Fast Evaluation of Polynomials over Binary Finite Fields and Application to Side-channel Countermeasures*

Jean-Sébastien Coron[1], Arnab Roy[1,2], Srinivas Vivek[1]

[1] University of Luxembourg
{jean-sebastien.coron, srinivasvivek.venkatesh}@uni.lu
[2] Technical University of Denmark
arroy@dtu.dk

Abstract. We describe a new technique for evaluating polynomials over binary finite fields. This is useful in the context of anti-DPA countermeasures when an S-box is expressed as a polynomial over a binary finite field. For n-bit S-boxes our new technique has heuristic complexity $\mathcal{O}(2^{n/2}/\sqrt{n})$ instead of $\mathcal{O}(2^{n/2})$ proven complexity for the Parity-Split method. We also prove a lower bound of $\Omega(2^{n/2}/\sqrt{n})$ on the complexity of any method to evaluate n-bit S-boxes; this shows that our method is asymptotically optimal. Here, complexity refers to the number of non-linear multiplications required to evaluate the polynomial corresponding to an S-box.

In practice we can evaluate any 8-bit S-box in 10 non-linear multiplications instead of 16 in the Roy-Vivek paper from CHES 2013, and the DES S-boxes in 4 non-linear multiplications instead of 7. We also evaluate any 4-bit S-box in 2 non-linear multiplications instead of 3. Hence our method achieves optimal complexity for the PRESENT S-box.

1 Introduction

The implementations of cryptographic algorithms on devices like PCs, microcontrollers, smart cards, etc. leak secret information to an adversary. Typical examples of such leakages are electro-magnetic emissions, power consumption and even acoustic emanations. An adversary can use this information to recover the secret key by applying different statistical techniques. Differential Power Analysis (DPA) – the most widely known and powerful technique – is based on statistical analysis of the power consumption of a device [KJJ99]. Other techniques including Template Attacks, Correlation Power Analysis Attacks (CPA), etc. were proposed in the past [CRR02,BCO04]. More recently, a side-channel attack on RSA was proposed using the acoustic emanations from a device [GST13].

* This paper has been published at CHES 2014. The final publication is available at www.springerlink.com. This is the full version.

Masking. A well known technique to protect implementations against power analysis based side-channel attacks is to mask internal *secret* variables. This is done by XORing any internal variable with a random variable r, for e.g., $x' = x \oplus r$. However, this will make the implementation secure against first-order attacks only. Second-order attacks against such counter-measures is proposed in [Mes00]. In this type of attack the adversary combines the information obtained from two internal variables. This will require more data (power consumption traces) in practice, which could make the attack infeasible in certain cases. In general the above masking technique can be extended to secure an implementation against higher-order attacks. This can be achieved by splitting an internal variable x into d shares, say, $x = \bigoplus_{i=1}^{d} x_i$. Using this idea it is easy to compute any linear/affine function ℓ in a secured way, since it is enough to compute $y_i = \ell(x_i)$ for $1 \leq i \leq d$. However, it is not obvious how to do this for non-linear functions. In practice, nearly every cryptographic primitive includes some non-linear function, e.g., S-box, modular addition, etc.

Generic Higher-Order Masking. The Rivain-Prouff masking scheme is the first provably secure higher-order masking technique for AES [RP10]. The main idea of this method is to perform secure monomial evaluation with d shares of a secret variable using the previously known ISW scheme [ISW03]. Namely the (non-linear part of) AES S-box can be represented by the monomial x^{254} over $\mathbb{F}_{2^8}$. Prouff and Rivain showed that this monomial can be evaluated securely using 4 non-linear multiplications and a few linear squarings. By using this scheme the AES S-box can be masked for any order d.

This method was extended to a generic technique for higher-order masking, in [CGP+12], by Carlet, Goubin, Prouff, Quisquater and Rivain (CGPQR). Any given n-bit S-box can be represented by a polynomial $\sum_{i=0}^{2^n-1} a_i\, x^i$ over $\mathbb{F}_{2^n}$ using Lagrange's interpolation theorem. Hence, any S-box can be masked by secure evaluation of this polynomial with d shares of a secret variable. This is the first generic technique to mask any S-box for any order d. In this technique a polynomial evaluation in $\mathbb{F}_{2^n}$ is split into simple operations over $\mathbb{F}_{2^n}$: addition, multiplication by constant, and regular multiplication of two elements. Note that multiplication of two same elements (i.e. squaring) and multiplication by a constant – both are linear operations over $\mathbb{F}_{2^n}$, hence easy to mask. For performing a secure multiplication of two distinct elements, *i.e.* a non-linear multiplication, the CGPQR masking scheme uses the ISW method as in [RP10].

Asymptotically, the running time of the Rivain-Prouff and CGPQR masking schemes is dominated by the number of non-linear multiplications required to evaluate a polynomial over $\mathbb{F}_{2^n}$. Namely with d shares, using the ISW method an affine function can be masked with only $\mathcal{O}(d)$ operations over $\mathbb{F}_{2^n}$, whereas a non-linear multiplication requires $\mathcal{O}(d^2)$ operations. Note that for achieving d-th order security the Rivain-Prouff scheme requires at least $2d + 1$ shares.[1]

[1] Originally it was claimed in [RP10] that the scheme is secure against d-th order attack with $d + 1$ shares. However, an attack of order $d/2$ was shown in [CPRR13]

Efficient Polynomial Evaluation for Masking. The CGPQR masking scheme can be made efficient by optimizing the number of multiplications required for the polynomial evaluation in $\mathbb{F}_{2^n}$. In [CGP+12] two techniques – *Parity-Split* and *Cyclotomic Class*, are used for optimizing the number of such non-linear multiplications. For arbitrary n-bit S-box, or equivalently for evaluating any polynomial over $\mathbb{F}_{2^n}$, the Parity-Split method has proven complexity $\mathcal{O}(2^{n/2})$. Here complexity refers to the number of non-linear multiplications required to evaluate the polynomial corresponding to an S-box. For the particular case of monomials (e.g. AES S-box) the Cyclotomic Class method gives the optimal number of multiplications in $\mathbb{F}_{2^n}$.

At CHES 2013, Roy and Vivek [RV13] adapted a generic method for improving the efficiency of polynomial evaluation in $\mathbb{F}_{2^n}$. They demonstrated the technique for the polynomials corresponding to several well known S-boxes including DES, PRESENT and CLEFIA. In particular, the Roy-Vivek method reduces the number of non-linear multiplications for DES to 7 (from 10), for CLEFIA to 16 (from 22) and for CAMELLIA to 15 (from 22). This technique also achieves the optimal number of 4 multiplications for the monomial corresponding to the AES S-box.

Our Results. In this article we propose an improved generic technique for fast polynomial evaluation in $\mathbb{F}_{2^n}$. For arbitrary n-bit S-box our method has heuristic complexity $\mathcal{O}(2^{n/2}/\sqrt{n})$, compared to the $\mathcal{O}(2^{n/2})$ proven complexity for the Parity-Split method from [CGP+12].

