# Reducing the Number of Non-linear Multiplications in Masking Schemes[⋆]

Jürgen Pulkus[1] and Srinivas Vivek[2]

[1] Giesecke & Devrient, Munich, Germany
Juergen.Pulkus@gi-de.com
[2] University of Bristol
sv.venkatesh@bristol.ac.uk

**Abstract.** In recent years, methods to securely mask S-boxes against side-channel attacks by representing them as polynomials over finite binary fields have become quite efficient. A good cost model for this is to count how many non-linear multiplications are needed. In this work we improve on the current state-of-the-art generic method published by Coron–Roy–Vivek at CHES 2014 by working over slightly larger fields than strictly needed. This leads us, for example, to evaluate DES S-boxes with only 3 non-linear multiplications and, as a result, obtain 25% improvement in the running time for secure software implementations of DES when using three or more shares.
On the theoretical side, we prove a logarithmic upper bound on the number of non-linear multiplications required to evaluate any $d$-bit S-box, when ignoring the cost of working in unreasonably large fields. This upper bound is lower than the previous lower bounds proved under the assumption of working over the field $\mathbb{F}_{2^d}$, and we show this bound to be sharp. We also achieve a way to evaluate the AES S-box using only 3 non-linear multiplications over $\mathbb{F}_{2^{16}}$.

**Keywords:** side-channel countermeasure, masking, probing security, block cipher, software implementation, polynomial evaluation.

## 1 Introduction

Side-channel attacks are a realistic and serious threat for cryptographic implementations [Koc96,KJJ99]. These attacks have the potential to leak one or more sensitive intermediate variables that would otherwise be unavailable in a black-box execution of a cryptographic primitive. Block ciphers are typical targets of such attacks. Secret sharing, a.k.a. *masking*, is a popular technique to protect block cipher implementations against leakage of one or more sensitive intermediate variables. Depending on how a sensitive variable is split into shares, processed and then re-combined, and the formal leakage model used for security analysis, there are several generic higher-order masking schemes that can

[⋆] The final version of this article is available at www.springer.com. DOI: 10.1007/978-3-662-53140-2_23.

secure block cipher computations, with the secrets shared into as many shares as we desire [ISW03,GM11,PR11,CGP$^+$12,BFGV12,Cor14,BFG15]. Indeed, these schemes can be used to secure any circuit.

The most popular among existing generic masking schemes for block cipher implementations are those where the secrets are additively shared. This is in part due to the effectiveness, efficiency and simplicity of additive masking [CJRR99,ISW03,PR13,DDF14,DFS15,BFG15]. Over binary fields this type of masking has also been called as *Boolean masking*. In fact, the very first generic higher-order masking scheme, due to Ishai, Sahai and Wagner [ISW03] (henceforth referred to as the *ISW method*), is based on additive masking. Their method can be used to secure arbitrary Boolean circuits in the so-called *probing model*, where an adversary can choose to leak, say, $t$ intermediate variables and the scheme is secure so long as the number of shares $s \geq 2t + 1$. Though working with Boolean circuits is probably well-suited to hardware implementations, representing a computation as a Boolean circuit will lead to huge overheads in software implementations. Nonetheless, this method and the probing security framework introduced in their work formed the basis for most of the later masking schemes. Rivain and Prouff [RP10] adapted the ISW method to secure AES by representing its S-box as an arithmetic circuit over $\mathbb{F}_{2^8}$.

**CGPQR Method**. Carlet *et al.* [CGP$^+$12] adapted the ISW method to secure software implementations of arbitrary block ciphers over binary finite fields $\mathbb{F}_{2^n}$ (hereafter referred to as the *CGPQR method*). For an additive masking scheme processing $\mathbb{F}_2$-linear or affine functions in the presence of shares is straightforward. Hence the main challenge is to securely process non-linear functions. Since in a block cipher the only non-linear operations are the S-box table lookups, the technique used in the CGPQR method to securely mask such table lookups is to first represent a $d$-to-$r$-bit S-box function ($d \geq r$) as a univariate polynomial over a binary finite field $\mathbb{F}_{2^d}$. Then this polynomial is evaluated in the presence of shares using the following operations: *addition* (of two polynomials over $\mathbb{F}_{2^d}$), *scalar multiplication* (i.e., multiplication of a polynomial by a constant from $\mathbb{F}_{2^d}$), squaring (of a polynomial over $\mathbb{F}_{2^d}$), and multiplications of two distinct polynomials (a.k.a. *non-linear multiplications*). While additions, scalar multiplications and squarings are $\mathbb{F}_2$-linear operations, the non-linear multiplications, as the name suggests, are *not* $\mathbb{F}_2$-linear. To process a non-linear multiplication in the CGPQR method, an adaptation of the technique used in the ISW method to mask (non-linear) AND gates is utilised. The overhead caused by the CGPQR method (relative to unshared evaluation), in terms of both the time and the randomness required, to securely mask a non-linear multiplication is $\mathcal{O}(s^2)$, where $s$ is the number of input shares. For a linear or affine function the overhead is only $\mathcal{O}(s)$.

**Relation to Polynomial Evaluation**. One of the relatively well-understood approaches to analysing and improving the efficiency of the CGPQR method is to investigate the problem of evaluating polynomials over binary finite fields. The goal is to minimise the number of non-linear multiplications needed to evaluate a polynomial over $\mathbb{F}_{2^d}$, while *ignoring* the cost of additions, scalar multiplications and squarings. As the works of Carlet *et al.* [CGP$^+$12], Roy and Vivek [RV13], and Coron, Roy and Vivek [CRV14,CRV15] demonstrate, this cost model of minimising the non-linear multiplications while evaluating an S-box polynomial has turned out to be a reasonably effective way to model the overall cost of processing a block cipher in software implementations, as long as one makes sure that the use of linear operations is not made "unreasonably" large.

In [CGP$^+$12], two methods to evaluate arbitrary polynomials over $\mathbb{F}_{2^d}$ are presented that are tailored to the non-linear cost model: the *cyclotomic-class* method (having complexity $\Omega(2^d/d)$) and the *parity-split* method (having proven complexity $\mathcal{O}(2^{d/2})$. These two methods were applied to various S-box polynomials to understand their complexity in terms of non-linear multiplications. In [RV13], improved evaluation techniques for various specific S-box polynomials were presented. In particular, it was shown that the 6-to-4-bit DES S-boxes can be evaluated with 7 non-linear multiplications, while 8-bit (i.e., 8-to-8-bit) CAMELLIA and CLEFIA S-boxes can be evaluated with 15 or 16 non-linear multiplications. The work of [RV13] also initiated a formal analysis of this cost model and established lower bounds on the necessary number of non-linear multiplications required to evaluate any polynomial over $\mathbb{F}_{2^d}$. In particular, they showed that, under certain representation over $\mathbb{F}_{2^6}$, the DES S-box polynomials need at least 3 non-linear multiplications to evaluate them, while the PRESENT S-box polynomial over $\mathbb{F}_{2^4}$ needs at least 2 non-linear multiplications.

**CRV Method**. In [CRV14,CRV15], Coron, Roy and Vivek proposed an improved method (henceforth referred to as the *CRV method*) to evaluate arbitrary polynomials over $\mathbb{F}_{2^d}$. Their method has a heuristic worst-case complexity of $\mathcal{O}(2^{d/2}/\sqrt{d})$ non-linear multiplications. They also show that the complexity of $\mathcal{O}(2^{d/2}/\sqrt{d})$ is optimal for any method to evaluate arbitrary polynomials over $\mathbb{F}_{2^d}$. Currently, w.r.t. the non-linear multiplications cost model, the CRV method is the most efficient way to implement the CGPQR countermeasure.

