{i \in \{-n, \ldots, \ell\}}$, we can see that there exists indices $m_0, m_1, \ldots, m_{n+r+(m-1)}$ with $m_i \in \{-n, \ldots, \ell\}$ for $i \in \{0, \ldots, n + r + (m-1)\}$ such that

- $m_j = -j - 1$ for $j \in \{0, \ldots, n-1\}$
  (i.e. $\pi_{m_j} = \pi_{-j-1} = x_{j+1}$ for $j \in \{0, \ldots, n-1\}$);
- for $k \in \{n, \ldots, n + r - 1\}$, there exist field elements $\beta_k, \beta'_k \in \mathbb{F}_{2^\lambda}$ and $\beta_{k,i,j}, \beta'_{k,i,j} \in \mathbb{F}_{2^\lambda}$ for $(i, j) \in \{0, \ldots, k-1\} \times \{0, \ldots, \lambda - 1\}$ such that

$$\pi_{m_k} = \left(\beta_k + \sum_{i=0}^{k-1} \sum_{j=0}^{\lambda-1} \beta_{k,i,j} \pi_{m_i}^{2^j}\right) \cdot \left(\beta'_k + \sum_{i=0}^{k-1} \sum_{j=0}^{\lambda-1} \beta'_{k,i,j} \pi_{m_i}^{2^j}\right) \mod (x_1^{2^\lambda} + x_1, \ldots, x_n^{2^\lambda} + x_n);$$

- for $k \in \{n + r, \ldots, n + r + (m-1)\}$ there exist field elements $\beta_k \in \mathbb{F}_{2^\lambda}$ and $\beta_{k,i,j} \in \mathbb{F}_{2^\lambda}$ for $(i, j) \in \{0, \ldots, n + r - 1\} \times \{0, \ldots, \lambda - 1\}$ such that

$$f_{k+1-(n+r)} = \pi_{m_k} = \beta_k + \sum_{i=0}^{n+r-1} \sum_{j=0}^{\lambda-1} \beta_{k,i,j} \pi_{m_i}^{2^j} \mod (x_1^{2^\lambda} + x_1, \ldots, x_n^{2^\lambda} + x_n).$$

The total number of parameters β in this evaluation scheme of P is simply equal to:

$$\sum_{k=n}^{n+r-1} 2 \cdot (1 + k \cdot \lambda) + m(1 + (n + r) \cdot \lambda) = r^2\lambda + r(\lambda m + 2\lambda n - \lambda + 2) + \lambda mn + m$$

and each parameter can take any value in $\mathbb{F}_{2^\lambda}$. The number of sequence of n-variate polynomials $f_1, \ldots, f_m \in \mathbb{F}_{2^\lambda}[x_1, \ldots, x_n]$ with multiplicative complexity $\mathcal{M}(f_1, \ldots, f_m) = r$ is thus upper-bounded by $2^{\lambda(r^2\lambda + r(\lambda m + 2\lambda n - \lambda + 2) + \lambda mn + m)}$.

Since the total number of sequence of n-variate polynomials $f_1, \ldots, f_m \in \mathbb{F}_{2^\lambda}[x_1, \ldots, x_n]$ defined $\mod (x_1^{2^\lambda} + x_1, \ldots, x_n^{2^\lambda} + x_n)$ is $((2^\lambda)^{2^{n\lambda}})^m$, in order to be able to evaluate all such polynomials with at most r non-linear multiplications, a necessary condition is to have

$$r^2\lambda + r(\lambda m + 2\lambda n - \lambda + 2) + \lambda mn + m \geq m2^{n\lambda}$$

and therefore

$$r^2\lambda + r(\lambda m + 2\lambda n - \lambda + 2) - (m2^{n\lambda} - \lambda mn - m) \geq 0.$$

Eventually, we obtain

$$r \geq \frac{\sqrt{\lambda 4m2^{n\lambda} + (\lambda m + 2\lambda n - \lambda + 2)^2} - (\lambda m + 2\lambda n - \lambda + 2)}{2\lambda} \tag{3}$$

and

$$r \geq \frac{\sqrt{4\lambda m 2^{n\lambda}} - 2(\lambda m + 2\lambda n - \lambda + 2)}{2\lambda} \geq \sqrt{\frac{m2^{n\lambda}}{\lambda}} - (2n + m - 1).$$

□

4 Generalized Decomposition Method

In this section, we propose a generalized decomposition method for s-boxes that aims at encapsulating previously proposed methods and at providing new median case decompositions. Depending on the s-box, we can then choose the parameters n, m and λ in order to obtain the best possible s-box decomposition in terms of multiplications over $\mathbb{F}_{2^\lambda}$. In particular, for 8×8 s-boxes, the CRV decomposition method [CRV14] ($n = 1$, $m = 1$, $\lambda = 8$) and the bitslice decomposition method [GR16] ($n = 8$, $m = 8$, $\lambda = 1$) are special cases of this generalized decomposition method.