Our method is as follows. We first generate a set L of monomials x^α, including all the monomials from a cyclotomic class. We then randomly generate a fixed set of "basis" polynomials $q_i(x)$, whose monomials are all in the precomputed set L. Then given a polynomial $P(x)$ over $\mathbb{F}_{2^n}$ we try to write $P(x)$ as:

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

where $p_i(x)$ are polynomials with monomials also in the set L, and t is some parameter. Since the $q_i(x)$ polynomials are fixed, the coefficients of the $p_i(x)$ polynomials can be obtained by solving a system of linear equations in $\mathbb{F}_{2^n}$. Then to evaluate $P(x)$ one first evaluates all the monomials in the set L; the polynomials $p_i(x)$ and $q_i(x)$ can then be evaluated without any further non-linear multiplication. The polynomials $P(x)$ is then evaluated from (1) with $t-1$ additional non-linear multiplications.

The number of monomials in the set L must be carefully chosen. Namely the larger the basis set L of monomials, the more degrees of freedom we have in solving (1), with fewer polynomials $p_i(x)$ and therefore fewer additional non-linear multiplications; however the number of non-linear multiplications to build L will increase. Therefore the number of monomials in the basis set L must be

against the scheme. The authors of [CPRR13] also showed a d-th order secure scheme with $d+1$ shares for some subset of S-boxes.

optimized to minimize the total number of non-linear multiplications, namely the non-linear multiplications for building the set L, and the additional $t-1$ non-linear multiplications for evaluating $P(x)$.

As a concrete application of our new method above, we show that for the generic higher-order masking of several well known S-boxes, e.g. DES, CLEFIA, PRESENT, etc., our method reduces the number of multiplications compared to the previously known methods [CGP$^+$12,RV13]. In particular, using our method PRESENT can be masked with 2 multiplications (instead of 3), and DES with 4 multiplications (instead of 7), see Table 1. Our method achieves optimal complexity for the PRESENT S-box since it was proved in [RV13] that 2 non-linear multiplications are necessary to evaluate it. In Table 5, we report the timing results for DES masked using our technique.

	S-box				
Methods	DES	PRESENT	SERPENT	CAMELLIA	CLEFIA
Parity-Split [CGP$^+$12]	10	3	3	22	22
Roy-Vivek [RV13]	7	3	3	15	15,16
Our Method (Sec. 4)	**4**	**2**	**2**	**10**	**10**

Table 1. Number of non-linear multiplications required for the CGPQR generic higher-order masking scheme.

We also prove a lower bound of $\Omega(2^{n/2}/\sqrt{n})$ for the complexity of any method to evaluate n-bit S-boxes, a.k.a. *masking complexity*; this shows that our method is asymptotically optimal. Our new lower bound significantly improves upon the previously known bound of $\Omega(\log_2 n)$ from [RV13].

2 Generic Polynomial Evaluation Technique

Before we describe our improved method to evaluate polynomials over $\mathbb{F}_{2^n}$, let us first recollect in Section 2.1 the method proposed by Roy and Vivek [RV13, Section 4] to evaluate the polynomials (over $\mathbb{F}_{2^6}$) corresponding to the DES S-boxes. Their method requires 7 non-linear multiplications. The method in [RV13] is based on the *Divide-and-Conquer* strategy, which is an adaptation of a polynomial evaluation technique by Paterson and Stockmeyer [PS73]. The technique consists in decomposing the polynomial to be evaluated in terms of polynomials having their monomials from a precomputed set. Our method is partly based on this idea.

2.1 The Roy-Vivek Method for DES S-boxes

Let $P_{DES}(x) \in \mathbb{F}_{2^6}[x]$ be the Lagrange interpolation polynomial corresponding to a DES S-box. Here the 4-bit output of a DES S-box is identified as a 6-bit output with two leading zeroes, and hence these bit strings are naturally identified

with the elements of $\mathbb{F}_{2^6}$. Note that for all the DES S-boxes, $\deg(P_{DES}(x)) = 62$. Write

$$P_{DES}(x) = q(x) \cdot x^{36} + R(x),$$

where $\deg(R) \leq 35$ and $\deg(q) = 26$. Then divide the polynomial $R(x) - x^{27}$ by $q(x)$:

$$R(x) - x^{27} = c(x) \cdot q(x) + s(x),$$

where $\deg(c) \leq 9$ and $\deg(s) \leq 25$, which gives

$$P_{DES}(x) = \big(x^{36} + c(x)\big) \cdot q(x) + x^{27} + s(x).$$

Next decompose the polynomials $q(x)$ and $x^{27} + s(x)$ in a similar way but, instead, dividing first by x^{18}, and then using x^9 as the "correction term". One gets

$$\begin{aligned} q(x) &= (x^{18} + c_1(x)) \cdot q_1(x) + x^9 + s_1(x), \\ x^{27} + s(x) &= (x^{18} + c_2(x)) \cdot q_2(x) + x^9 + s_2(x) \end{aligned}$$

where $\deg(q_1) = 8$, $\deg(c_1) \leq 9$, $\deg(s_1) \leq 7$, $\deg(q_2) = 9$, $\deg(c_2) \leq 8$, and $\deg(s_2) \leq 8$. Finally,

$$\begin{aligned} P_{DES}(x) =& (x^{36} + c(x)) \cdot \Big(((x^{18} + c_1(x)) \cdot q_1(x)) + (x^9 + s_1(x))\Big) \\ &+ \Big((x^{18} + c_2(x)) \cdot q_2(x) + (x^9 + s_2(x))\Big). \end{aligned} \tag{2}$$

In [RV13], the monomials x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{18}, x^{36} are first evaluated using 4 non-linear multiplications. Namely a non-linear multiplication is required for each of the monomials x^3, x^5, x^7 and x^9; the rest of the monomials can be evaluated using linear squarings only. Each of the individual polynomials in the above expression such as $x^{36} + c(x)$, $x^{18} + c_1(x)$, $q_1(x)$, and so on, can then be evaluated for free, that is without further non-linear multiplications. To evaluate $P_{DES}(x)$ from (2), 3 more non-linear multiplications are needed, and hence totally 7 non-linear multiplications are sufficient to evaluate a DES S-box.

To sum up, the basic idea behind the above technique is to precompute a set of monomials, and then obtain a decomposition of the required polynomial in terms of polynomials having their monomials only from the precomputed set. Note that the said decomposition is obtained in a "fixed" way that depends only on the degree of the polynomial, which is required to be of the form $k\,(2^p - 1) \pm c$, for some parameters k, p and c; we refer to [RV13] for more details.

In the new method we propose next, we also precompute a set of monomials as above, but we also include every other monomial that can be computed for free by the squaring operation; that is we always generate the full cyclotomic class for any computed monomial. Then we try to decompose the polynomial as a sum of product of two polynomials having their monomials from the precomputed set. One of the two polynomials in every summand is randomly chosen, and we try to determine the other polynomial by solving (for unknown coefficients) the system of linear equations obtained by evaluating the polynomial at every point of the domain $\mathbb{F}_{2^n}$. This approach of determining the unknown coefficients of the polynomials is similar to the Lagrange interpolation technique.