In the CRV method, a $d$-to-$r$-bit S-box $S$ is represented by a polynomial $P(X) \in \mathbb{F}_{2^d}[X]$ that is actually computed in the process. The $d$-bit and the $r$-bit strings are identified with the elements of $\mathbb{F}_{2^d}$. The polynomial $P(X)$ satisfies the property that its evaluation on the elements of $\mathbb{F}_{2^d}$ produces output elements of $\mathbb{F}_{2^d}$ that agree in the lower-order $r$-bits with the corresponding S-box outputs. Briefly the CRV method for a generic $d$-to-$r$-bit S-box is as follows:

Step 1: Pre-compute a collection of monomials $L$ in $\mathbb{F}_{2^d}[X]$ (a) that is closed w.r.t. squaring (because squarings are free) (b) has the property that $L \cdot L$ generates all the monomials $X^i$ $(i = 0, 1, \ldots, 2^d - 1)$.

Step 2: Consider the following relation over $\mathbb{F}_{2^d}[X]$:

$$P(X) = \sum_{i=1}^{t-1} p_i(X) \cdot q_i(X) + p_t(X) \mod X^{2^d} + X \tag{1}$$

for some chosen parameter $t$, where the polynomials $p_i(X)$ and $q_i(X)$ have monomials only from the set $L$, and the polynomials $q_i(X)$ are randomly chosen but the values of $P(X)$ and the coefficients of $p_i(X)$ are *unknown*. Next they write down a set of $r \cdot 2^d$ linear equations over $\mathbb{F}_2$ (in the unknown bits), corresponding to each S-box output bit, by evaluating the above relation at the elements of $\mathbb{F}_{2^d}$. Finally, the unknown bits are obtained by solving the resulting linear system over $\mathbb{F}_2$ whose matrix has dimension $r \cdot 2^d \times d \cdot t \cdot |L|$, which is approximately $r \cdot 2^d \times r \cdot 2^d$. The total number of non-linear multiplications required is about $t - 1 + |L|/d$.

It is shown in [CRV15] that any 4-bit S-box can be evaluated with 2 non-linear multiplications in the worst case (which is optimal), any 6-bit S-box with at most 5, any 6-to-4-bit S-box (in particular, DES S-boxes) with at most 4, any 8-bit S-box with at most 10 non-linear multiplications (cf. Table 1). As, in a block cipher, the time required for S-box table lookups grows quadratically with the number of shares, seemingly marginal reductions in the count of non-linear multiplications per S-box evaluation indeed lead to significant gains in the overall execution time, as demonstrated in [Cor14,CRV15].

One obvious approach to improve the CRV method is to simultaneously solve for the unknown coefficients of both the set of polynomials $p_i(X)$ and $q_i(X)$ (including $P(X)$) in Step 2 of the CRV method described above, instead of linearising (1) by choosing random polynomials $q_i(X)$. This results in $r \cdot 2^d$ multivariate homogeneous quadratic equations over $\mathbb{F}_2$ in approximately $d \cdot 2^d$ variables. To our knowledge, determining the roots of such a system of equations seems infeasible with current techniques even for small values of $d = 6$ or $d = 8$. Hence it is of interest to find alternative ways to reduce the parameters of the CRV method (particularly, the parameters $t$ and $L$) that affect the total number of non-linear multiplications for the S-box polynomials. This is one of the main themes of this paper.

## 1.1 Our Contribution

We give an improved generic method to reduce the number of non-linear multiplications required to evaluate various S-box polynomials. Our method may be viewed as an extension of the CRV method (cf. page 3). While in the CRV method and other previous works the inputs/outputs of a $d$-to-$r$-bit S-box are naturally identified with the elements of $\mathbb{F}_{2^d}$, we instead encode them in fields $\mathbb{F}_{2^n}$, where $n \geq d$. Our heuristic analysis seems to suggest that the complexity of the CRV method improves by a factor of two in the limiting case, though both the methods have the same heuristic asymptotic (worst-case) complexity of $\mathcal{O}(2^{d/2}/\sqrt{d})$ non-linear multiplications.

From a technical point of view, apart from the problem of encoding mentioned above, the main and the only other difference between our method and the CRV method is in the selection of the following two parameters: $L$ (the pre-computed monomial list) and $t$ (the number of summands in the decomposition in (1)). Once these parameters are carefully determined, then the remaining steps to obtain a decomposition of the form (1) by setting up a linear system of equations is exactly the same. Since in the matrix step of the CRV method (cf. page 3) we heuristically need $n \cdot t \cdot |L| \approx r \cdot 2^d$, it is evident that we could end up with smaller values of $t$, and hence a reduction in the total number of non-linear multiplications required. Some technical hurdles arise due to the fact that we would not gain anything if we insist, as in the CRV method, that the pre-computed set of monomials $L$ must span all monomials in $\mathbb{F}_{2^n}[X]$. Our generic method and its analysis is presented in Section 2.

Our method leads to improvements for most of the S-boxes found in practice. Table 1 lists the (worst-case) cost of processing *arbitrary* $d$-to-$r$-bit S-boxes using our method over $\mathbb{F}_{2^8}$ and $\mathbb{F}_{2^{16}}$, and compares these with those of the previous methods. In particular, any 6-to-4 bit S-box, including all the DES S-boxes, now need at most 3 non-linear multiplications to evaluate them instead of the previous best of 4 non-linear multiplications required by the CRV method that works over $\mathbb{F}_{2^6}$ in this case (cf. Table 2). We discuss how to select suitable parameters for various S-box dimensions in Section 2.2.

We have made a proof-of-concept implementation in software of our improved method for DES. As Table 5 suggests, the CGPQR method combined with our technique outperforms the CGPQR+CRV method by around 25% in the overall processing time of the block cipher when there are 3 shares, and even better when there are greater numbers of shares. Our implementation also needs less (RAM) memory and fewer calls to a Pseudo Random Generator (PRG) (that outputs bytes) than the CRV method. We believe that since it is convenient to manipulate bytes in a software implementation, working over $\mathbb{F}_{2^8}$ instead of $\mathbb{F}_{2^6}$ or $\mathbb{F}_{2^7}$ should not cause any noticeable overhead. This reasoning is also confirmed by our above implementation, the details of which are presented in Section 2.3. Our improvements obtained by working over $\mathbb{F}_{2^{16}}$ could possibly be interesting for microprocessors such as ARM with Neon core [Lim13] that has a SIMD instruction to perform several parallel multiplications of two degree-7 polynomials over $\mathbb{F}_2$ represented as bytes. This instruction can be used to perform parallel multiplications in $\mathbb{F}_{2^{16}}$ with considerably less overhead than on a sequential processor thanks to Barrett reduction [Bar86,WVGX15]. But the downside is that the number of calls to a PRG is still double compared to the case of $\mathbb{F}_{2^8}$. Besides, note that such processors can also be targets of side-channel attacks [GMPT15].

Finally, in Section 3, we analyse the advantage and the limitations of using larger fields $\mathbb{F}_{2^n}$ $(n \geq d)$ to encode the input/output bit-strings of $d$-to-$r$-bit S-boxes as arbitrary subspaces in $\mathbb{F}_{2^n}$. Note that since additive masking is $\mathbb{F}_2$-linear, the set of encodings must be an ($\mathbb{F}_2$-linear) subspace of $\mathbb{F}_{2^n}$ (when viewed as an $\mathbb{F}_2$-vector space). We *prove* a logarithmic upper bound of $\lceil \log_2 d \rceil$, which is also optimal, on the complexity of evaluating $d$-to-$r$-bit S-boxes, when working over some huge field extension of $\mathbb{F}_{2^d}$. We stress that this result does not contradict the exponential lower bound results of [CRV15] as they hold over the (smaller field) $\mathbb{F}_{2^d}$. Using the techniques introduced to obtain the above results, we achieve a way to evaluate the AES S-box using only 3 non-linear multiplications over $\mathbb{F}_{2^{16}}$, instead of 4 non-linear multiplications over $\mathbb{F}_{2^8}$. We then generalise the lower bound results of [CRV15] to determine a lower bound on the exact complexity of generic $d$-to-$r$-bit S-boxes when working over any specified field $\mathbb{F}_{2^n}$ $(n \geq d)$.