4.1 Decomposition of a Single Coordinate Function

Let us define the *linear power class* of a function $\phi \in \mathcal{F}_{\lambda,n}$, denoted by $\mathcal{C}_\phi$, as the set

$$\mathcal{C}_\phi = \{\phi^{2^i} : i = 0, \ldots, \lambda - 1\}. \tag{4}$$

Intuitively, $\mathcal{C}_\phi$ corresponds to the set of functions in $\mathcal{F}_{\lambda,n}$ that can be computed from ϕ using only the squaring operation. It is not hard to see that $\{\mathcal{C}_\phi\}_\phi$ are equivalence classes partitioning $\mathcal{F}_{\lambda,n}$. For any set $\mathcal{B} \subseteq \mathcal{F}_{\lambda,n}$, let us define the *linear power closure* of $\mathcal{B}$ as the set

$$\overline{\mathcal{B}} = \bigcup_{\phi \in \mathcal{B}} \mathcal{C}_\phi$$

and the *linear span* of $\mathcal{B}$ as the set

$$\langle \mathcal{B} \rangle = \Big\{ \sum_{\phi \in \mathcal{B}} a_\phi \phi \;\Big|\; a_\phi \in \mathbb{F}_{2^\lambda} \Big\}.$$

Let f be a function in $\mathcal{F}_{\lambda,n}$. The proposed decomposition makes use of a basis of functions $\mathcal{B} \subseteq \mathcal{F}_{\lambda,n}$ and consists in writing f as:

$$f(x) = \sum_{i=0}^{t-1} g_i(x) \cdot h_i(x) + h_t(x), \tag{5}$$

where $g_i, h_i \in \langle \overline{\mathcal{B}} \rangle$ and $t \in \mathbb{N}$. By definition, the functions g_i and h_i can be written as

$$g_i(x) = \sum_{j=1}^{|\mathcal{B}|} \ell_j(\varphi_j(x)) \quad \text{and} \quad h_i(x) = \sum_{j=1}^{|\mathcal{B}|} \ell'_j(\varphi_j(x)),$$

where the ℓ_j, ℓ'_j are *linearized polynomials* over $\mathcal{F}_{\lambda,n}$ (i.e. polynomials for which the exponents of all the constituent monomials are powers of 2) and where $\{\varphi_j\}_{1\leq j\leq|\overline{\mathcal{B}}|} = \overline{\mathcal{B}}$. We now explain how to find such a decomposition by solving a linear system.

Solving a linear system. In the following, we shall consider a basis $\mathcal{B}$ such that $1 \in \mathcal{B}$ and we will denote $\mathcal{B}^* = \mathcal{B} \setminus \{1\} = \{\phi_1, \phi_2, \ldots, \phi_{|\mathcal{B}|-1}\}$. We will further heuristically assume $|\mathcal{C}_{\phi_i}| = \lambda$ for every $i \in \{1, 2, \ldots, |\mathcal{B}| - 1\}$. We then get $|\overline{\mathcal{B}}| = 1 + \lambda|\mathcal{B}^*| = 1 + \lambda(|\mathcal{B}| - 1)$.

We first sample t random functions g_i from $\langle\overline{\mathcal{B}}\rangle$. This is simply done by picking $t \cdot |\overline{\mathcal{B}}|$ random coefficients $a_{i,0}, a_{i,j,k}$ of $\mathbb{F}_{2^\lambda}$ and setting $g_i = a_{i,0} + \sum_{j,k} a_{i,j,k}\phi_j^{2^k}$ for every $i \in [0, t-1]$ where $1 \leq k \leq \lambda$ and $1 \leq j \leq |\mathcal{B}| - 1$. Then we search for a family of $t+1$ functions $\{h_i\}_i$ satisfying (5). This is done by solving the following system of linear equations over $\mathbb{F}_{2^\lambda}$:

$$A \cdot c = b \tag{6}$$

where $b = (f(e_1), f(e_2), \ldots, f(e_{2^{n\lambda}}))^\mathsf{T}$ with $\{e_i\} = \mathbb{F}_{2^\lambda}^n$ and where A is a block matrix defined as

$$A = (\mathbf{1}|A_0|A_1|\ \cdots\ |A_t), \tag{7}$$

where $\mathbf{1}$ is the all-one column vector and where

$$A_i = (A_{i,0}|A_{i,1}|\ \cdots\ |A_{i,|\mathcal{B}|-1}) \tag{8}$$

with

$$A_{i,0} = (g_i(e_1), g_i(e_2), \ldots, g_i(e_{2^{n\lambda}}))^\mathsf{T} \tag{9}$$