2.2 Our New Generic Method

Let us first recollect the notion of cyclotomic class over $\mathbb{F}_{2^n}$ and introduce some notations. The cyclotomic class of α w.r.t. n ($n \geq 1$, $0 \leq \alpha \leq 2^n - 2$), denoted by C_α, is defined as the set of integers

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

Intuitively, C_α corresponds to the exponents of all the monomials that can be computed from $x^\alpha \in \mathbb{F}_{2^n}[x]$ using only the squaring operations (modulo $x^{2^n} + x$). Since our goal is only to evaluate polynomials over $\mathbb{F}_{2^n}$, we will be actually working in the ring $\mathbb{F}_{2^n}[x]/(x^{2^n} + x)$, which is an abuse of the notation $\mathbb{F}_{2^n}[x]$. In other words, we treat any polynomial $P(x) \in \mathbb{F}_{2^n}[x]$ to be the same as $P(x)$ modulo $x^{2^n} + x$; hence $P(x)$ has degree at most $2^n - 1$.

By $d \xleftarrow{\$} D$ we denote an element d chosen uniformly at random from a set D. For any subset $\Lambda \subseteq \{0, 1, \ldots, 2^n - 2\}$, x^Λ denotes the set of monomials $x^\Lambda = \left\{ x^i \ : \ i \in \Lambda \right\} \subseteq \mathbb{F}_{2^n}[x]$. Finally we denote by $\mathcal{P}(x^\Lambda)$ the set of all polynomials in $\mathbb{F}_{2^n}[x]$ whose monomials are only from the set x^Λ.

Description. Consider an n-bit to n-bit S-Box represented by a polynomial $P(x) \in \mathbb{F}_{2^n}[x]$. We consider a collection $\mathcal{S}$ of ℓ cyclotomic classes w.r.t. n:

$$\mathcal{S} = \{C_{\alpha_1=0}, C_{\alpha_2=1}, C_{\alpha_3}, \ldots, C_{\alpha_\ell}\}. \tag{3}$$

Also, define L as the set of all integers in the cyclotomic classes of $\mathcal{S}$:

$$L = \bigcup_{C_i \in \mathcal{S}} C_i. \tag{4}$$

We choose the set $\mathcal{S}$ of ℓ cyclotomic classes in (3) so that the set of corresponding monomials x^L from $\mathcal{S}$ can be computed using only $\ell - 2$ non-linear multiplications. We require that every monomial x^0, x^1, ... , x^{2^n-1}, can be written as product of some two monomials in $\mathcal{P}(x^L)$. Moreover, we try to choose only those cyclotomic classes with the maximum number of n elements (except C_0 which has only a single element). This gives

$$|L| = 1 + n \cdot (\ell - 1). \tag{5}$$

Next, we generate $t - 1$ random polynomials $q_i(x) \xleftarrow{\$} \mathcal{P}(x^L)$ that have their monomials only in x^L. Suitable values for the parameters t and $|L|$ will be determined later. Then, we try to find t polynomials $p_i(x) \in \mathcal{P}(x^L)$ such that

$$P(x) = \sum_{i=1}^{t-1} p_i(x) \cdot q_i(x) \ + \ p_t(x). \tag{6}$$

It is easy to see that the coefficients of the $p_i(x)$ polynomials can be obtained by solving a system of linear equations in $\mathbb{F}_{2^n}$, as in the Lagrange interpolation

theorem. More precisely, to find the polynomials $p_i(x)$, we solve the following system of linear equations over $\mathbb{F}_{2^n}$:

$$A \cdot \boldsymbol{c} = \boldsymbol{b} \tag{7}$$

where the matrix A is obtained by evaluating the R.H.S. of (6) at every element of $\mathbb{F}_{2^n}$, and by treating the unknown coefficients of $p_i(x)$ as variables. This matrix has 2^n rows and $t \cdot |L|$ columns, since each of the t polynomials $p_i(x)$ has $|L|$ unknown coefficients. The matrix A can also be written as a block concatenation of smaller matrices:

$$A = (A_1|A_2|\ldots|A_t), \tag{8}$$

where A_i is a $2^n \times |L|$ matrix corresponding to the product $p_i(x) \cdot q_i(x)$. Let $a_j \in \mathbb{F}_{2^n} \quad (j = 0, 1, \ldots, 2^n - 1)$ be all the field elements and $p_i(x)$ consists of the monomials $x^{k_1}, x^{k_2}, \ldots, x^{k_{|L|}} \in x^L$. Then, the matrix A_i has the following structure:

$$A_i = \begin{pmatrix} a_0^{k_1} \cdot q_i(a_0) & a_0^{k_2} \cdot q_i(a_0) & \ldots & a_0^{k_{|L|}} \cdot q_i(a_0) \\ a_1^{k_1} \cdot q_i(a_1) & a_1^{k_2} \cdot q_i(a_1) & \ldots & a_1^{k_{|L|}} \cdot q_i(a_1) \\ a_2^{k_1} \cdot q_i(a_2) & a_2^{k_2} \cdot q_i(a_2) & \ldots & a_2^{k_{|L|}} \cdot q_i(a_2) \\ \vdots & \vdots & \ldots & \vdots \\ a_{2^n-1}^{k_1} \cdot q_i(a_{2^n-1}) & a_{2^n-1}^{k_2} \cdot q_i(a_{2^n-1}) & \ldots & a_{2^n-1}^{k_{|L|}} \cdot q_i(a_{2^n-1}) \end{pmatrix} \tag{9}$$

The unknown vector $\boldsymbol{c}$ in (7) corresponds to the unknown coefficients of the polynomials $p_i(x)$. The vector $\boldsymbol{b}$ is formed by evaluating $P(x)$ at every element of $\mathbb{F}_{2^n}$. Note that since $P(x)$ corresponds to an S-box, the vector $\boldsymbol{b}$ can be directly obtained from the corresponding S-box lookup table.

If the matrix A has rank 2^n, then we are able to guarantee that the decomposition in (6) exists for every polynomial $P(x)$. To be of full rank 2^n the matrix must have a number of columns $\geq 2^n$. This gives us the necessary condition

$$t \cdot |L| \geq 2^n. \tag{10}$$

We stress that (10) is only a necessary condition. Namely we don't know how to prove that the matrix A will be full rank when the previous condition is satisfied; this makes our algorithm heuristic. In practice for random polynomials $q_i(x)$ we almost always obtain a full rank matrix under condition (10).

From (5), we get the condition

$$t \cdot (1 + n \cdot (\ell - 1)) \geq 2^n \tag{11}$$

where t is the number of polynomials $p_i(x)$ and ℓ the number of cyclotomic classes in the set $\mathcal{S}$, to evaluate a polynomial $P(x)$ over $\mathbb{F}_{2^n}$.