| $(d, r)$ | $(4, 4)$ | $(5, 5)$ | $(6, 4)$ | $(6, 6)$ | $(7, 7)$ | $(8, 8)$ |
|---|---|---|---|---|---|---|
| Cyclotomic Class method [CGP$^+$12] | 3 | 5 | 11 | 11 | 17 | 33 |
| Parity-Split method [CGP$^+$12] | 4 | 6 | 10 | 10 | 14 | 22 |
| CRV method [CRV15] | 2 | 4 | 4 | 5 | 7 | 10 |
| **Our method** (over $\mathbb{F}_{2^8}$) | **2** | **3** | **3** | **4** | **6** | **10** |
| **Our method** (over $\mathbb{F}_{2^{16}}$) | **2** | **3** | **3** | **3** | **4** | **6** |

**Table 1.** Comparison of the worst-case complexity of generic methods for various $d$-to-$r$-bit S-boxes.

### 1.2 Related Works

Another generic masking scheme based on the additive masking is by Coron [Cor14]. This countermeasure is a generalisation of the table-recomputation technique [CJRR99,SP06] to the higher-order setting. As shown in [CRV15], the CGPQR method combined with the CRV method outperforms this countermeasure, both asymptotically and in practice, w.r.t. time and memory complexity, and also the randomness required.

As far as the CGPQR method is concerned, there are other interesting approaches to improving its efficiency than by minimising the number of non-linear multiplications. One such way was introduced by Coron *et al.* [CPRR13] and further considered by Grosso, Prouff and Standaert [GPS14]. This approach is based on the observation that certain types of non-linear multiplications are more than efficient than the rest. Hence the efficiency can be gained by trading the costlier non-linear multiplications for more efficient ones. Recently, Carlet *et al.* [CPRR15] introduced techniques based on the algebraic decomposition of a non-linear function as a sequence of low algebraic-degree functions. The CGPQR method combined with their technique outperforms the CGPQR+CRV method in many realistic scenarios.

It must be stressed that the above approaches to making the CGPQR method more efficient are not mutually exclusive of one another but, indeed, complementary. In fact, the improvements w.r.t. the non-linear multiplications cost model have motivated the approaches of [GPS14,CPRR15]. Finally, we would like to note that the relative simplicity of the non-linear multiplications cost model has made it amenable to a rigorous analysis, in particular, the lower bound results in [RV13,CRV15], while relatively little is known about the other cost models. Also, this cost model and its variant where the circuit depth w.r.t. non-linear multiplications also matters have found applications in fully homomorphic encryption and multi-party computation settings [GHS12a,GHS12b,ARS$^+$15]. We do not consider such applications in this work, and hence, prefer to work in the non-linear multiplications cost model.

## 2 Improved Generic Method for S-boxes

Consider a $d$-to-$r$-bit S-box, where $d \geq r$. We identify the $d$-bit and the $r$-bit strings with the elements of $\mathbb{F}_{2^n}$ $(r, d \leq n)$ in the "usual" way. That is, let $\mathbb{F}_{2^n} = {}^{\mathbb{F}_2[Y]}/_{(g(Y)\cdot\mathbb{F}_2[Y])}$, where $g(Y) \in \mathbb{F}_2[Y]$ is an irreducible polynomial over $\mathbb{F}_2$ with $\mathsf{deg}(g) = n$ that is used to represent $\mathbb{F}_{2^n}$. A $d$-bit string is encoded as follows

$$\begin{aligned} \mathsf{E}_{d,n} : \{0,1\}^d &\rightarrow \mathbb{F}_{2^n}, \\ \langle b_{d-1}, b_{d-2}, \ldots, b_0 \rangle &\mapsto \sum_{i=0}^{d-1} b_i\, Y^i. \end{aligned}$$

An element of $\mathbb{F}_{2^n}$ is decoded to a $d$-bit string by dropping its corresponding higher-degree coefficients

$$\begin{aligned} \mathsf{D}_{n,d} : \mathbb{F}_{2^n} &\rightarrow \{0,1\}^d, \\ \sum_{i=0}^{n-1} b_i\, Y^i &\mapsto \langle b_{d-1}, b_{d-2}, \ldots, b_0 \rangle . \end{aligned}$$

The functions $\mathsf{E}_{r,n} : \{0,1\}^r \rightarrow \mathbb{F}_{2^n}$ and $\mathsf{D}_{n,r} : \mathbb{F}_{2^n} \rightarrow \{0,1\}^r$ are similarly defined, as are $\mathsf{E}_{n,n} : \{0,1\}^n \rightarrow \mathbb{F}_{2^n}$ and $\mathsf{D}_{n,n} : \mathbb{F}_{2^n} \rightarrow \{0,1\}^n$.

*Remark 1.* The composition map $\mathsf{D}_{d,d} \circ \mathsf{E}_{d,n} : \mathbb{F}_{2^d} \rightarrow \mathbb{F}_{2^n}$ is a group homomorphism w.r.t. addition. But, in general, this map is not homomorphic w.r.t. multiplication.

We say that a polynomial $P(X) \in \mathbb{F}_{2^n}[X]$ evaluates a $d$-to-$r$ bit S-box $S$ if the trailing $r$ bits of its evaluation on the encodings of every $d$-bit string matches with the output of $S$. Formally,

$$S(i) = \mathsf{D}_{n,r}\left(P\left(\mathsf{E}_{d,n}(i)\right)\right), \qquad \forall i \in \{0,1\}^d. \tag{2}$$

Our goal is to find a polynomial representation for a given S-box whose evaluation requires as small a number of non-linear multiplications as possible.

Let $C_\alpha^n$ denote the *cyclotomic class* of $\alpha$ w.r.t $n$ $(n \geq 1, 0 \leq \alpha < 2^n)$ [CGP+12,RV13], that is, $C_0^n = \{0\}$, $C_{2^n-1}^n = \{2^n - 1\}$ and

$$C_\alpha^n := \left\{\alpha \cdot 2^i \pmod{2^n - 1} \ : \ i = 0, 1, \ldots, n-1\right\} \text{ for } 0 < \alpha < 2^n - 1.$$

For any subset $\Lambda \subseteq \{0, 1, \ldots, 2^n - 1\}$, let $X^\Lambda$ denote the set $X^\Lambda := \left\{X^i \ : \ i \in \Lambda\right\} \subseteq \mathbb{F}_{2^n}[X]$. Define $X^\Lambda \cdot X^\Lambda := \left\{X^i \cdot X^j \ : \ i, j \in \Lambda\right\}$. Finally, $\mathcal{P}(X^\Lambda) \subseteq \mathbb{F}_{2^n}[X]$ denotes the set of all polynomials (of degree at most $2^n - 1$) that have their monomials only from $X^\Lambda$.

### 2.1 Our Method

Our method is a variant of the CRV method [CRV15]. The main difference is that we allow $n \geq d$, which requires a change in the way the pre-computed list of monomials is chosen. Our method is summarised in Algorithm 1.

**Step 1.** Choose a collection $\mathcal{T}'$ of $\ell$ cyclotomic classes w.r.t. $d$:

$$\mathcal{T}' = \left\{ C^d_{\alpha_1=0}, C^d_{\alpha_2=1}, C^d_{\alpha_3}, \ldots, C^d_{\alpha_\ell} \right\}. \tag{3}$$

Let

$$L' = \bigcup_{C^d_{\alpha_i} \in \mathcal{T}'} C^d_{\alpha_i}. \tag{4}$$

Now "lift" the above collection of cyclotomic classes w.r.t. $d$ to a collection w.r.t. $n$. That is, for every $C^d_{\alpha_i}$, we choose $C^n_{\alpha_i}$ for *some* representative $\alpha_i \in C^d_{\alpha_i}$. Define

$$\mathcal{T} = \left\{ C^n_{\alpha_1=0}, C^n_{\alpha_2=1}, C^n_{\alpha_3}, \ldots, C^n_{\alpha_\ell} \right\}. \tag{5}$$

Let

$$L = \bigcup_{C^n_{\alpha_i} \in \mathcal{T}} C^n_{\alpha_i}. \tag{6}$$

Note that we will be using only the collection $\mathcal{T}$ and the set $L$ in the decomposition step of our method (cf. (8)).