for every $i \in [0, t]$, with

$$A_{i,j} = \begin{pmatrix} \phi_j(e_1)\cdot g_i(e_1) & \phi_j^2(e_1)\cdot g_i(e_1) & \ldots & \phi_j^{2^{\lambda-1}}(e_1)\cdot g_i(e_1) \\ \phi_j(e_2)\cdot g_i(e_2) & \phi_j^2(e_2)\cdot g_i(e_2) & \ldots & \phi_j^{2^{\lambda-1}}(e_2)\cdot g_i(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_j(e_{2^{n\lambda}})\cdot g_i(e_{2^{n\lambda}}) & \phi_j^2(e_{2^{n\lambda}})\cdot g_i(e_{2^{n\lambda}}) & \ldots & \phi_j^{2^{\lambda-1}}(e_{2^{n\lambda}})\cdot g_i(e_{2^{n\lambda}}) \end{pmatrix}, \tag{10}$$

for every $i \in [0, t-1]$ and $j \in [1, |\mathcal{B}| - 1]$, and with

$$A_{t,j} = \begin{pmatrix} \phi_j(e_1) & \phi_j^2(e_1) & \ldots & \phi_j^{2^{\lambda-1}}(e_1) \\ \phi_j(e_2) & \phi_j^2(e_2) & \ldots & \phi_j^{2^{\lambda-1}}(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_j(e_{2^{n\lambda}}) & \phi_j^2(e_{2^{n\lambda}}) & \ldots & \phi_j^{2^{\lambda-1}}(e_{2^{n\lambda}}) \end{pmatrix}, \tag{11}$$

for every $j \in [1, |\mathcal{B}| - 1]$.

It can be checked that the vector c, solution of the system, gives the coefficients of the h_i's over the basis $\overline{\mathcal{B}}$ (plus the constant term in first position). A necessary condition for this system to have a solution whatever the target vector b (*i.e.* whatever the coordinate function f) is to get a matrix A of full rank. In particular, the following inequality must hold:

$$(t+1)|\overline{\mathcal{B}}| + 1 \geq 2^{n\lambda} \ . \tag{12}$$

Another necessary condition to get a full-rank matrix is that the squared linear power closure $\overline{\mathcal{B}} \times \overline{\mathcal{B}}$ spans the entire space $\mathcal{F}_{\lambda,n}$. More details about the choice of such basis are discussed in the following.

4.2 S-box Decomposition

Let $\mathcal{S}\colon x \mapsto (f_1(x), f_2(x), \ldots, f_m(x))$ be an s-box. We could apply the above decomposition method to each of the m coordinate functions f_i, which could roughly result in multiplying by m the multiplicative complexity of a single function in $\mathcal{F}_{\lambda,n}$. As suggested in [BMP13,GR16], we can actually do better: the product involved in the decomposition of a coordinate function can be added to the basis for the subsequent decompositions. Specifically, we start with some basis $\mathcal{B}_1$ and, for every $i \geq 1$, we look for a decomposition

$$f_i(x) = \sum_{j=0}^{t_i-1} g_{i,j}(x) \cdot h_{i,j}(x) + h_{i,t_i}(x), \tag{13}$$

where $t_i \in \mathbb{N}$ and $g_{i,j}, h_{i,j} \in \langle \overline{\mathcal{B}}_i \rangle$. Once such a decomposition has been found, we carry on with the new basis $\mathcal{B}_{i+1}$ defined as:

$$\mathcal{B}_{i+1} = \mathcal{B}_i \cup \{g_{i,j} \cdot h_{i,j}\}_{j=0}^{t_i-1}. \tag{14}$$

This update process implies that, for each decomposition, the basis grows and hence the number t_i of multiplicative terms in the decomposition of f_i might decrease. In this context, the necessary condition on the matrix rank (see (12)) is different for every i. In particular, the number t_i of multiplications at step i satisfies:

$$t_i \geq \frac{2^{n\lambda} - 1}{\lambda |\mathcal{B}_i^*| + 1} - 1 \ , \tag{15}$$

where as above $\mathcal{B}_i^*$ stands for $\mathcal{B}_i \setminus \{1\}$.