We summarize the above method in Algorithm 1 below. The number of non-linear multiplications required in the combining step (6) is $t - 1$. As mentioned

earlier, we need $\ell - 2$ non-linear multiplications to precompute the set x^L. Hence the total number of non-linear multiplications required is then

$$N_{mult} = \ell - 2 + t - 1 = \ell + t - 3. \tag{12}$$

where t is the number of polynomials $p_i(x)$ and ℓ the number of cyclotomic classes in the set $\mathcal{S}$.

Algorithm 1 New generic polynomial decomposition algorithm

Input: $P(x) \in \mathbb{F}_{2^n}[x]$.

Output: Polynomials $p_i(x), q_i(x)$ such that $P(x) = \sum_{i=1}^{t-1} p_i(x) \cdot q_i(x) + p_t(x)$.

1: Choose ℓ cyclotomic classes C_{α_i} : $L \leftarrow \bigcup_{i=1}^{l} C_{\alpha_i}$, and the basis set x^L can be computed using $\ell - 2$ non-linear multiplications.
2: Choose t such that $t \cdot |L| \geq 2^n$.
3: For $1 \leq i \leq t$, choose $q_i(x) \xleftarrow{\$} \mathcal{P}\left(x^L\right)$.
4: Construct the matrix $A \leftarrow (A_1|A_2|\ldots|A_t)$, where each A_i is the $2^n \times |L|$ matrix given by (9).
5: Solve the linear system $A \cdot \boldsymbol{c} = \boldsymbol{b}$, where $\boldsymbol{b}$ is the evaluation of $P(x)$ at every element of $\mathbb{F}_{2^n}$.
6: Construct the polynomials $p_i(x)$ from the solution vector $\boldsymbol{c}$.

Remark 1. If A has rank 2^n, then the same set of basis polynomials $q_i(x)$ will yield a decomposition as in (6) for *any* polynomial $P(x)$. That is, the matrix A is independent from the polynomial $P(x)$ to be evaluated.

Remark 2. Our decomposition method is heuristic because for a given n in $\mathbb{F}_{2^n}$ we do not know how to guarantee that the matrix A has full rank 2^n. However for typical values of n, say $n = 4, 6, 8$, we can definitely check that the matrix A has full rank, for a particular choice of random polynomials $q_i(x)$. Then any polynomial $P(x)$ can be decomposed using these polynomials $q_i(x)$. In other words for a given n we can once and for all generate the random polynomials $q_i(x)$ and check that the matrix A has full rank 2^n, which will prove that any polynomial $P(x) \in \mathbb{F}_{2^n}[x]$ can then be decomposed as above. In summary our method is heuristic for large values of n, but can be proven for small values of n. Such proof requires to compute the rank of a matrix with 2^n rows and a slightly larger number of columns, which takes $\mathcal{O}(2^{3n})$ time using Gaussian elimination.

Asymptotic Analysis. Substituting (12) in (11) to eliminate the parameter ℓ, we get

$$\begin{aligned} & t \cdot (1 + n \cdot (N_{mult} - t + 2)) \geq 2^n, \\ \implies \quad & N_{mult} \geq \frac{2^n}{n \cdot t} + t - \left(2 + \frac{1}{n}\right). \end{aligned} \tag{13}$$

The R.H.S. of the above expression is minimized when $t \approx \sqrt{\frac{2^n}{n}}$, and hence we obtain

$$N_{mult} \geq 2 \cdot \sqrt{\frac{2^n}{n}} - \left(2 + \frac{1}{n}\right). \tag{14}$$

Hence, our heuristic method requires $\mathcal{O}(\sqrt{2^n/n})$ non-linear multiplications, which is asymptotically slightly better than the Parity-Split method [CGP$^+$12], which has proven complexity $\mathcal{O}(\sqrt{2^n})$. If one has to rigorously establish the above bound for our method, then we may have to prove the following statements, which we leave as open problems:

- We can sample the collection S of cyclotomic classes in (3), each having maximal length n (other than C_0), using at most $\ell - 2$ non-linear multiplications.
- The condition $t \cdot |L| \geq 2^n$ suffices to ensure that the matrix A has full rank 2^n.

Table 2 lists the expected minimum number of non-linear multiplications, as determined by (14), for binary fields $\mathbb{F}_{2^n}$ of practical interest. It also lists the actual number of non-linear multiplications that suffices to evaluate any polynomial, for which we have verified that the matrix A has full rank 2^n, for a particular random choice of the $q_i(x)$ polynomials. We also provide a performance comparison of our method with that of the Cyclotomic Class and the Parity-Split methods from [CGP$^+$12]. Here we do not compare with the results from [RV13] since that work is mainly concerned with the optimization of specific S-boxes and polynomials of specific degrees; however such comparison will be made for specific S-boxes in Section 4. In Appendix B, we list the specific choice of parameters t and L that we used in this experiment.

n	4	5	6	7	8	9	10
Cyclotomic Class method [CGP$^+$12]	3	5	11	17	33	53	105
Parity-Split method [CGP$^+$12]	4	6	10	14	22	30	46
Expected minimum value of N_{mult} (cf. (14))	2	3	5	7	10	13	19
Achievable value of N_{mult}	**2**	**4**	**5**	**7**	**10**	**14**	**19**

Table 2. Minimum values of N_{mult}

Counting the Linear Operations. From (5) and (6), we get $(2t-1) \cdot (|L| - 1) + (t-1)$ as an upper-bound on the number of addition operations required to evaluate $P(x)$. This is because each of the $2t-1$ polynomials $p_i(x)$ and $q_i(x)$ in (6) have (at most) $|L|$ terms, and there are t summands in (6). From (10), we get:

$$(2t-1) \cdot (|L| - 1) + (t-1) \leq 2\,t\,|L| \approx 2 \cdot 2^n$$

Similarly, we get $(2t-1)\cdot|L| \approx 2\cdot 2^n$ as an estimate for the number of scalar multiplications. Since the squaring operations are used only to compute the list L, we need $|L|-\ell \leq |L| \approx \sqrt{n\cdot 2^n}$ many of them (cf. (13)).

Remark 3. The above count of the linear operations can be significantly reduced if the linear operations are replaced by table lookups as much as possible. Such an approach is particularly well suited for application in the higher-order masking scheme of [CGP+12], where we need to evaluate a given polynomial with many shares and that the processing of linear polynomials with shares is particularly straightforward. More specifically, we can write each $p_i(x)$ as a sum of $\mathbb{F}_2$-linear polynomials $p_{i,j}$, one for each cyclotomic class in the pre-computed set $\mathcal{S}$ (cf. (3), (4)):

$$p_i(x) = \sum_{C_{\alpha_j}\in\mathcal{S}} p_{i,j}\left(x^{\alpha_j}\right).$$

The ℓ polynomials $p_{i,j}$ are $\mathbb{F}_2$-linear and hence are of the form $\sum_{k=0}^{n-1}\gamma_k\, x^{2^k}$. Similarly, the polynomials $q_i(x)$ can also be expressed in the above form. If we tabulate the values of each of the linear polynomials $p_{i,j}$ and $q_{i,j}$, then it suffices to evaluate x^{α_j} for each cyclotomic class $C_{\alpha_j}\in\mathcal{S}$ using only NLMs. Then the polynomials $p_{i,j}$ and $q_{i,j}$ can be evaluated by just table lookups, and then each of the $2t-1$ polynomials p_i and q_i can be eventually evaluated with $\ell-1$ additions each. Finally, we need $t-1$ more additions in the step (6). Hence, we need no scalar multiplications nor squarings using this table lookup technique. The total number of additions we need is

$$(2t-1)\cdot(\ell-1)+t-1 \approx \frac{2\cdot 2^n}{n}.$$

Note that this technique is not very effective for the evaluation method of [RV13] since nearly every linear polynomial that appears has at most two non-zero terms.