*Heuristic 1.* We assume that it is possible to choose a $\mathcal{T}$ as specified above (for any $\ell$ "sufficiently smaller" than $2^d$) in such a way that:

1. each cyclotomic class (except $C^n_0$) in $\mathcal{T}$ has (maximal) length $n$,
2. $X^L$ can be computed using only $\ell - 2$ non-linear multiplications,
3. $X^{\{0,1,2,\ldots,X^{2^d-1}\}} \subseteq X^{L'} \cdot X^{L'} \subseteq \mathbb{F}_{2^d}[X]$. We refer to this property by saying that $X^{L'}$ spans the set $\{1, X, X^2, \ldots, X^{2^d-1}\}$ in $\mathbb{F}_{2^d}[X]$.

The first two heuristics above are also used in the CRV method. The difference is in the third heuristic (Heuristic 1.3). Note that the condition is only on the set $L'$, not $L$. Note that in the CRV method it is required that $X^L$ spans $\{1, X, X^2, \ldots, X^{2^n-1}\}$ in $\mathbb{F}_{2^n}[X]$ (in their case $n = d$). But as we prescribe only the values on $\mathbb{F}_{2^d}$, not on all of $\mathbb{F}_{2^n}$ we do not need such a strong condition. Indeed if we use this (stronger) condition from the CRV method, then we cannot expect any improvement over the CRV method (it will actually be worse since we are working in a bigger field).

*Remark 2.* In general, $X^L$ does *not* span $\{1, X, X^2, \ldots, X^{2^n-1}\}$ nor $\{1, X, X^2, \ldots, X^{2^d-1}\}$ in $\mathbb{F}_{2^n}[X]$.

So we will make another assumption that turns out to be true experimentally for instances of practical relevance.

*Heuristic 2.* Corresponding to any $d$-to-$r$-bit S-box $S$, there exists a polynomial in $\mathcal{P}(X^L \cdot X^L) \subseteq \mathbb{F}_{2^n}[X]$ that evaluates $S$.

The CRV method does not need to make the above assumption as the condition is implied by Heuristic 1.3 when $n = d$.

*Remark 3.* As noted in [RV13, Proof of Theorem 1], if $d|n$, then the cyclotomic classes $C_u^n$ "lie above" $C_z^d$ for every $u \in C_z^d$. That is, $\big(\delta \mod 2^d - 1\big) \in C_z^d$ for every $\delta \in C_v^n$ and every $v \in C_z^d$.

Note that

$$|L| = 1 + n \cdot (\ell - 1)\,. \tag{7}$$

We would like to choose as small a value for $\ell$ as possible but still satisfying Heuristic 1.3 (as we shall soon see, that $\ell$ must satisfy another (relatively milder) condition in Heuristic 4). We use the following heuristic from the CRV method for choosing a value of $\ell$.

*Heuristic 3.* There exists a collection of cyclotomic classes $\mathcal{T}'$ (w.r.t. $d$) satisfying Heuristic 1.3 such that $\ell \approx \sqrt{\frac{2^d}{d}}$.

**Step 2.** Then, as in the CRV method [CRV15, Section 4.3], we choose $t-1$ random polynomials $q_i(X) \overset{\$}{\leftarrow} \mathcal{P}(X^L) \subseteq \mathbb{F}_{2^n}[X]$, for some parameter $t$ to be determined later, that have their monomials only from $X^L$. Then we try to find $t$ polynomials $p_i(X) \in \mathcal{P}(X^L)$ such that

$$P(X) = \sum_{i=1}^{t-1} p_i(X) \cdot q_i(X) \,+\, p_t(X) \bmod X^{2^n} + X \tag{8}$$

evaluates $S$.

Note that Heuristic 3 guarantees that the decomposition of (8) exists for every $d$-to-$r$-bit S-box $S$ for some $t \leq |L| \cdot (|L| - 1)$. But we need to find as small a value of $t$ as possible for a chosen $L$.

The unknown coefficients of the polynomials $p_i(X)$ are obtained by evaluating $P(X)$ at $\mathsf{E}_{d,n}(j)$ $\forall j \in \{0,1\}^d$ and then writing the resulting set of linear equations over $\mathbb{F}_2$ instead of $\mathbb{F}_{2^n}$. That is, we obtain a system of linear equations over $\mathbb{F}_2$ with each equation corresponding to an output bit of $S(j)$. Note that the unknowns in these equations correspond to the (unknown) $n$ "bits" of the unknown coefficients (from $\mathbb{F}_{2^n}$) of $p_i(X)$. Denote the resulting system of linear equations as

$$A \cdot \boldsymbol{c} = \boldsymbol{b}, \tag{9}$$

where the matrix $A$ over $\mathbb{F}_2$ will have $r \cdot 2^d$ rows and $t \cdot |L| \cdot n$ columns, the $\mathbb{F}_2$-vector $\boldsymbol{c}$ corresponds to the unknown bits of the (to-be-determined) coefficients of $p_i(X)$, and the $\mathbb{F}_2$-vector $\boldsymbol{b}$ corresponds to the bits of the outputs of the S-box $S$. We can solve the above linear equation for any $\boldsymbol{b}$ if $A$ has rank $r \cdot 2^d$. We make the following assumption, similar to the CRV method, that says that if the number of columns exceed the number of rows, then the matrix $A$ has full rank $r \cdot 2^d$.

*Heuristic 4.* The condition $t \cdot |L| \cdot n \geq r \cdot 2^d$ suffices for $A$ to have (full) rank $r \cdot 2^d$.

Once the solution vector $\boldsymbol{c}$ is computed, then the unknown coefficients (from $\mathbb{F}_{2^n}$) of the polynomials $p_i(X)$, and hence the polynomial $P(X)$, are readily obtained. This completes the description of our method.

*Remark 4.* If the matrix $A$ has full rank $(r \cdot 2^d)$ for a randomly chosen set of polynomials $q_i(X) \in \mathcal{P}(X^L)$, then this same set of polynomials will yield the decomposition of (8) for any $d$-to-$r$-bit S-box.

**Algorithm 1** Our method to evaluate generic S-boxes

```
Input: A $d$-to-$r$-bit S-box table $S$.
Output: Polynomials $p_i(X), q_i(X) \in \mathbb{F}_{2^n}[X]$ such that $P(X) = \sum_{i=1}^{t-1} p_i(X) \cdot q_i(X) + p_t(X)$ satisfies (2).
1: Choose a collection $\mathcal{T}$ of $\ell$ cyclotomic classes w.r.t. some $n \geq d$ that satisfies Heuristics 1 and 3.
2: Compute $X^L$, where $L \leftarrow \bigcup_{C \in \mathcal{T}} C$.
3: Choose $t$ such that $t \cdot |L| \cdot n \geq r \cdot 2^d$.
4: For $1 \leq i \leq t$, choose $q_i(X) \xleftarrow{\$} \mathcal{P}\left(X^L\right)$.
5: Set up a linear system of equations over $\mathbb{F}_2$, $A \cdot \boldsymbol{c} = \boldsymbol{b}$, to solve for the $\mathbb{F}_2$-vector $\boldsymbol{c}$ that corresponds to the unknown coefficients of the polynomials $p_i(X)$ (cf. (9)).
6: Construct the polynomials $p_i(X)$ from the solution vector $\boldsymbol{c}$.
```

**Complexity Analysis**. The number of non-linear multiplications required to pre-compute the set $X^L$ is $\ell - 2$, and the number required in (8) is $t - 1$. Hence in total the number of non-linear multiplications required is

$$M_{d,r,n} = \ell - 2 + t - 1 = \ell + t - 3. \tag{10}$$

From Heuristic 4, we get the condition

$$t \geq \frac{r \cdot 2^d}{|L| \cdot n}.$$

By substituting from (10) and (7) in the above inequality, we get

$$M_{d,r,n} \geq \ell - 3 + \frac{r \cdot 2^d}{(1 + n \cdot (\ell - 1)) \cdot n}.$$

Since, from Heuristic 3, we can set $\ell \approx \sqrt{\frac{2^d}{d}}$, we obtain from the above inequality

$$M_{d,r,n} \approx \sqrt{\frac{2^d}{d}} - 3 + \frac{r \cdot 2^d}{n \cdot \left(1 + n \cdot \left(\sqrt{\frac{2^d}{d}} - 1\right)\right)} \tag{11}$$

Note that if $d = r = n$, then we recover an estimate close to that found in [CRV15, Section 2.2]. If $n \gg \sqrt[4]{2^d \cdot d \cdot r^2}$, then

$$M_{d,r,\infty} \approx \sqrt{\frac{2^d}{d}}.$$

Hence in the limiting case the complexity of our method is half that of the CRV method.