4.3 Basis Selection

Let us recall that the basis $\mathcal{B}_1$ needs to be such that the squared basis $\overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1$ spans the entire space $\mathcal{F}_{\lambda,n}$, *i.e.* $\langle \overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1 \rangle = \mathcal{F}_{\lambda,n}$ in order to have a solvable linear system. This is called the *spanning property* in the following. This property can be rewritten in terms of linear algebra. For every $\mathcal{S} \subseteq \mathcal{F}_{\lambda,n}$, let us define $\mathrm{Mat}(\mathcal{S})$

as the $(\lambda n \times |\mathcal{S}|)$-matrix for which each column corresponds to the evaluation of one function of $\mathcal{S}$ in every point of $\mathbb{F}_{2^\lambda}^n$, that is

$$\mathrm{Mat}(\mathcal{S}) = \begin{pmatrix} \varphi_1(e_1) & \varphi_2(e_1) & \dots & \varphi_{|\mathcal{S}|}(e_1) \\ \varphi_1(e_2) & \varphi_2(e_2) & \dots & \varphi_{|\mathcal{S}|}(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(e_{2^{n\lambda}}) & \varphi_2(e_{2^{n\lambda}}) & \dots & \varphi_{|\mathcal{S}|}(e_{2^{n\lambda}}) \end{pmatrix}, \tag{16}$$

where $\{\varphi_1, \varphi_2, \dots, \varphi_{|\mathcal{S}|}\} = \mathcal{S}$. Then, we have

$$\langle \overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1 \rangle = \mathcal{F}_{\lambda,n} \iff \mathrm{rank}(\mathrm{Mat}(\overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1)) = 2^{\lambda n}. \tag{17}$$

To construct the basis $\mathcal{B}_1$, we proceed as follows. We start with the basis composed of all monomials of degree 1 plus unity, *i.e.* the following basis:

$$\mathcal{B}_1 = \{1, x_1, x_2, \dots, x_n\}. \tag{18}$$

Then, we iterate $\mathcal{B}_1 \leftarrow \mathcal{B}_1 \cup \{\phi \cdot \psi\}$, where ϕ and ψ are randomly sampled from $\langle \overline{\mathcal{B}}_1 \rangle$ until reaching a basis with the desired cardinality and satisfying $\mathrm{rank}(\mathrm{Mat}(\overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1)) \geq 2^{\lambda n}$. We add the constraint that, at each iteration, a certain amount of possible products are tried and only the best product is added to the basis, namely the one inducing the greatest increase in the rank of $\mathrm{Mat}(\overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1)$. To summarize, the construction of the basis $\mathcal{B}_1$ is given in the following algorithm:

Algorithm 1 $\mathcal{B}_1$ construction algorithm

Input: Parameters λ, n, and N
Output: A basis $\mathcal{B}_1$ such that $\langle \overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1 \rangle = \mathcal{F}_{\lambda,n}$

$\mathcal{B}_1 = \{1, x_1, x_2 \dots, x_n\}$
rank $= 0$
while rank$< 2^{n\lambda}$ **do**
  for $i = 1$ to N **do**
    $\phi, \psi \xleftarrow{\$} \langle \overline{\mathcal{B}}_1 \rangle$
    $\mathcal{S}_i \leftarrow \mathcal{B}_1 \cup \{\phi \cdot \psi\}$
    $r_i \leftarrow \mathrm{rank}(\mathrm{Mat}(\mathcal{S}_i \times \mathcal{S}_i))$
  end for
  $j \leftarrow \mathrm{argmax}\ r_i$
  if $r_j =$ rank **then**
    return error
  end if
  rank $\leftarrow r_j$
  $\mathcal{B}_1 \leftarrow \mathcal{S}_j$
end while
return $\mathcal{B}_1$

Remark 2. In [GR16], the starting basis $\mathcal{B}_1$ is constructed from a basis $\mathcal{B}_0$ that has the spanning property by design. In practice, the optimal parameters for s-boxes are always obtained by taking $\mathcal{B}_1 = \mathcal{B}_0$ and could be improved with a smaller basis. Our experiments showed that we can achieve a smaller basis $\mathcal{B}_1$ with the spanning property (hence slightly improving the optimal parameters from [GR16]) with a random generation as described above. For instance, in the Boolean case applied on a 8-bit s-box, Algorithm 1 easily finds a basis $\mathcal{B}_1$ of 26 elements (involving 17 multiplications) instead of 31 by taking $\mathcal{B}_1 = \mathcal{B}_0$ as in [GR16].