3 New Lower Bound for Polynomial Evaluation

In this section, we show that our method from the previous section is asymptotically optimal. More precisely, we show that to evaluate any polynomial over $\mathbb{F}_{2^n}$, any algorithm must use at least $\mathcal{O}(\sqrt{2^n/n})$ non-linear multiplications. This improves the previously known bound of $\Omega(\log_2 n)$ from [RV13].

To establish our lower bound we first need a formal model that describes polynomial evaluation over $\mathbb{F}_{2^n}$. Such a model, the $\mathbb{F}_{2^n}$-*polynomial chain*, has been described in [RV13, Section 3]. For the sake of completeness, we briefly recollect the definition in Appendix A.

Previous Result. Let us recollect in slightly more details the previous lower bound of $\Omega(\log_2 n)$. The following proposition gives a lower bound on the number

of non-linear multiplications necessary to evaluate a polynomial $P(x)$, a.k.a. *non-linear complexity* of $P(x)$, as the maximum of the quantity necessary to evaluate its monomials. Let $\mathcal{M}(P(x))$ denote the non-linear complexity of $P(x)$. If $P(x)$ corresponds to an n-bit S-box S, then $\mathcal{M}(P(x))$ is also called the *masking complexity* of S.

Proposition 1. [[RV13], Proposition 3] *Let* $P(x) := \sum_{i=0}^{2^n-1} a_i\, x^i$ *be a polynomial in* $\mathbb{F}_{2^n}[x]$. *Then*

$$\mathcal{M}(P(x)) \geq \max_{\substack{0 \leq i < 2^n - 1 \\ a_i \neq 0}} m_n(i),$$

where $m_n(i)$ *is the length of the shortest cyclotomic-class (CC) addition chain of* i *w.r.t.* n.

The following result gives a lower bound on the value of $m_n(i)$ in terms of the Hamming weight of i.

Proposition 2. [[RV13], Proposition 1] $m_n(i) \geq \lceil \log_2(\nu(i)) \rceil$, *where* $\nu(i)$ *is the Hamming weight of the binary representation of* i $(0 \leq i \leq 2^n - 2)$.

Since $\nu\left(2^n - 2\right) = n-1$, hence polynomials having the monomial x^{2^n-2} will have non-linear complexity at least $\log_2\left(n-1\right)$. Hence $\Omega\left(\log_2 n\right)$ is a lower bound on the number of necessary non-linear multiplications required to evaluate polynomials over $\mathbb{F}_{2^n}$.

New Lower Bound. Our technique to prove the lower bound of $\Omega(\sqrt{2^n/n})$ on the non-linear complexity is similar to the one used in the proof of [PS73, Theorem 2]. But we would like to emphasize that their result is not applicable to our setting since they work over the integers and the cost model used there is different from the one used in our case.

Proposition 3. *There exists a polynomial* $P(x) \in \mathbb{F}_{2^n}[x]$ *such that* $\mathcal{M}(P(x)) \geq \sqrt{\frac{2^n}{n}} - 2$.

Proof. At a more abstract level, an $\mathbb{F}_{2^n}$-polynomial chain evaluating $P(x) \in \mathbb{F}_{2^n}[x]$ that uses r non-linear multiplications $(r \geq 0)$ can be equivalently described as a sequence $\mathcal{Z}$ of polynomials z_{-1}, z_0, …, z_r, where

$$\begin{aligned} z_{-1} &= 1, \\ z_0 &= x, \\ z_k &= \left(\beta_{k,-1} + \sum_{i=0}^{k-1} \sum_{j=0}^{n-1} \beta_{k,i,j}\, z_i^{2^j}\right) \cdot \left(\beta'_{k,-1} + \sum_{i=0}^{k-1} \sum_{j=0}^{n-1} \beta'_{k,i,j}\, z_i^{2^j}\right) \\ &\qquad \pmod{x^{2^n} + x}, \end{aligned} \tag{15}$$

where $k = 1, 2, \ldots, r$, $\beta_{k,-1}, \beta'_{k,-1}, \beta_{k,i,j}, \beta'_{k,i,j} \in \mathbb{F}_{2^n}$. Lastly,

$$P(x) = \beta_{r+1,-1} + \sum_{i=0}^{r} \sum_{j=0}^{n-1} \beta_{r+1,i,j}\, z_i^{2^j} \pmod{x^{2^n} + x}, \tag{16}$$

where again $\beta_{r+1,-1}, \beta_{r+1,i,j} \in \mathbb{F}_{2^n}$. .

Since the squaring operation is $\mathbb{F}_2$-linear in $\mathbb{F}_{2^n}$, and that $x^{2^n} = x$ for all $x \in \mathbb{F}_{2^n}$, it is easy to see that any polynomial that can be evaluated using at most t non-linear multiplications will be of the form as given in (16).

The number of parameters $\beta_{k,-1}$, $\beta'_{k,-1}$, $\beta_{k,i,j}$, $\beta'_{k,i,j}$ in (15) for a given value of k ($k = 1, \ldots, r$) is $2 \cdot (k \cdot n + 1)$. In (16), the number of parameters $\beta_{r+1,-1}$, $\beta_{r+1,i,j}$ is $(r+1) \cdot n + 1$. Totally, the number of parameters are

$$(r+1)\, n + 1 + \sum_{k=1}^{r} 2\,(kn+1)\,.$$

Since there are only $|\mathbb{F}_{2^n}|^{2^n}$ distinct polynomials in $\mathbb{F}_{2^n}[x]$ (i.e. up to evaluation), and a given set of values for the parameters enables to evaluate a single polynomial only, we get the following necessary condition to evaluate all polynomials over $\mathbb{F}_{2^n}[x]$

$$\begin{aligned} & |\mathbb{F}_{2^n}|^{(r+1)n+1+\sum_{k=1}^{r} 2(kn+1)} \geq |\mathbb{F}_{2^n}|^{2^n}, \\ \Longrightarrow \quad & (r+1)\, n + 1 + \sum_{k=1}^{r} 2\,(kn+1) \geq 2^n, \\ \Longrightarrow \quad & n \cdot r^2 + (2n+2) \cdot r - (2^n - n - 1) \geq 0, \\ \Longrightarrow \quad & r \geq \sqrt{\frac{2^n}{n}} - 2. \end{aligned} \tag{17}$$

Hence there exists polynomials over $\mathbb{F}_{2^n}$ that require $\Omega(\sqrt{2^n/n})$ non-linear multiplications to evaluate them. □

The above proposition shows that our new method from Section 2.2 is asymptotically optimal.