**Numerical Experiments.** In Table 2, we compare the estimate of (11) (on rounding up to the successive integer) with the observed complexity for various cases of practical interest. It turns out that the observed values are close to the estimated values.

*Remark 5.* Experiments tend to indicate that the value of $t$ *cannot* be made arbitrarily small with increasing values of $n$. The resulting ranks of the matrices seem to saturate after a certain value of $n$. This, of course, has to do with the structure of the pre-computed set $X^L$. But the dependency is currently unclear, and hence we are unable to give a lower bound on the value of $t$, unlike the case of $\ell$.

| $d$ | 4 | | | 5 | | | 6 | | | | | | 7 | | | 8 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $r$ | 4 | | | 5 | | | 4 | | | 6 | | | 7 | | | 8 | |
| $n$ | 4 | 8 | 16 | 5 | 8 | 16 | 4 | 8 | 16 | 6 | 8 | 16 | 7 | 8 | 16 | 8 | 16 |
| Estimated $M_{d,r,n}$ | 3 | 0 | 0 | 4 | 2 | 0 | 4 | 2 | 1 | 5 | 3 | 1 | 7 | 6 | 3 | 10 | 5 |
| Observed $M_{d,r,n}$ | 2 | 2 | 2 | 4 | 3 | 3 | 4 | 3 | 3 | 5 | 4 | 3 | 7 | 6 | 4 | 10 | 6 |

**Table 2.** Expected and observed (worst-case) complexity $M_{d,r,n}$ of evaluating $d$-to-$r$-bit S-boxes over $\mathbb{F}_{2^n}$ (cf. (11)).

**Linear Operations.** An upper bound on the number of additions (over $\mathbb{F}_{2^n}$) required by our method to evaluate the polynomial $P(X)$ in (8) is $(2t - 1) \cdot$

$(|L|-1)+(t-1)$ since each of the polynomials $p_i(X)$ and $q_i(X)$ have at most $|L|$ non-zero coefficients. Since, from Heuristic 4, we have $t \cdot |L| \approx \frac{r \cdot 2^d}{n}$, an upper bound on the number of additions is about $\frac{r \cdot 2^{d+1}}{n}$. Note that working in bigger fields can lead to smaller numbers of additions, though each such field addition operation now takes a greater number of bit operations.

The number of scalar multiplications (over $\mathbb{F}_{2^n}$) that is required is at most $(2t-1) \cdot (|L|) \approx \frac{r \cdot 2^{d+1}}{n}$, while the number of squarings (over $\mathbb{F}_{2^n}$) that is required is $n \cdot (\ell - 1) \approx n \cdot \sqrt{\frac{2^d}{d}}$.

## 2.2 Concrete Parameters for Various S-boxes

Table 3 suggests how to choose the parameters $t$ and $L$ in Algorithm 1 for various $d$-to-$r$-bit S-box dimensions depending on the choice of $n$. If these parameters of Algorithm 1 are chosen as indicated, then the number of non-linear multiplications required to evaluate any S-box of given dimension is upper bounded as specified by Table 2. For the special case of $d = r = n$ the parameters are as suggested in [CRV15, Appendix B] except for the case $d = r = n = 8$.

As Remark 4 suggests, once a chosen set of random polynomials $q_i(X)$ in Algorithm 1 yields the decomposition of (8) for a given $d$-to-$r$-bit S-box, then the same set of $q_i(X)$ will yield a decomposition for any other S-box of the same dimension. In practice, we have observed that a randomly chosen set of polynomials $q_i(X)$ almost always yield the decomposition of (8).

| $d$ | 4 | | 5 | | 6 | | | | | 7 | | | 8 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $r$ | 4 | | 5 | | 4 | | 6 | | | 7 | | | 8 | |
| $n$ | 4 | 8 | 5 | 8 | 4 | 8 | 6 | 8 | 16 | 7 | 8 | 16 | 8 | 16 |
| $l$ | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 6 | 6 |
| $t$ | 2 | 2 | 3 | 2 | 3 | 2 | 4 | 3 | 2 | 5 | 4 | 2 | 7 | 3 |
| $\|L\|$ | 9 | 17 | 16 | 25 | 19 | 25 | 19 | 25 | 49 | 29 | 33 | 65 | 41 | 81 |

**Table 3.** Choosing parameters $l$, $t$ and $L$ for evaluating $d$-to-$r$-bit S-boxes over $\mathbb{F}_{2^n}$, where $L$ is always the union of the first $l$ elements of $\{C_0^n, C_1^n, C_3^n, C_7^n, C_{29}^n, C_{87}^n\}$

## 2.3 Software Implementation of DES

We have performed a software implementation of the CGPQR method [CGP$^+$12] combined with our technique for the DES block cipher [oST93] that needs only 3 non-linear multiplications over $\mathbb{F}_{2^8}$. Note that DES uses eight 6-to-4-bit S-boxes. We used the C implementation of the CGPQR method combined with the improvements of [RV13] and [CRV15] that is publicly available from [Cor13]. For a fair comparison, we have compared our improvement only with the CGPQR method combined with the improvements of [RV13] (that needs 7 non-linear multiplications) and the CRV method (that needs 4 non-linear multiplications) [CRV15], both of which are analysed in the non-linear multiplication cost model. The results are presented in Table 5.

We decomposed the DES S-boxes as

$$P_{\mathtt{DES}}(X) = p_1(X) \cdot q_1(X) + p_2(X),$$

where $p_1, q_1, p_2 \in \mathcal{P}(X^L) \subseteq \mathbb{F}_{2^8}[X]$, $L = C_0^8 \cup C_1^8 \cup C_3^8 \cup C_7^8$, and the coefficients of $q_1$ are randomly chosen from $\mathbb{F}_{2^8}$ (cf. Table 3 and Algorithm 1). Table 4 describes a polynomial $q_1$ that will yield the above decomposition for each of the 8 S-boxes of DES.

| |
|---|
| $(a^2)\cdot x^{224}+(a^7+a^6+a^5+a^4+a^2+1)\cdot x^{193}+(a^7+a^4+a+1)\cdot x^{192}+(a^7+a^5+a^3+a^2+a+1)\cdot x^{131}+(a^6+a^3+1)\cdot x^{129}+(a^7+a^5+a)\cdot x^{128}+(a^6+a^5+a^4+a)\cdot x^{112}+(a^7+a^5+a^4+a^2+a)\cdot x^{96}+(a^7+a^5+a^4+a^3+a^2+1)\cdot x^{64}+(a^5+a^4)\cdot x^{56}+(a^7+a^6+a^3+a^2+a)\cdot x^{48}+(a^6+a^3+a^2+a)\cdot x^{32}+(a^6+a^3+1)\cdot x^{28}+(a^5+a)\cdot x^{24}+(a^7+a^5+a^4+a^3+a+1)\cdot x^{16}+(a^7+a^6+a^5+a^4+a+1)\cdot x^{14}+(a^5+a^4+a^3)\cdot x^{12}+(a^7+a^4+a+1)\cdot x^8+(a^3+1)\cdot x^7+(a^7+a^4+a+1)\cdot x^6+(a^6+a^4+a^3)\cdot x^4+(a^6+a^5+a^4+a^3+a^2+a+1)\cdot x^3+(a^7+a^3+a)\cdot x^2+(a^6+a^4+a^3+a^2+a)\cdot x+(a^7+a^3+a^2+a+1)$ |

**Table 4.** A polynomial $q_1$ that could be used in common for all the DES S-boxes. The irreducible polynomial used to represent $\mathbb{F}_{2^8}$ is $a^8 + a^4 + a^3 + a + 1$.