4.4 Optimal Parameters

Assuming that satisfying the lower bound on t_i (see (15)) is sufficient to get a full-rank system, we can deduce optimal parameters for our generalized decomposition method. Specifically, if we denote $s_i = |\mathcal{B}_i^*|$, we get a sequence $(s_i)_i$ that satisfies

$$\begin{cases} s_1 = r + n \\ s_{i+1} = s_i + t_i \quad \text{with} \quad t_i = \left\lceil \frac{2^{n\lambda}-1}{\lambda s_i + 1} \right\rceil - 1 \end{cases} \tag{19}$$

for $i = 1$ to $m-1$, where r denotes the number of multiplications involved in the construction of the first basis $\mathcal{B}_1$ (the n *free* elements of $\mathcal{B}_1$ being the monomials $x_1, x_2, \ldots, x_n$). From this sequence, we can determine the optimal multiplicative complexity of the method C^* which then satisfies

$$C^* = \min_{r \geq r_0}(r + t_1 + t_2 + \cdots + t_m) \ , \tag{20}$$

where r_0 denotes the minimal value of r for which we can get an initial basis $\mathcal{B}_1$ satisfying the spanning property (that is $\langle \overline{\mathcal{B}}_1 \times \overline{\mathcal{B}}_1 \rangle = \mathcal{F}_{\lambda,n}$) and where the t_i's are viewed as functions of r according to the sequence (19).

Table 1 provides a set of optimal parameters r, t_1, t_2, $\ldots$, t_m and corresponding C^* for several s-box sizes and several parameters λ and $n = m$ (as for bijective s-boxes). For the sake of completeness, we included the extreme cases $n = 1$, *i.e.* standard CRV method [CRV14], and $\lambda = 1$, *i.e.* Boolean case [GR16]. We obtain the same results as in [CRV14] for the standard CRV method. For the Boolean case, our results slightly differ from [GR16]. This is due to our improved generation of $\mathcal{B}_1$ (see Remark 2) and to our bound on the t_i's (see (15)) which is slightly more accurate than in [GR16].

Table 2 gives the size of the smallest randomised basis we could *achieve* using Algorithm 1 for various parameters. The number of tries made was $N = 1000$ before adding a product of random linear combination to the current basis.

5 Experimental Results

In this section, we report the concrete parameters for random and specific s-boxes achieved using our generalized decomposition method. Table 3 compares

Table 1. Theoretical optimal parameters for our decomposition method.

(λ, n)	$\lvert\mathcal{B}_1\rvert$	r	$t_1, t_2, \ldots, t_n$	C^*
4-bit s-boxes				
(1,4)	7	2	2,1,1,1	7
	8	3	1,1,1,1	7
	9	4	1,1,1,1	8
(2,2)	4	1	2,1	4
	5	2	1,1	4
	6	3	1,1	5
(4,1)	3	1	1	2
	4	2	1	3
6-bit s-boxes				
(1,6)	14	7	4,3,2,2,2,2	22
	15	8	4,3,2,2,2,2	23
(2,3)	8	4	4,2,2	12
	9	5	3,2,2	12
	10	6	3,2,2	13
(3,2)	6	3	3,2	8
	7	4	3,2	9
	8	5	2,2	9
(6,1)	4	2	3	5
	5	3	2	5
	6	4	2	6
8-bit s-boxes				
(1,8)	24	17	9,7,6,5,4,4,4,3	59
	25	18	9,7,5,5,4,4,4,3	59
	28	19	9,6,5,5,4,4,4,3	59
	29	20	8,6,5,5,4,4,4,3	59
	30	21	8,6,5,5,4,4,4,3	60
(2,4)	15	9	9,5,4,4	31
	16	10	8,5,4,4	31
	17	11	8,5,4,3	31
	18	12	7,5,4,3	31
	19	13	7,5,4,3	32
(4,2)	8	5	8,4	17
	9	6	7,4	17
	10	7	6,4	17
	11	8	6,3	17
	12	9	5,3	17
	13	10	5,3	18
(8,1)	5	3	7	10
	6	4	6	10
	7	5	5	10
	8	6	4	10
	9	7	3	10
	10	8	3	11

(λ, n)	$\lvert\mathcal{B}_1\rvert$	r	$t_1, t_2, \ldots, t_n$	C^*
9-bit s-boxes				
(1,9)	35	25	14,10,8,7,6,6,5,5,5	91
	36	26	14,10,8,7,6,6,5,5,5	92
	37	27	13,10,8,7,6,6,5,5,5	92
(3,3)	13	9	13,6,5	33
	14	10	12,6,5	33
	15	11	11,6,5	33
	16	12	11,6,5	34
(9,1)	8	6	7	13
	9	7	6	13
	10	8	6	14
10-bit s-boxes				
(1,10)	49	38	20,14,12,10,9,8,8,7,7,7	140
	50	39	20,14,12,10,9,8,8,7,7,7	141
	51	40	20,14,12,10,9,8,8,7,7,7	142
(2,5)	25	19	20,11,9,7,7	73
	26	20	20,11,9,7,7	74
	27	21	19,11,9,7,7	74
(5,2)	13	10	16,7	33
	14	11	15,7	33
	15	12	14,7	33
	16	13	13,7	33
	17	14	12,7	33
	18	15	11,7	33
	19	16	11,7	34
(10,1)	9	7	12	19
	10	8	11	19
	11	9	10	19
	12	10	9	19
	13	11	8	19
	14	12	7	19
	15	13	7	20

Table 2. Achievable smallest randomised basis computed according to Algorithm 1.

