Noname manuscript No.
(will be inserted by the editor)

Threshold Implementations of Small S-boxes

**Begül Bilgin · Svetla Nikova ·
Ventzislav Nikov · Vincent Rijmen ·
Natalia Tokareva · Valeriya Vitkup**

the date of receipt and acceptance should be inserted later

Abstract Threshold implementation (TI) is a masking method that provides security against first-order DPA with minimal assumptions on the hardware. It is based on multi-party computation and secret sharing. In this paper, we provide an efficient technique to find TIs for all 3 and 4-bit permutations which also covers the set of 3×3 and 4×4 invertible S-boxes. We also discuss alternative methods to construct shared functions by changing the number of variables or shares. Moreover, we further consider the TI of 5-bit almost bent and 6-bit almost perfect nonlinear permutations. Finally, we compare the areas of these various TIs.
Keywords: DPA, masking, glitches, sharing, nonlinear functions, S-box, decomposition

1 Introduction

The computation of a cryptographic algorithm leaks *side-channel information*. Side-channel analysis (SCA) uses this information to reveal the secret such as the key that is used in the algorithm. The most common SCA technique is to analyze the power consumption of the device using differential power analysis (DPA). This analysis exploits the correlation between the instantaneous power consumption of a device and the intermediate results of a cryptographic algorithm.

Several countermeasures against side-channel attacks have been proposed. Some introduce noise in the side-channel such as executing random delays or dummy operations. There are also circuit design approaches [47] that try to balance the

Begül Bilgin · Svetla Nikova · Vincent Rijmen
Katholieke Universiteit Leuven, ESAT-COSIC and iMinds, Belgium
firstname.lastname@esat.kuleuven.be

Begül Bilgin
University of Twente, EEMCS-DIES, The Netherlands

Ventzislav Nikov
NXP Semiconductors, Belgium

Natalia Tokareva · Valeriya Vitkup
Sobolev Institute of Mathematics and Novosibirsk State University, Russia

power consumption of different data values. Masking is an alternative approach to randomize the intermediate values of an algorithm. This can be done at the algorithm level [1, 13, 26, 41, 44], at the gate level [27, 48] or even in combination with circuit design approaches [42]. However, it has been shown that *hardware* implementations are difficult to protect against DPA [33, 35] due to presence of glitches. The main advantages of the threshold implementation approach are that it provides provable security against first-order DPA attacks with minimal assumptions on the hardware technology, in particular, it is also secure in the presence of glitches, and that the method allows to construct realistic-size circuits [38–40]. However, a TI can still be broken by univariate mutual information analysis [3] or univariate higher order attacks [34].

It has been shown that all 3 and 4-bit permutations have TI with 3, 4 or 5 shares [11]. There are many S-boxes used in cryptographic algorithms that are chosen from these sets of permutations. The TI approach is also applied to some algorithms such as PRESENT [43], AES [8, 36] and KECCAK [4, 7].

Contribution. This paper is an extended and completed version of the paper presented at CHES 2012 [11]. In [11] we have proposed two techniques to find sharings which satisfy the properties of TI, namely direct sharing and sharing with correction terms. We have shown how these techniques can be applied to all 3 and 4-bit permutations. We left then as an open question if TI with 3 shares for some quadratic permutation classes can be found.

The current submission contains the following new contributions. First, we provide an answer to the above mentioned open question posed in the earlier version of the paper (Section 3.2). Second, we extend our previous work by providing TI for 5-bit almost bent and 6-bit almost perfect non-linear permutations which have cryptographic significance since they provide optimum differential and linear properties (Section 4). Third, we provide area distributions for various classes with different numbers of shares in the open cell library NANGATE [37] (Section 5). Finally, we propose two extensions to the basic sharing approach by using virtual shares and virtual variables, and by varying the number of the shares (Section 6).

Organization. After providing some preliminary information on classification of permutations and threshold implementation in Section 2, we recall the previous work on 3 and 4-bit permutations in Section 3. We provide an answer in Section 3.2 to an open question from [11] whether 3 shares sharing exists for all 3-bit permutations. In Section 4, we improve the previous work to some 5 and 6-bit permutations which have cryptographic significance. We provide area requirements of all these permutations in Section 5. In the basic approach of TI, the number of input and output shares and variables are the same. In Section 6, we propose two extensions to the basic sharing approach: namely using virtual shares and virtual variables, and varying the number of the shares.

2 Preliminaries

2.1 Permutations and Affine Equivalence Relations

We consider n-bit permutations sometimes defined over a vector space $\mathcal{F}_2^n$ or over a finite field $GF(2^n)$ some of which define $n \times n$ invertible S-boxes that are used in cryptographic algorithms. The degree of such a permutation F is the algebraic degree of the (n, n) vectorial Boolean function [14]. Any such function $F(x)$ can be considered as an n-tuple of Boolean functions $(f_1(x), \ldots, f_n(x))$ called the coordinate functions of $F(x)$.