The experiments were performed on a DELL LATITUDE E55450 laptop (with CORE i3 processor and 64-bit architecture) running CentOS 7 in a virtual machine with 4 GB allotted memory. For efficiency, we have tabulated the computation of all the linear polynomials that appear in the evaluation of DES S-boxes (cf. [CRV15, Remark 3]). Note that these polynomials need to be stored only in the ROM. In Table 5, the parameter $t'$ refers to the order of security in the full security model of [ISW03], and $n' = 2t' + 1$ is the number of shares. The RAM memory usage (in bytes) that is reported is only for the S-box computations and the total CPU time for a DES encryption is measured in milliseconds. The penalty factor (PF) is the ratio of the total execution time for a given method to that of an unprotected implementation. The total number of calls made to the PRG that outputs random bytes is 1000 times the reported quantity.

## 3 The Power of Using Bigger Fields and its Limitations

In this section we ignore the higher cost of field operations when using a bigger field, so that we can gain some understanding of what can and what cannot be achieved by working with bigger fields.

| Method | $t'$ | $n'$ | Rand $\times 10^3$ | RAM Mem (bytes) | Time (ms) | PF |
|---|---|---|---|---|---|---|
| Unprotected | | | | | 0.005 | 1 |
| CGPQR+RV | 1 | 3 | 2752 | 72 | 0.290 | 58 |
| CGPQR+CRV | 1 | 3 | 1600 | 40 | 0.093 | 18 |
| **CGPQR+this work** | 1 | 3 | 1216 | 34 | 0.068 | 13 |
| CGPQR+RV | 2 | 5 | 9152 | 118 | 0.538 | 107 |
| CGPQR+CRV | 2 | 5 | 5312 | 64 | 0.175 | 35 |
| **CGPQR+this work** | 2 | 5 | 4032 | 54 | 0.133 | 26 |
| CGPQR+RV | 3 | 7 | 19200 | 164 | 0.824 | 164 |
| CGPQR+CRV | 3 | 7 | 11136 | 88 | 0.293 | 58 |
| **CGPQR+this work** | 3 | 7 | 8448 | 74 | 0.214 | 42 |
| CGPQR+RV | 4 | 9 | 32896 | 210 | 1.188 | 237 |
| CGPQR+CRV | 4 | 9 | 19072 | 112 | 0.455 | 91 |
| **CGPQR+this work** | 4 | 9 | 14464 | 94 | 0.323 | 64 |

**Table 5.** Comparison of secure masked implementations of DES.

As in our general cost model linear maps are for free, the domain $\mathbb{F}_2^d$ of our $d$-to-$r$-bit S-box table can be chosen to be any fixed $d$-dimensional subspace of the field $\mathbb{F}_{2^n}$ seen as a vector space over $\mathbb{F}_2$. When passing from using the field $\mathbb{F}_{2^n}$ for the non-linear multiplications to some extension field $\mathbb{F}_{2^{n'}}$ containing $\mathbb{F}_{2^n}$, one can therefore assume that the table is defined on a subspace $\mathbb{F}_2^d$ of $\mathbb{F}_{2^n}$, and use the same sequence of non-linear multiplications as for $\mathbb{F}_{2^n}$, but now viewed as products of polynomials over $\mathbb{F}_{2^{n'}}$ instead. So switching to an extension field never increases the number of non-linear multiplications.

Any two finite fields $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^d}$ of characteristic 2 are contained in some bigger field with $\mathbb{F}_{2^{lcm(n,d)}}$ being the minimal one. Hence we can assume in this section that *$d$ divides $n$* and that the table is defined on the *subfield* $\mathbb{F}_{2^d}$ of $\mathbb{F}_{2^n}$.

Since, for a polynomial $f(X) = \sum_{0 \le i \le \deg f} f_i X^i \in \mathbb{F}_{2^n}[X]$, we are only interested in the values on a subspace $\mathbb{F}_2^d \le \mathbb{F}_{2^n}$, we can reduce it modulo $p(X) := \prod_{z \in \mathbb{F}_2^d} (X - z)$ without changing these. So we can work with polynomials of degree $< 2^d$ instead of $2^n$ as is the case when the table is defined on all of $\mathbb{F}_{2^n}$. However, in general, the polynomial $p$ does not have a nice structure. But if $\mathbb{F}_2^d = \mathbb{F}_{2^d}$ is the unique subfield of order $2^d$ of $\mathbb{F}_{2^n}$, then $p(X) = X^{2^d} + X$ and the equation $x^{2^d} = x$ for all elements $x \in \mathbb{F}_{2^d} (\le \mathbb{F}_{2^n})$ implies

$$f(x) = \sum_{0 \le i \le \deg f} f_i x^i = f_0 + \sum_{0 < j < 2^d} \left( \sum_{i = j \bmod 2^d - 1} f_i \right) x^j .$$

Working over a bigger field than in the original CRV method has two benefits. The cyclotomic classes over $\mathbb{F}_{2^n}$ have sizes up to $n$, and hence more elements than the possible $d$ over $\mathbb{F}_2^d$, so that one gathers more degrees of freedom per non-linear multiplication in Step 1.[1] Additionally some extra power is given by being able to choose the coefficients of the polynomials in Step 2 from a bigger field:

**Lemma 1.** *Given $2k$ polynomials $f_i, g_i \in \mathbb{F}_{2^n}[X]$ $(0 \leq i < k)$ there exists an extension field $\mathbb{F}_{2^{n'}}$ of $\mathbb{F}_{2^n}$ and elements $a_i, b_i \in \mathbb{F}_{2^{n'}}$ such that for every $i$ the function $x \mapsto f_i(x) \cdot g_i(x)$ defined on $\mathbb{F}_{2^n}$ is an $\mathbb{F}_2$-linear image of the single non-linear product $h := (\sum_i a_i \cdot f_i(X)) \cdot (\sum_i b_i \cdot g_i(X)) \in \mathbb{F}_{2^{n'}}[X]$, i.e. there are $\mathbb{F}_2$-linear functions $\lambda_i : \mathbb{F}_{2^{n'}} \to \mathbb{F}_{2^n}$ with $\lambda_i \circ h(x) = f_i(x) \cdot g_i(x)$ for all $x \in \mathbb{F}_{2^n}$.*

*In particular, any finite number of* independent *non-linear multiplications over any finite field can be replaced by a* single *non-linear multiplication over a bigger field when restricting the maps to the smaller field.*

*Proof.* Take a prime $q > k^2$ not dividing $n$, and set $n' = q \cdot n$. For any element $z \in \mathbb{F}_{2^{n'}} \setminus \mathbb{F}_{2^n}$ the set $\{1, z, z^2, \ldots, z^{k^2-1}\}$ is linearly independent over $\mathbb{F}_{2^n}$ (otherwise it would span a proper intermediate field between $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^{n'}}$, but this extension has prime degree). Hence there exist $\mathbb{F}_{2^n}$-linear and therefore $\mathbb{F}_2$-linear maps $\lambda_i : \mathbb{F}_{2^{n'}} \to \mathbb{F}_{2^n}$ with $\lambda_i(\sum_{0 \leq j < k^2} c_j z^j) = c_{i+ki}$ when all $c_j \in \mathbb{F}_{2^n}$. For $a_i := z^i$ and $b_i := z^{ki}$ we get

$$\left(\sum_{0 \leq i < k} a_i \cdot f_i(X)\right) \cdot \left(\sum_{0 \leq j < k} b_j \cdot g_j(X)\right) = \sum_{0 \leq i,j < k} z^{i+kj}(f_i(X) \cdot g_j(X)).$$

Since, for $x \in \mathbb{F}_{2^n}$, we have also $f_i(x)g_j(x) \in \mathbb{F}_{2^n}$, the claim is proved.