	4-bit s-boxes			5-bit s-boxes		6-bit s-boxes				7-bit s-boxes	
(λ, n)	(1,4)	(2,2)	(4,1)	(1,5)	(5,1)	(1,6)	(2,3)	(3,2)	(6,1)	(1,7)	(7,1)
$\lvert\mathcal{B}_1\rvert$	7	4	3	10	4	14	8	6	4	19	4
r	2	1	1	4	2	7	4	3	2	11	2

	8-bit s-boxes				9-bit s-boxes			10-bit s-boxes			
(λ, n)	(1,8)	(2,4)	(4,2)	(8,1)	(1,9)	(3,3)	(9,1)	(1,10)	(2,5)	(5,2)	(10,1)
$\lvert\mathcal{B}_1\rvert$	26	14	8	5	35	13	5	49	25	11	6
r	17	9	5	3	25	9	3	38	19	8	4

the achievable parameters vs. the optimal estimate for random s-boxes. Note that in the table, the parameters $|\mathcal{B}_1|$, r, $t_1, t_2, \ldots, t_n$ correspond to the parameters in the achievable decomposition for randomly chosen s-boxes. The last column gives the probability of obtaining a successful decomposition for random S-boxes and for randomly chosen coefficients in the basis computation as well as the decomposition step. In all the cases 10 trials each were made to compute the probabilities except for the decomposition of 8-bit S-boxes over $\mathbb{F}_{2^2}$ where 100 trials each were made.

Table 3. Optimal and achievable parameters for random s-boxes.

Optimal/Achievable	(λ, n)	$\lvert\mathcal{B}_1\rvert$	r	$t_1, t_2, \ldots, t_n$	C^*	proba.
4-bit s-boxes						
Optimal	(2,2)	5	2	1,1	4	-
Achievable	(2,2)	5	2	1,1	4	0.2
6-bit s-boxes						
Optimal	(2,3)	8	4	4,2,2	12	-
Achievable	(2,3)	8	4	5,2,2	13	0.3
Optimal	(3,2)	6	3	3,2	8	-
Achievable	(3,2)	6	3	4,2	9	0.9
8-bit s-boxes						
Optimal	(2,4)	16	11	8,5,4,3	31	-
Achievable	(2,4)	16	11	9,6,5,3	34	0.02
Optimal	(4,2)	10	7	6,4	17	-
Achievable	(4,2)	10	7	7,4	18	1.0
9-bit s-boxes						
Optimal	(3,3)	15	11	11,6,5	33	-
Achievable	(3,3)	15	11	14,6,5	36	0.8

In the experiments, successive basis elements were added by products of random linear combinations of elements from the current basis. The basis $\mathcal{B}_1$ was chosen such that the corresponding matrix for the first coordinate function resulted in full rank (implying that the spanning property of the basis $\mathcal{B}_1$ was

satisfied). The basis was successively updated with the t_i products formed in the decomposition step of the ith coordinate function. While the parameter t_1 is invariant of the chosen s-box, the other t_i are indeed dependent on it. As we see from Table 3, the probabilities increase with the size of the field used for the decomposition.

Table 4 gives the concrete parameters to achieve decomposition for s-boxes of popular block ciphers (namely PRESENT [BKL+07], DES S1 and S8 [DES77], SC2000 S6 [SYY+02], CLEFIA S0 and S1 [SSA+07] and KHAZAD [BR00]). Note that for all the cases considered the parameters from Table 4 yield a decomposition. As above, the last column of the table gives the success probability over the random choice of the coefficients in the basis computation as well as the decomposition step. In all the cases 10 trials each were made to compute the probabilities except for the decomposition of 8-bit S-boxes over $\mathbb{F}_{2^2}$ where 100 trials each were made.