All permutations from a set X to itself form the *symmetric group* on X denoted by S_X. A transposition is a permutation which exchanges two elements and keeps all others fixed. A classical theorem states that every permutation can be written as a product of transpositions [45], and although the representation of a permutation as a product of transpositions is not unique, the number of transpositions needed to represent a given permutation is either always even or always odd. The set of all even permutations form a normal subgroup of S_X, which is called the *alternating group* on X and denoted by A_X. The alternating group contains half of the elements of S_X. Instead of A_X and S_X, we will write here A_m and S_m, where m is the size of the set X.

Lemma 1 ([49]) *For all $n \geq 3$, the n-bit affine permutations are in the alternating group.*

Definition 1 ([21]) Two permutations $S_1(x)$ and $S_2(x)$ are *affine/linear equivalent* if there exists a pair of affine/linear permutations $A(x)$ and $B(x)$, such that $S_1 = B \circ S_2 \circ A$.

Every affine permutation $A(x)$ can be written as $\mathrm{A} \cdot x + a$ with a an n-bit constant and A an $n \times n$ matrix which is invertible over $GF(2)$. It follows that there are

$$2^n \times \prod_{i=0}^{n-1} (2^n - 2^i) \tag{1}$$

different affine permutations.

The relation "being affine equivalent" can be used to define equivalence classes. Note that the algebraic degree is invariant under affine equivalence, hence all permutations in a class have the same algebraic degree. Moreover, if a permutation is represented with an even (resp. odd) number of transpositions, all of its affine equivalent permutations are also represented with an even (resp. odd) number of transpositions.

It is well known that all 2-bit permutations are affine, hence there is only one class. The set of 3-bit permutations contains 4 equivalence classes [21]: 3 classes containing quadratic functions, and one class containing the affine functions. The Inversion in $GF(2^3)$ and the S-boxes of the PRINTcipher [28], the Threeway [19] and the Baseking [20] algorithms, which are the only cryptographically significant 3×3 S-boxes, belong to the same class, denoted by Q_3^3 in this paper with the representative given in Table 9.

The maximal algebraic degree of a balanced n-variable Boolean function is $n-1$ [16, 31]. De Cannière [12, 21] uses an algorithm to search for the affine equivalent classes which guesses the effect of the affine permutation A for as few input points as possible, and then uses the linearity of A and B (as given in Definition 1) to follow the implications of these guesses as far as possible. This search is accelerated by applying the next observation, which follows from linear algebra arguments (change of basis):

Lemma 2 ([30]) *Let S be an n-bit permutation. Then S is affine equivalent to another permutation $\tilde{S}$ with $\tilde{S}(i) = i$, for $i \in \{0, 1, 2, 4, 8, \ldots, 2^{n-1}\}$.*

In the case $n = 4$, this observation reduces the search space from $16! \approx 2^{44}$ to $11! \approx 2^{25}$.

De Cannière lists the 302 equivalence classes for the 4-bit permutations [21]: the class of affine functions, 6 classes containing quadratic functions and the remaining 295 classes containing cubic functions. The classes are listed in Tables 10–12 in the Appendix. The numberings of the classes are derived from the lexicographical ordering of the truth tables of the permutations. In order to increase readability, we introduce the following notation $\mathcal{A}^n_i$, $\mathcal{Q}^n_j$, $\mathcal{C}^n_k$ to denote the Affine class number i, Quadratic class number j and Cubic class number k of permutations of $\mathcal{F}^n_2$. There are many cryptographically significant 4-bit permutations. First Leander and Poschmann [30] and later Saarinen et al. [46] classify all 4×4 invertible S-boxes up to affine equivalence and provide 16 "golden" S-box classes that provide optimal differential and linear properties which would bring an advantage against cryptanalysis. Table 14 in the Appendix lists some of the S-boxes used in the design of cryptographic algorithms together with golden S-boxes (depicted as Optimal G_i) and the classes to which they belong.

On the other hand, the number of classes increase exponentially when permutations with bigger sizes are considered. There exists roughly 2^{61} and 2^{215} different classes for 5-bit and 6-bit permutations respectively [21]. The new hash function standard Keccak [5] uses a 5-bit permutation as its non-linear layer. We will provide a separate discussion for the 5-bit and 6-bit permutations in Section 4.