*Remark 6.* The technique in the proof of Lemma 1 can be used to evaluate the non-linear part of the S-box of AES given by the monomial $X^{254}$ (over $\mathbb{F}_{2^8}$) with 3 non-linear multiplications over $\mathbb{F}_{2^{16}}$. The first non-linear multiplication is spent to get $X^3$, the second to multiply $X^2 + z \cdot X^3$ by $(X^3)^4$, where $z$ is any element of $\mathbb{F}_{2^{16}} \setminus \mathbb{F}_{2^8}$. From the result $X^{14} + z \cdot X^{15}$, one can $\mathbb{F}_2$-linearly extract the functions $x \mapsto x^{14}$ and $x \mapsto x^{15}$ defined over the subfield $\mathbb{F}_{2^8}$, which enables one to finally obtain $X^{254} = X^{14} \cdot (X^{15})^{16}$.

*Corollary 1.* With $l$ non-linear multiplications, *all* monomial functions $x \mapsto x^k$ defined on $\mathbb{F}_{2^d}$ with Hamming weight $k \leq 2^l$ can be obtained in parallel. In particular, for some huge extension field $\mathbb{F}_{2^n}$ of $\mathbb{F}_{2^d}$ *all* functions $\mathbb{F}_{2^d} \to \mathbb{F}_{2^n}$, including $d$-to-$r$-bit S-boxes, require just $\lceil \log_2 d \rceil$ non-linear multiplications.

The bound given in Corollary 1 is sharp: as the linear functions have *algebraic degree*[2] 1 and the algebraic degree of a product is at most the sum of the algebraic degrees of its factors, the function $f : \mathbb{F}_{2^d} \to \mathbb{F}_{2^n}$ given by monomial $X^{2^d-1}$ that maps 0 to 0 and the rest of $\mathbb{F}_{2^d}$ to 1 has algebraic degree $d$.

[1] Lemma 2 generalizes this statement about monomials to polynomials.
[2] For a polynomial $f = \sum_l f_l \cdot X^l$ this is $\max\{\text{Hamming weight(l)} \mid f_l \neq 0\}$.

For judging the usefulness of the result of a specific non-linear multiplication we have (denoting the space of functions from $Z$ to $Y$ as $Y^Z$ for sets $Y$ and $Z$):[3]

**Lemma 2.** *For $f : Z \to \mathbb{F}_{2^n}$ the set $F := \{g \circ f \mid g : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is $\mathbb{F}_2$-linear$\}$ is an $\mathbb{F}_{2^n}$-subspace of $\mathbb{F}_{2^n}^Z$ whose dimension over $\mathbb{F}_{2^n}$ equals the dimension over $\mathbb{F}_2$ of the $\mathbb{F}_2$-subspace of $\mathbb{F}_{2^n}$ generated by the image of $f$.*

*Proof.* $F$ is the image of the $\mathbb{F}_{2^n}$–linear map $\varphi : g \mapsto g \circ f$ from the set $\mathrm{End}_{\mathbb{F}_2}(\mathbb{F}_{2^n})$ of $\mathbb{F}_2$-linear maps $\mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ to the set $\mathbb{F}_{2^n}^Z$ of functions $Z \to \mathbb{F}_{2^n}$. $\mathrm{End}_{\mathbb{F}_2}(\mathbb{F}_{2^n}) = \mathbb{F}_{2^n}^* \otimes_{\mathbb{F}_2} \mathbb{F}_{2^n}$ has dimension $n$ over $\mathbb{F}_{2^n}$ (with $^*$ denoting the dual $\mathbb{F}_2$-vector space). The kernel of $\varphi$ is the $\mathbb{F}_{2^n}$-subspace of $\mathbb{F}_2$-linear maps whose restriction to the image of $f$ in $\mathbb{F}_{2^n}$ is 0. This is the tensor product with $\mathbb{F}_{2^n}$ (over $\mathbb{F}_2$) of the annihilator ($\leq \mathbb{F}_{2^n}^*$) of the image $\{f(x) \mid x \in Z\}$ of $f$ in $\mathbb{F}_{2^n}$, proving the claim.

*Example 1.* For monomials $X^\alpha$ the dimension of the set $F$ from Lemma 2 is the cardinality of the cyclotomic class containing $\alpha$. For example, in the field $\mathbb{F}_{64}$ the cyclotomic classes of $9 = 1001_2$ and $21 = 10101_2$ have order 3 resp. 2, so the dimension of the corresponding $F$ over $\mathbb{F}_{64}$ is 3 resp. 2. On the other hand, the images under $f(x) = x^9$ resp. $g(x) = x^{21}$ of the multiplicative group $\mathbb{F}_{64}^\times \cong Z_{63}$ have order 7 resp. 3, and are therefore the multiplicative groups of the subfields $\mathbb{F}_8$ resp. $\mathbb{F}_4$. Their dimensions over $\mathbb{F}_2$ are 3 resp. 2 as claimed by the lemma.

A criterion for having enough degrees of freedom in Step 2 is given by:

**Lemma 3.** *Let $F$ be $\mathbb{F}_{2^n}$-subspace of $\mathbb{F}_{2^n}[X]/(X^{2^n} + X)$ that is closed under taking squares. Then the $\mathbb{F}_{2^n}$-subspace $\langle F \cdot F \rangle_{\mathbb{F}_{2^n}}$ generated by the products of pairs of elements of $F$ contains $F$, is also closed under taking squares and has dimension at most $\dim F + \binom{\dim F}{2}$.*

*Proof.* As squaring is a field automorphism, only the statement about the dimensions needs to be proved. But this follows from the commutativity of multiplication as for any base $(f_i)$ of $F$ the set $(f_i \cdot f_j)_{i \leq j}$ generates $\langle F \cdot F \rangle_{\mathbb{F}_{2^n}}$.

The remainder of this section is devoted to proving a lower worst-case bound for the number of non-linear multiplications over $\mathbb{F}_{2^n}$ needed for functions from $\mathbb{F}_2^d$ to $\mathbb{F}_2^r$ with $d, r \leq n$ but not necessarily $d|n$. The proof is an adaption of [CRV14, Proposition 3] to our situation with minor improvements.

**Proposition 1.** *For $d, r \leq n$ and fixed subspaces $\mathbb{F}_2^d, \mathbb{F}_2^r \leq \mathbb{F}_{2^n}$ there is a function $f : \mathbb{F}_2^d \to \mathbb{F}_2^r$ that cannot be represented by any polynomial in $\mathbb{F}_{2^n}[X]$ that requires less than $\frac{\sqrt{r(2^d-1-d)+(d+\frac{r-n}{2})^2}-(d+\frac{r-n}{2})}{n}$ non-linear multiplications for evaluation.*

*In case of $n = r = d$ this term simplifies to $\sqrt{\frac{2^n-1}{n}} - 1$.*

[3] Corresponding to choosing $L$ in Algorithm 1 as the union of cyclotomic classes that have as many elements as possible to get as many degrees of freedom as possible for the linear equation system being constructed.