Table 4. Achievable parameters to decompose specific s-boxes.

s-box	(λ, n)	$\lvert\mathcal{B}_1\rvert$	r	$t_1, t_2, \ldots, t_n$	C^*	proba.
4-bit s-boxes						
PRESENT [BKL+07]	(2,2)	5	2	1,1	4	0.3
(6,4)-bit s-boxes						
DES S1 [DES77]	(2,3)	7	4	5,2	11	0.3
DES S8 [DES77]	(2,3)	7	4	5,2	11	0.5
6-bit s-boxes						
SC2000 S6 [SYY+02]	(2,3)	8	4	5,2,2	13	0.2
SC2000 S6 [SYY+02]	(3,2)	6	3	4,2	9	0.8
8-bit s-boxes						
CLEFIA S0 [SSA+07]	(4,2)	10	7	7,4	18	1.0
CLEFIA S0 [SSA+07]	(2,4)	16	11	9,5,4,3	32	0.01
CLEFIA S1 [SSA+07]	(4,2)	10	7	7,4	18	1.0
CLEFIA S1 [SSA+07]	(2,4)	16	11	9,6,5,3	34	0.01
KHAZAD [BR00]	(4,2)	10	7	7,4	18	1.0
KHAZAD [BR00]	(2,4)	16	11	9,5,4,3	32	0.02

6 Implementation

Based on our generic decomposition method, we now describe our implementation of an s-box layer protected with higher-order masking in ARM v7. We focused our study on the common scenario of a layer applying 16 8-bit s-boxes to a 128-bit state. We apply our generalized decomposition with parameters $n = m = 2$ and $\lambda = 4$ (medium case) to compare the obtained implementation to the ones for the two extreme cases:

– **Plain field case** ($\lambda = 8$, $n = 1$): standard CRV decomposition [CRV14];

– **Boolean case** ($\lambda = 1$, $n = 8$): Boolean decomposition from [GR16].

Our implementation is based on the decomposition obtained for the CLEFIA S0 s-box with parameters $(r, t_1, t_2) = (7, 7, 4)$. Note that it would have the same performances with any other 8-bit s-box with the same decomposition parameters (which we validate on all our tested random 8-bit s-boxes). The input (x_1, x_2) of each s-box is shared as $([x_1], [x_2])$ where

$$[x_i] = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \quad \text{such that} \quad \sum_{j=1}^{d} x_{i,j} = x_i \ . \tag{21}$$

Note that for those chosen parameters (n, m, λ), the input x_1 and x_2 are 4-bit elements, *i.e.* the inputs of the 8-bit s-boxes are split into 2. The output of the computation is a pair $([y_1], [y_2])$ where y_1 and y_2 are the two 4-bit coordinates of the s-box output.

We start from a basis that contains the input sharings $\{[z_1], [z_2]\} = \{[x_1], [x_2]\}$. Then for $i = 3$ to 21 each of the 18 multiplications is performed between two linear combinations of the elements of the basis, that is

$$[z_i] = [u_i] \odot [v_i] \ , \tag{22}$$

where $\odot$ denotes the ISW multiplication with refreshing of one of the operand (see [GR17] for details) and where

$$u_{i,j} = \sum_{k<i} \ell_{i,k}(z_{k,j}) \ \text{ and } \ v_{i,j} = \sum_{k<i} \ell'_{i,k}(z_{k,j}) \ \text{ for every } j \in [1, d], \tag{23}$$

for some linearized polynomials $\ell_{i,k}$ and $\ell'_{i,k}$ obtained from the s-box decomposition. Once all the products have been computed, the output sharings $[y_1]$ and $[y_2]$ are simple linear combinations of the computed $[z_i]$.

To make the most of the 32-bit architecture, the s-box evaluations are done eight-by-eight since we can fill a register with eight 4-bit elements. The ISW-based multiplications can then be parallelized as suggested in [GR17] except for the field multiplications between two shares. To perform those multiplications, we simply need to unpack the eight 4-bit elements in each 32-bit operand, and then to sequentially perform the 8 field multiplications. These field multiplications are fully tabulated which only takes 0.25KB of ROM on $\mathbb{F}_{16}$ (following the results of [GR17]). Using such a degree-8 parallelized ISW multiplication allows to improve by 58% the asymptotic gain compared to 8 serials ISW multiplications [GR17].

We compare our results with the bitslice implementation from [GR16] and the CRV-based optimized implementation from [GR17]. The former evaluates 16 s-boxes in parallel (based on bitslicing), whereas the latter performs 4 times 4 s-boxes in parallel (by filling 32-bits registers with four 8-bit elements). Table 5 summarizes the obtained performances in terms clock cycles, RAM consumption, and the random usage (needed by both the ISW multiplication and the

Table 5. Performances in clock cycles.

	CRV [GR17]	**Bitslice Decomposition [GR16]**	**Our implementations**
	$4 \times 4/\!\!/$ s-boxes	$16/\!\!/$ s-boxes	$2 \times 8/\!\!/$ s-boxes
Clock Cycle	$2576\,d^2 + 5476\,d + 2528$	$656\,d^2 + 19786\,d + 5764$	$2757\,d^2 + 17671\,d + 2402$
Code size	27.5 KB	4.6 KB	8.7 KB
RAM	$80d$ bytes	$644d$ bytes	$92d$ bytes
Random usage	$28d^2 - 28$ bytes	$61d^2 - 61$ bytes	$28d^2 - 28$ bytes

refreshing procedure) with respect to the masking order d and in terms of code size (including the look-up tables).