Concrete Lower Bound. In Table 3 we compare, for various values of n, the previously known lower bound for non-linear complexity with the new lower bound as determined by (17).

n	4	5	6	7	8	9	10	11	12
Previous lower bound [CGP+12,RV13]	2	2	3	3	4	4	4	4	4
Our lower bound (cf. (17))	**0**	**1**	**2**	**3**	**4**	**6**	**9**	**12**	**17**

Table 3. Lower bound for non-linear complexity in $\mathbb{F}_{2^n}$.

Note that there is still a gap between the lower bound from Table 3 and the achievable value of N_{mult} for our method in Table 2. This is because in our

method the decomposition of $P(x)$ as

$$P(x) = \sum_{i=1}^{t-1} p_i(x) \cdot q_i(x) + p_t(x) \tag{18}$$

is performed by first generating the polynomials $q_i(x)$ randomly and independently of $P(x)$, in order to have a linear system of equations over the coefficients of $p_i(x)$. Instead one could try to solve (18) for both the $p_i(x)$ and the $q_i(x)$ polynomials simultaneously; however this gives a quadratic system of equations, which is much harder to solve.

4 Application to various S-boxes

In this section, we apply the generic method described in Section 2, to several well known S-boxes. Using our new method, we reduce the number of non-linear multiplications required in each case, resulting in an improvement over the previously known techniques.

We stress that in our method for an n-bit S-box, the maximum number of non-linear multiplications required is invariant of the choice of the S-box when n is fixed. Hence, the number of non-linear multiplications obtained for a fixed n actually provides an upper bound on the masking complexity of an S-box of size n.

4.1 CLEFIA and Other 8-bit S-boxes

The CLEFIA block cipher has two 8-bit S-boxes [SSA+07]. Let us denote the S-box lookup table for either of the S-boxes as $\mathrm{S}_{\texttt{clefia}}$. We choose

$$L = C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{87} \cup C_{251}. \tag{19}$$

This implies that after choosing $t = 6$, and then 5 basis polynomials $q_i \xleftarrow{\$} \mathcal{P}(x^L)$ $(1 \leq i \leq 5)$, the following system of equations is constructed in $\mathbb{F}_{2^8}$:

$$\mathrm{S}_{\texttt{clefia}}[x_j] = \underbrace{\sum_{i=1}^{5} p_i(x_j) \cdot q_i(x_j)}_{Q} + p_6(x_j) \qquad j = 0, \ldots, 255. \tag{20}$$

We have checked that for some random choice of the polynomials $q_i(x)$ the corresponding matrix A has full rank 256, and therefore we can determine the polynomials $p_i(x)$. Given the solution to the above system, the S-box evaluation is then the same as evaluating the polynomial $Q(x) + p_6(x)$. To evaluate all the monomials in $\{x, x^3, x^7, x^{29}, x^{87}, x^{251}\}$ we need 5 non-linear multiplications, implying that any monomial in x^L, any $q_i(x)$ (randomly chosen from $\mathcal{P}(x^L)$) and any $p_i(x)$ can all together be evaluated with 5 non-linear multiplications. Moreover the evaluation of $Q(x)$ requires 5 additional non-linear multiplications.

Therefore the total number of non-linear multiplications required for evaluating the S-box is 10.

Note that it requires at least 4 non-linear multiplications to evaluate the polynomials corresponding to the two S-boxes of CLEFIA by any method. This is because these two polynomials over $\mathbb{F}_{2^8}$ have degrees 252 (S-box S_0) and 254 (S-box S_1), and the result follows from Proposition 1.

Invariance. If we choose some other 8-bit S-box, then the matrix corresponding to the resulting system remains the same. Hence, we will still get a solution to the system for the same set of polynomials $q_i(x)$. This implies that we can use the same set of basis polynomials to obtain polynomials $p_i(x)$ for any other 8-bit S-box. Hence, for any S-box of size 8, the number of non-linear multiplications is at most 10.

4.2 PRESENT and Other 4-bit S-boxes

For the 4-bit S-box of PRESENT [BKL+07], we choose $t = 2$ and $L = C_0 \cup C_1 \cup C_3$. By selecting $q_1 \xleftarrow{\$} \mathcal{P}(x^L)$, we construct the following linear system of equations:

$$\mathrm{S}_{\mathtt{present}}[x_j] = p_1(x_j) \cdot q_1(x_j) + p_2(x_j) \tag{21}$$

The monomials used to construct $q_1(x)$, $q_2(x)$ are $\{x, x^2, x^4, x^8, x^3, x^6, x^{12}, x^9\}$. All of these monomials can be evaluated with a single non-linear multiplication and to evaluate $p_1(x) \cdot q_1(x)$ we need only one more non-linear multiplication. Hence, the PRESENT S-box evaluation requires 2 multiplications. As in the case of 8-bit S-boxes, this proves that with the same $q_1(x)$ any 4-bit S-box can be evaluated with 2 multiplications. Table 4 gives the corresponding polynomials for the PRESENT S-box.

The polynomial corresponding to the PRESENT S-box has degree 14 and hence, from Proposition 1, its masking complexity is at least 2 [RV13]. This implies that our evaluation method achieves optimal complexity for the PRESENT S-box.

4.3 (m, n)-bit S-box: Application to DES

We now consider S-boxes whose output size n is smaller than the input size m, as for the DES S-boxes with $m = 6$ and $n = 4$. We can view an (m, n)-bit S-box $(m > n)$ as a mapping from $\mathbb{F}_{2^m}$ to $\mathbb{F}_{2^n}$. Given any such S-box table S, we want to construct a system of linear equations

$$\mathsf{S}[x_j] = \underbrace{\sum_{i=1}^{t-1} p_i(x_j) \cdot q_i(x_j) + p_t(x_j)}_{G(x)} \tag{22}$$

Basis Polynomial	
q_1	$(a^3+a^2+1)\cdot x^{12}+(a^3+a^2+a+1)\cdot x^9+a^2\cdot x^8+x^6+(a^3+a^2+a)\cdot x^4+x^2+(a^3+a)\cdot x+a$
Solution to linear System	
p_1	$(a^3+a)\cdot x^{12}+x^9+(a^3+a^2)\cdot x^8+(a^2+1)\cdot x^6+(a^3+a^2+1)\cdot x^4+(a^3+a^2+a+1)\cdot x^3+(a^2+1)\cdot x^2+(a^2+1)\cdot x+a^2$
p_2	$(a^2+1)\cdot x^8+(a^3+a^2+1)\cdot x^6+(a+1)\cdot x^4+a\cdot x^3+x^2+(a^3+1)\cdot x+a^2$

Table 4. Basis polynomial $q_1(x)$ for 4-bit S-boxes, and solutions $p_1(x), p_2(x)$ to PRESENT S-box. The irreducible polynomial is a^4+a+1 over $\mathbb{F}_2$.