*Proof.* Without loss of generality, we may look only at functions that map 0 to 0: the only monomial not fixing zero is 1, and on $\mathbb{F}_{2^n} \setminus \{0\}$ the monomial $X^{2^{n-1}}$ is constant 1. This allows us to work with linear functions where the authors of [CRV14] used affine functions instead. Starting with $z_0 = id|_{\mathbb{F}_2^d}$ one can get all $\mathbb{F}_2$-linear functions $\mathbb{F}_2^d \to \mathbb{F}_{2^n}$ without using any non-linear multiplication. Having obtained $z_0, \dots, z_j$ using exactly $j$ non-linear multiplications, one can choose $\mathbb{F}_2$-linear maps $\lambda_{0,j}, \lambda'_{0,j} : \mathbb{F}_2^d \to \mathbb{F}_{2^n}$ and $\lambda_{1,j}, \lambda'_{1,j}, \dots, \lambda_{j,j}, \lambda'_{j,j} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ to get

$$z_{j+1} = \left( \sum_{i=0}^{j} \lambda_{i,j} \circ z_i \right) \cdot \left( \sum_{i=0}^{j} \lambda'_{i,j} \circ z_i \right).[4]$$

With the help of $z_0, \dots, z_k$ we then can evaluate

$$f = \sum_{0 \le i \le k} \lambda_i \circ z_i$$

for $\mathbb{F}_2$-linear maps $\lambda_0 : \mathbb{F}_2^d \to \mathbb{F}_2^r$ and $\lambda_1, \dots, \lambda_k : \mathbb{F}_{2^n} \to \mathbb{F}_2^r$ without further non-linear multiplication. Conversely, any $f : \mathbb{F}_2^d \to \mathbb{F}_2^r$ fixing 0 that can be evaluated using at most $k$ non-linear multiplications is of this form.

In total we have to choose $2k$ $\mathbb{F}_2$-linear maps from $\mathbb{F}_2^d$ to $\mathbb{F}_{2^n}$, $2\sum_{i=0}^{k-1} i = k(k-1)$ from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$, one from $\mathbb{F}_2^d$ to $\mathbb{F}_2^r$ and $k$ from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2^r$ giving us $((2^n)^d)^{2k} \cdot ((2^n)^n)^{k(k-1)} \cdot (2^r)^d \cdot ((2^r)^n)^k = 2^{2ndk+n^2k(k-1)+rd+rnk}$ choices. As there are $(2^r)^{2^d-1} = 2^{r(2^d-1)}$ functions from $\mathbb{F}_2^d$ to $\mathbb{F}_2^r$ mapping 0 to 0, to get enough functions we need

$$2ndk + n^2k(k-1) + rd + rnk \ge r(2^d - 1).$$

This is via $(nk)^2 + (2d + r - n)nk \ge r(2^d - 1) - rd$ and $(nk + (d + \frac{r-n}{2}))^2 = (nk)^2 + (2d + r - n)nk + (d + \frac{r-n}{2})^2 \ge r(2^d - 1 - d) + (d + \frac{r-n}{2})^2$ equivalent to

$$k \ge \frac{\sqrt{r(2^d - 1 - d) + (d + \frac{r-n}{2})^2} - (d + \frac{r-n}{2})}{n}.$$

*Remark 7.* As the images of the $z_j$s in the proof of Proposition 1 can span at most a $(2^d - 1)$-dimensional $\mathbb{F}_2$-subspace of $\mathbb{F}_{2^n}$, Lemma 2 shows that for $n \ge 2^d - 1$ the $\lambda_{i,j}$, $\lambda'_{i,j}$ and $\lambda_i$ with $i > 0$ have to be defined only on these $(2^d - 1)$-dimensional subspaces reducing the degrees of freedom for obtaining the next $z_j$ resp. $f$. With $n' := \max\{n, 2^d - 1\}$ the number of choices reduces to $2^{2ndk+nn'k(k-1)+rd+rn'k}$, but as one gets better lower bounds by using the algebraic degree, we do not expand upon this.

**Acknowledgements.** We would like to thank Jean-Sébastien Coron and Johann Großschädl for pointing out possible applications of our improvements to ARM

[4] Adding a constant to either factor changes $z_{j+1}$ by a summand that can be represented already by the $z_i$ with $i \le j$.

Neon processors and also for pointing out [GMPT15]. Thanks to Jake Longo Galea for clarifying the work of [GMPT15]. We are also thankful to Carolyn Whitnall, Srinivas Karthik and the anonymous reviewers of CHES 2016 for many helpful comments.

Srinivas Vivek's work has been supported in part by the European Union's H2020 programme under grant agreement number ICT-644209.

## References

ARS$^+$15. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Oswald and Fischlin [OF15], pages 430–454.

Bar86. Paul Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In Andrew M. Odlyzko, editor, *CRYPTO 1986, Proc.*, volume 263 of *LNCS*, pages 311–323. Springer, 1986.

BFG15. Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner product masking revisited. In Oswald and Fischlin [OF15], pages 486–510.

BFGV12. Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and practice of a leakage resilient masking scheme. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012, Proc.*, volume 7658 of *LNCS*, pages 758–775. Springer, 2012.

CGP$^+$12. Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for S-Boxes. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 366–384. Springer, 2012.

CJRR99. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [Wie99], pages 398–412.

Cor13. Jean-Sébastien Coron, 2013. Available at `https://github.com/coron/htable/`.

Cor14. Jean-Sébastien Coron. Higher order masking of look-up tables. In Nguyen and Oswald [NO14], pages 441–458.

CPRR13. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 410–424. Springer, 2013.

CPRR15. Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Proc., Part I*, volume 9215 of *LNCS*, pages 742–763. Springer, 2015.

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, *CHES 2014. Proc.*, volume 8731 of *LNCS*, pages 170–187. Springer, 2014.

CRV15. Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. *J. Cryptographic Engineering*, 5(2):73–83, 2015.

DDF14. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Nguyen and Oswald [NO14], pages 423–440.

DFS15. Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Proc., Part II*, volume 9057 of *LNCS*, pages 159–188. Springer, 2015.

GHS12a. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. *IACR Cryptology ePrint Archive*, 2012:99, 2012.

GHS12b. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012. Proc.*, volume 7417 of *LNCS*, pages 850–867. Springer, 2012.

GM11. Louis Goubin and Ange Martinelli. Protecting AES with Shamir's secret sharing scheme. In Preneel and Takagi [PT11], pages 79–94.

GMPT15. Jake Longo Galea, Elke De Mulder, Dan Page, and Michael Tunstall. SoC it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015, Proc.*, volume 9293 of *LNCS*, pages 620–640. Springer, 2015.

GPS14. Vincent Grosso, Emmanuel Prouff, and François-Xavier Standaert. Efficient masked S-boxes processing - A step forward -. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 2014. Proc.*, volume 8469 of *LNCS*, pages 251–266. Springer, 2014.

ISW03. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

KJJ99. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [Wie99], pages 388–397.

Koc96. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO 1996, Proc.*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

Lim13. ARM Limited. NEON Programmer's Guide, 2013.

NO14. Phong Q. Nguyen and Elisabeth Oswald, editors. *EUROCRYPT 2014. Proc.*, volume 8441 of *LNCS*. Springer, 2014.

OF15. Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Proc., Part I*, volume 9056 of *LNCS*. Springer, 2015.

oST93. National Institute of Standards and Technology. FIPS 46-3: Data Encryption Standard, March 1993. Available via csrc.nist.gov.

PR11. Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In Preneel and Takagi [PT11], pages 63–78.

PR13. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013. Proc.*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.

PT11. Bart Preneel and Tsuyoshi Takagi, editors. *CHES 2011. Proc.*, volume 6917 of *LNCS*. Springer, 2011.

RP10. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010. Proc.*, volume 6225 of *LNCS*, pages 413–427. Springer, 2010.

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, *CHES 2013. Proc.*, volume 8086 of *LNCS*, pages 417–434. Springer, 2013.

SP06. Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 208–225. Springer, 2006.

Wie99. Michael J. Wiener, editor. *CRYPTO 1999, Proc.*, volume 1666 of *LNCS*. Springer, 1999.

WVGX15. Junwei Wang, Praveen Kumar Vadnala, Johann Großschädl, and Qiuliang Xu. Higher-order masking in practice: A vector implementation of masked AES for ARM NEON. In Kaisa Nyberg, editor, *CT-RSA 2015. Proc.*, volume 9048 of *LNCS*, pages 181–198. Springer, 2015.