These results show that our implementation is slightly less efficient in terms of timings. However, it provides an interesting tradeoff in terms of memory consumption. Indeed the bitslice implementation has the drawback of being quite consuming in terms of RAM (with $644d$ bytes needed) and the CRV-based implementation has the drawback of having an important code size (27.5 KB) which is mainly due to the *half-table* multiplication and the tabulation linearized polynomials over $\mathbb{F}_{256}$. Our implementation offers a nice alternative when both RAM and code size are constrained. It also needs the same amount of randomness than the CRV decomposition and more than twice less than the bitslice decomposition.

Additionally, implementations based on our medium case decomposition might provide further interesting tradeoffs on smaller (8-bit or 16-bit) architectures where bitslice would be slowed down and where the optimized CRV-based implementation from [GR17] might be too consuming in terms of code size.

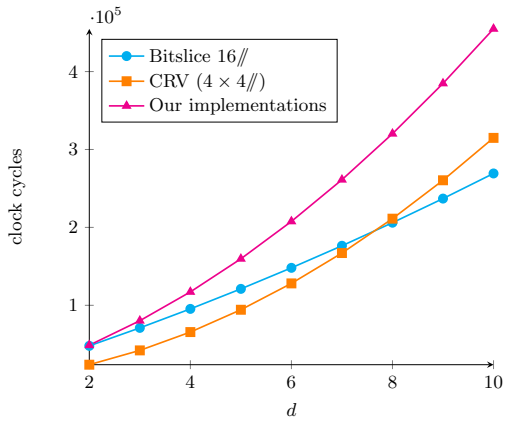

Fig. 1. Timings for $n = 8$.

Acknowledgements

We would like to thank Jürgen Pulkus for helpful discussions regarding choosing basis elements as random products. We would also like to thank the anonymous reviewers of CHES 2017 for valuable feedback that helped to improve the paper. Srinivas Vivek's work was partially supported by the European Union's H2020 Programme under grant agreement number ICT-644209 (HEAT).

References

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466, Vienna, Austria, September 10–13, 2007. Springer, Heidelberg, Germany.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *Journal of Cryptology*, 26(2):280–312, April 2013.

[BR00] Paulo Barreto and Vincent Rijmen. The Khazad legacy-level block cipher. First Open NESSIE Workshop, KU-Leuven, 2000. `http://www.cosic.esat.kuleuven.ac.be/nessie/`.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for S-boxes. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384, Washington, DC, USA, March 19–21, 2012. Springer, Heidelberg, Germany.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 170–187, Busan, South Korea, September 23–26, 2014. Springer, Heidelberg, Germany.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*,

volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[DES77] Data encryption standard. National Bureau of Standards, NBS FIPS PUB 46, U.S. Department of Commerce, January 1977.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Heidelberg, Germany.

[GR16] Dahmun Goudarzi and Matthieu Rivain. On the multiplicative complexity of boolean functions and bitsliced higher-order masking. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 457–478, Santa Barbara, CA, USA, August 17–19, 2016. Springer, Heidelberg, Germany.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, May 1-4, 2017, Proceedings*, Lecture Notes in Computer Science. Springer, 2017. to appear. Available on Cryptology ePrint Archive. `http://eprint.iacr.org/2016/264`.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[PS73] Mike Paterson and Larry J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, 2(1):60–66, 1973.

[PV16] Jürgen Pulkus and Srinivas Vivek. Reducing the number of non-linear multiplications in masking schemes. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 479–497, Santa Barbara, CA, USA, August 17–19, 2016. Springer, Heidelberg, Germany.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427, Santa Barbara, CA, USA, August 17–20, 2010. Springer, Heidelberg, Germany.

[RV13] Arnab Roy and Srinivas Vivek. Analysis and improvement of the generic higher-order masking scheme of FSE 2012. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 417–434, Santa Barbara, CA, USA, August 20–23, 2013. Springer, Heidelberg, Germany.

[SSA+07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In Alex Biryukov, editor, *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195, Luxembourg, Luxembourg, March 26–28, 2007. Springer, Heidelberg, Germany.

[SYY+02] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. The block cipher SC2000. In Mitsuru Matsui, editor, *Fast Software Encryption – FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 312–327, Yokohama, Japan, April 2–4, 2002. Springer, Heidelberg, Germany.