2.2 Glitches and First-Order DPA

Glitches are undesired transitions in the output of a cell that occur before the signal settles to its intended value. In a CMOS circuit, which is one of the most widely used circuit types, glitches occur a lot mainly because of two reasons. The first reason is that the wires in the circuit which carry input values of a cell have different lengths and propagation delays which might cause the inputs to a cell arrive in different times. Similarly, each type of cell can also have a different propagation delay which affect the cells that take an output value of a cell as an input. Another reason is that some cells take output values of other cells, which are in different stages of the circuit, as inputs. This glitching naturally increases together with the increase of stages in the combination circuit. The amount of glitching cells has a strong impact on the power consumption and the fact that this glitching is data dependent makes the circuit vulnerable to DPA. More information on glitches can be found in [32].

A first-order DPA can be put in correspondence with observing the value of one single wire in the circuit [18, 25]. In a glitchy circuit, that can be considered as observing (unintended) intermediate values of the circuit during the calculation of a function which can give considerably high amount of information on the data the function uses.

2.3 Threshold Implementations (TIs)

TI is a kind of side-channel attack countermeasure, based on (t, n) secret sharing schemes and techniques from multiparty computation. The approach can be summarized as follows. Split a variable $x \in \mathcal{F}_2^m$ into s shares $x_i \in \mathcal{F}_2^m$ by means of Boolean masking satisfying (s, s) secret sharing schemes. An (s, s) secret sharing is defined as distributing parts of a secret x among $t = s$ players such that the information from at least $n = s$ players are required to calculate the secret, i.e. in the TI setting without loss of generality the shares $x_1, \ldots, x_{s-1}$ are randomly chosen variables from a uniform distribution and x_s is calculated so that the XOR sum of these shares is equal to the variable itself ($x = \sum_i x_i$). Hence, the knowledge of up to $s - 1$ shares does not reveal any information on x. We denote the vector of the s shares x_i by $\mathbf{x} \in \mathcal{F}_2^{ms}$ s.t $\mathbf{x} = (x_1, x_2, \ldots, x_s)$. In order to implement a function $\mathcal{F}_2^n \ni a = F(x)$ from $\mathcal{F}_2^m$ to $\mathcal{F}_2^n$, the TI method requires a *sharing*, i.e. a vector $\mathbf{F}$ of s functions F_i which together compute the output(s) of F. We call each component F_i of $\mathbf{F}$ as a component function[1]. A sharing needs to satisfy three properties for TI:

Correctness: For all $a \in \mathcal{F}_2^n$, $a = F(x)$ implies that $a = \sum_i a_i = \sum_i F_i(\mathbf{x})$ for all $\mathbf{x}$ satisfying $\sum_i x_i = x$ and $x \in \mathcal{F}_2^m$.

Non-completeness: Every function is independent of at least one share of the input variable x to provide security against first order side channel attacks. This is often translated as "F_i should be independent of x_i" hence x_i is not an input of F_i. This property, which is not enforced in standard masking methods, provides first-order DPA security of $\mathbf{F}$ in the presence of glitches with the condition that the input is an (s, s) sharing. Hence the lack of this property would make the circuit of $\mathbf{F}$ vulnerable against glitch attacks as will be discussed later.

Uniformity: For all $(a_1, a_2, \ldots, a_s)$ satisfying $\sum_i a_i = a$, the number of valid sharing $\mathbf{x} \in \mathcal{F}_2^{ms}$ for which $F_j(\mathbf{x}) = a_j, 1 \leq j \leq s$, is equal to $2^{(s-1)(m-n)}$ times the number of $x \in \mathcal{F}_2^m$ for which $a = F(x)$.

If F is a permutation on $\mathcal{F}_2^m$, then the functions F_i define together a permutation on $\mathcal{F}_2^{ms}$. This further implies that the sharing $\mathbf{F}$ is balanced (as defined in [16]). If, on the other hand, $m < n$, then the uniformity can be achieved only if the number of shares in the input is larger than the number of shares in the output. In this paper, except Section 6, we restrict ourselves to the case where the input and output have the same number of shares (denoted by s).

If the input sharing is uniform and the sharing is correct and non-complete, then any single component function F_i is independent of the unmasked value x. Since each of the F_i is independent of x, each of the sub-circuits implementing one of the F_i is independent of x. Alternatively, the intermediate values of one wire in the circuit mentioned in Section 2.2, do not provide information depending on all the shares which could lead to x. Hence, none of the sub-circuits, by itself, can leak any information on the unmasked value even in the presence of glitches. Under the assumption that the total leakage of all the sub-circuits when implemented on the same chip is the same as the sum of the leakages of the sub-circuits when implemented separately, we can invoke the linearity of the expectation operator to derive that the average power consumption of the circuit doesn't leak any information on the unmasked values, even in the presence of glitches. Hence, we have perfect resistance against attacks based on the average power consumption, e.g. first-order side-channel attacks.