Note that each $\mathsf{S}[x_j]$ is an element of the smaller field $\mathbb{F}_{2^n}$, but each $G(x_j)$ is an element in the larger field $\mathbb{F}_{2^m}$. One trivial way to remove this inconsistency is to consider $\mathsf{S}[x_j]$ as an element of the larger field $\mathbb{F}_{2^m}$, by padding the most significant bit of the S-box output with 0's. Then, we determine the polynomials $p_i(x)$ by solving the corresponding system $A \cdot \boldsymbol{c} = \mathsf{S}$, as described in Section 2.2. However intuitively this is not optimal, since we are creating an artificial constraint to be satisfied by the coefficients of the polynomials $p_i(x)$, namely that the $m-n$ most significant bits of $G(x)$ must be 0, while eventually these most significant bits will simply be discarded after the evaluation of $G(x)$, since to get $\mathsf{S}(x)$ we only keep the n least significant bits of $G(x)$.

Instead, we consider the representations of the unknown coefficients of the polynomials $p_i(x)$ in $\mathbb{F}_2$ instead of $\mathbb{F}_{2^m}$, and we transform the system of linear equations (22) over $\mathbb{F}_{2^m}$, into a system of linear equations over $\mathbb{F}_2$. By doing this, from each constraint $G(x_j)$, we generate m equations over $\mathbb{F}_2$, instead of one equation over $\mathbb{F}_{2^m}$. Note that each of these m equations will be an affine combination of the unknown bits of the coefficients of the polynomials $p_i(x)$. Only n of these equations are actually necessary, since the output of the S-box is of size n bits. By equating each of these equations to the corresponding output bit of the S-box, we get a transformed system of linear equations $B \cdot \boldsymbol{c} = \mathsf{S}$, where B is an $(n \cdot 2^m) \times (t \cdot |L| \cdot m)$ matrix over $\mathbb{F}_2$ and L is the set of elements from the chosen cyclotomic classes. By solving this transformed system over $\mathbb{F}_2$ we determine the polynomials $p_i(x)$.

Example of DES. The DES block cipher has 8 $(6,4)$-bit S-boxes [oST93]. A DES S-box is a mapping from $\mathbb{F}_{2^6}$ to $\mathbb{F}_{2^4}$. In [RV13], the authors consider the S-boxes as a mapping from $\mathbb{F}_{2^6}$ to $\mathbb{F}_{2^6}$, where the two most significant bits of the output of S-box are fixed to 0, and as recalled in Section 2.1 the evaluation can be done with 7 non-linear multiplications. Also, for the same representation, there is a lower bound of 3 non-linear multiplications necessary to evaluate each DES S-box [RV13]. From Table 2, using our generic method over $\mathbb{F}_{2^6}$ we can perform the evaluation with 5 non-linear multiplications. Below we show that

by working over $\mathbb{F}_2$ as explained above, only 4 non-linear multiplications are required.

We choose $L = C_0 \cup C_1 \cup C_3 \cup C_7$, $t = 3$, and $q_1(x), q_2(x) \xleftarrow{\$} \mathcal{P}(x^L)$. Then using our method we transform the following linear system of equations

$$\mathrm{S}_{\mathtt{des}}[x_j] = \underbrace{\sum_{i=1}^{2} p_i(x_j) \cdot q_i(x_j)}_{Q(x)} + p_3(x_j) \tag{23}$$

to a system over $\mathbb{F}_2$. That is, instead of embedding $\mathrm{S}_{\mathtt{des}}$ into $\mathbb{F}_{2^6}$, we write the system of equations over $\mathbb{F}_2$. This can be done by considering the binary representation of x^α evaluated at any given value in $\mathbb{F}_{2^6}$. This will give 6 equations over $\mathbb{F}_2$ for each equation $Q(x_j) + p_3(x_j)$. Out of these 6 equations only 4 will be necessary since the output of DES S-box has 4-bit values. By solving this new system of linear equations over $\mathbb{F}_2$ we can determine $p_i(x)$ for each i.

The number of multiplications required to evaluate $q_1(x), q_2(x)$ is 2, and $Q(x)$ can be evaluated with 2 additional multiplications. Hence, the total number of non-linear multiplications required is only 4. In Appendix C we give an example of basis polynomials $q_1(x)$, $q_2(x)$ for DES and the solution polynomials $p_i(x)$ corresponding to the system of linear equations for the first DES S-box S_1.

As previously, once we obtain a full rank matrix for a set of randomly fixed $q_1(x), q_2(x)$, for any other $(6, 4)$-bit S-box we can use this basis to find the corresponding polynomials $p_i(x)$, since the matrix A is independent from the S-box. Hence we can conclude that the masking complexity of any $(6, 4)$-bit S-box is at most 4.

4.4 Implementation Results: DES

We have performed a software implementation of the CGPQR countermeasure [CGP+12] for DES that incorporates our new polynomial evaluation technique requiring only 4 NLMs. We have implemented this in C on a Dell Latitude 13 notebook running Ubuntu 12.04 Linux. The processor is Intel Core 2 Duo (32-bit architecture) running at 1.3 GHz. Our implementation is based on the source code available from [Cor13]. The present implementation is also publicly available at [Cor13]. We have used the technique of tabulating linear polynomials from Remark 3 in the implementation of our polynomial evaluation method. Note that these tables corresponding to the linear polynomials need to be stored only in the ROM.

In Table 5, we have compared the above timing results with that of the CGPQR countermeasure implemented with the Roy-Vivek technique [RV13] that requires 7 NLMs, and also compared with that of the higher-order table recomputation method of Coron [Cor14]. In Table 5, the parameter t' refers to the order of security and n' refers to the number of shares in the full security model of [ISW03]. Note the relation $n' = 2t' + 1$. The (RAM) memory requirement (in bytes) is provided only for the S-box computations and the overall execution

time for a DES encryption is in milliseconds. The penalty factor (PF) gives the ratio of the execution time of a given method to that of an unprotected implementation. The number of calls to the random number generator is 1000 times that of the reported quantity.

Table 5. Comparison of secure implementations of DES.

Method	t'	n'	Rand $\times 10^3$	Mem (bytes)	Time (ms)	PF
Unprotected					0.008	1
CGPQR+RV	1	3	2752	72	0.552	69
Table Recomputation	1	3	8512	423	0.279	34
CGPQR+this work	1	3	2368	40	0.166	20
CGPQR+RV	2	5	9152	118	0.966	120
Table Recomputation	2	5	33472	691	0.612	76
CGPQR+this work	2	5	7872	64	0.336	42
CGPQR+RV	3	7	19200	164	1.507	188
Table Recomputation	3	7	74880	959	1.091	136
CGPQR+this work	3	7	16512	88	0.591	73
CGPQR+RV	4	9	32896	210	2.167	270
Table Recomputation	4	9	132736	1227	1.696	212
CGPQR+this work	4	9	28288	112	0.905	113

Acknowledgements

We would like to thank Matthieu Rivain for suggesting us the technique of tabulating linear polynomials described in Remark 3.

References

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[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, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[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*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.

[Cor13] Jean-Sebastien Coron. `https://github.com/coron/htable/`, 2013.

[Cor14] Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.

[CPRR13] Jean-Sebastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In *FSE*, 2013. To appear.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[GST13] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. *IACR Cryptology ePrint Archive*, 2013:857, 2013.

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

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

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

[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.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, 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*, volume 8086 of *Lecture Notes in Computer Science*, pages 417–434. Springer, 2013.

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

A $\mathbb{F}_{2^n}$-Polynomial Chain

Definition 1. [[RV13], Definition 4] *An* $\mathbb{F}_{2^n}$-polynomial chain S *for a polynomial* $P(x) \in \mathbb{F}_{2^n}[x]$ *is defined as*

$$\lambda_{-1} = 1,\ \lambda_0 = x,\ \ldots\ ,\ \lambda_r = P(x) \tag{24}$$

where

$$\lambda_i = \begin{cases} \lambda_j + \lambda_k & -1 \leq j, k < i, \\ \lambda_j \cdot \lambda_k & -1 \leq j, k < i, \\ \alpha_i \odot \lambda_j & -1 \leq j < i, \ \alpha_i \text{ is a scalar}, \\ \lambda_j^2 & -1 \leq j < i. \end{cases}$$

Though $\cdot$ and $\odot$ both perform multiplication in $\mathbb{F}_{2^n}$, the operator "$\odot$" is reserved for the multiplication by a scalar. A step such as $\lambda_j \cdot \lambda_k$ denotes a non-linear multiplication. Let the number of non-linear multiplications involved in a chain S be denoted as $\mathcal{N}(S)$. Then the non-linear complexity *of $P(x)$, denoted by $\mathcal{M}(P(x))$, is defined as $\mathcal{M}(P(x)) = \min_S \mathcal{N}(S)$.*

B Heuristics for choosing parameters t and L

n	t	L	$\lvert L\rvert$
4	2	$C_0 \cup C_1 \cup C_3$	9
5	3	$C_0 \cup C_1 \cup C_3 \cup C_7$	16
6	3	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{11}$	25
7	4	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{11} \cup C_{15}$	36
8	6	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{87} \cup C_{251}$	49
9	8	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{45} \cup C_{119} \cup C_{191} \cup C_{255}$	73
10	11	$C_0 \cup C_1 \cup C_3 \cup C_7 \cup C_{29} \cup C_{45} \cup C_{119} \cup C_{191} \cup C_{155} \cup C_{255} \cup C_{339}$	101

C Evaluation Polynomials for DES S-boxes

In Table 6 we give an example of basis polynomials $q_1(x)$, $q_2(x)$ for DES and Table 7 shows the solution polynomials $p_i(x)$ corresponding to the system of linear equations for the first DES S-box S_1.

Basis Polynomials	
q_1	$(a^5+a^4+1)\cdot x^{56}+(a^5+1)\cdot x^{49}+(a^2+a)\cdot x^{48}+(a^4+a^3)\cdot x^{35}+(a^5+a^4+a^2)\cdot x^{33}+(a^5+a+1)\cdot x^{32}+(a^3+a)\cdot x^{28}+a^2\cdot x^{24}+(a^5+1)\cdot x^{16}+(a^4+a+1)\cdot x^{14}+x^{12}+(a^4+a^3+a^2+1)\cdot x^8+(a^5+a^3+a^2+a+1)\cdot x^7+(a^5+a^4+a^3+a^2+1)\cdot x^6+(a^5+a^4+a^3+1)\cdot x^4+(a^5+a^2+a+1)\cdot x^3+(a^3+a^2+a)\cdot x^2+(a^4+a^2+a+1)\cdot x+a^5+a^4+a^3+a^2+a$
q_2	$(a+1)\cdot x^{56}+(a^5+1)\cdot x^{49}+(a+1)\cdot x^{48}+a\cdot x^{35}+(a+1)\cdot x^{33}+(a^4+a^3+a+1)\cdot x^{32}+(a^3+a^2+a)\cdot x^{28}+(a^5+a^3+a+1)\cdot x^{24}+(a^3+1)\cdot x^{16}+(a^4+a^2+1)\cdot x^{14}+(a+1)\cdot x^{12}+(a^5+a^4+1)\cdot x^8+(a^5+a^4+a^3+a+1)\cdot x^7+(a^5+a^4+a^3)\cdot x^6+(a+1)\cdot x^4+(a^5+a^3+a^2+a)\cdot x^2+a\cdot x+a^5+a^4+a^3+a^2+1$

Table 6. Basis polynomials q_1, q_2 obtained from $\mathcal{P}(x^L)$, for DES.

Solution to linear system	
p_1	$(a^5+a^4+a^3+a^2+1)\cdot x^{56}+(a^5+a^2+1)\cdot x^{49}+a^4\cdot x^{48}+(a^4+a^3+a)\cdot x^{35}+(a^5+a^4+a^2)\cdot x^{33}+(a^5+1)\cdot x^{32}+a\cdot x^{28}+(a^4+a^2)\cdot x^{24}+(a^5+a)\cdot x^{16}+(a^5+a^2)\cdot x^{14}+(a^5+a+1)\cdot x^{12}+(a^5+a^4+a^3+a)\cdot x^8+(a^5+a^4+a^3+a)\cdot x^7+(a^5+a^4+a^3)\cdot x^6+(a^2+a+1)\cdot x^4+(a^5+a^4+a)\cdot x^2+(a^5+a^4+1)\cdot x+a^4+a^3+a^2$
p_2	$(a^5+a^2)\cdot x^{49}+(a^3+1)\cdot x^{48}+(a^5+a^3+a+1)\cdot x^{35}+(a^4+a^2+1)\cdot x^{33}+(a^5+a^4+1)\cdot x^{32}+(a^5+a^4+a^3+a+1)\cdot x^{28}+(a^3+a^2)\cdot x^{24}+(a^2+a+1)\cdot x^{16}+(a^5+a^4+a^3)\cdot x^{14}+(a^4+a^3+a+1)\cdot x^{12}+(a^4+a^3)\cdot x^8+(a^5+a)\cdot x^7+(a^5+a^4)\cdot x^6+(a^5+a^4+a^3+a^2+a+1)\cdot x^4+(a^5+a^4+a)\cdot x^3+(a^5+a^3+a+1)\cdot x^2+(a^5+a)\cdot x+a^5+a^4+a^2+a$
p_3	$a\cdot x^7+a\cdot x^6+(a^4+a+1)\cdot x^4+(a^5+a^2+a)\cdot x^3+(a^5+a^4+a+1)\cdot x^2+(a^4+a^2)\cdot x$

Table 7. Solution to the system of linear equations for DES S-box (S_1). The irreducible polynomial is a^6+a+1 over $\mathbb{F}_2$.