[1] The component function defined for shared functions in this paper is different than the definition provided in [16]

In order to achieve correctness and non-completeness, we need at least $n+1$ shares for a function with algebraic degree n as proven in [40]. Even though it is possible to consider the full permutation from a cryptographic algorithm as F and try to satisfy the correctness and non-completeness properties directly, such a TI is infeasible due to the high degree of the permutation. Typically, the cryptographic algorithms have more than one rounds each of them taking the previous round output as input. Hence, one can consider each round as F and with a lower degree and satisfy the correctness and non-completeness properties with uniform input sharing for each round. In that case, in order to satisfy the non-completeness in the following rounds, each round should be separated by registers. Moreover, in order to guarantee the uniformity of the input sharing on the following rounds, we require the sharing of each round function to be uniform [8].

2.4 TI and Affine Equivalence

Theorem 1 *If we have a TI for a representative of an affine equivalence class (as in Definition 1), then we can derive a TI for all permutations from the same class.*

Proof Let S be an n-bit permutation which has a uniform, non-complete and correct sharing $\mathbf{S}$ using s shares S_i. Denote the input vector of S by x, and the shares by x_i. Each S_i contains n coordinate shared functions depending on at most $(s-1)$ of the x_i, such that the non-completeness property is satisfied. Without loss of generality, we denote by $\mathbf{x}_i$ the vector $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_s)$ which contains the $s-1$ inputs of S_i.

We now construct a uniform, non-complete and correct sharing for any permutation $\tilde{S}$ which is affine equivalent to S. By Definition 1, there exist two n-bit affine permutations A and B s.t. $\tilde{S} = B \circ S \circ A$. In order to lighten the notation, we give the proof for the case that A and B are linear permutations. We define $\mathbf{A}, \mathbf{B}$ as the $ns \times ns$ permutations that apply A, respectively B, to each of the shares separately:

$$\mathbf{A}(x_1, x_2, \ldots x_s) = (A(x_1), A(x_2), \ldots A(x_s)),$$
$$\mathbf{B}(x_1, x_2, \ldots x_s) = (B(x_1), B(x_2), \ldots B(x_s)).$$

Denote $y_i = A(x_i), 1 \leq i \leq s$ and without loss of generality define $\mathbf{y}_i$ as the vector $y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_s$ that we need to compute S_i. Consider $\mathbf{S}(\mathbf{A}(x_1, x_2, \ldots, x_s)) = (S_1(\mathbf{y}_1), S_2(\mathbf{y}_2), \ldots, S_s(\mathbf{y}_s))$. By slight abuse of notation we can write $\mathbf{y}_i = \mathbf{A}(\mathbf{x}_i)$ and see that the non-completeness of the S_i is preserved in $\mathbf{S} \circ \mathbf{A}$. Since $\mathbf{A}$ is a permutation, it preserves the uniformity of the input and since $\mathbf{S}$ is uniform so will be the composition $\mathbf{S} \circ \mathbf{A}$. The correctness follows from the fact that $\mathbf{S}$ is a correct sharing and that

$$y_1 + y_2 + \ldots + y_s = A(x_1) + A(x_2) + \ldots + A(x_s) = A(x_1 + x_2 + \ldots x_s) = A(x).$$

Consider now $\mathbf{B}(\mathbf{S}(\mathbf{A}(\mathbf{x}))) = (B(S_1(\mathbf{y}_1)), B(S_2(\mathbf{y}_2)), \ldots, B(S_s(\mathbf{y}_s)))$. Since $\mathbf{B}$ is a permutation, it preserves uniformity of the output and since $\mathbf{S}$ is uniform, the composition $\mathbf{B} \circ \mathbf{S}$ is uniform. The composition is non-complete since the S_i are non-complete and $\mathbf{B}$ does not combine different shares. Correctness follows from the fact that $\mathbf{S}$ is a correct sharing and hence

$$\begin{aligned} & B(S_1(\mathbf{y}_1)) + B(S_2(\mathbf{y}_2)) + \ldots + B(S_s(\mathbf{y}_s)) \\ = \ & B(S_1(\mathbf{y}_1) + S_2(\mathbf{y}_2) + \ldots + S_s(\mathbf{y}_s)) = B(S(A(x))). \end{aligned} \qquad \square$$

3 Permutations of Size $n = 3, 4$

In this section, we only consider 3-bit and 4-bit permutations unless stated otherwise explicitly for generalization.

Lemma 3 *There is a transformation which expands $\mathcal{Q}_1^3$, $\mathcal{Q}_2^3$ and $\mathcal{Q}_3^3$ (in Table 9) into $\mathcal{Q}_4^4$, $\mathcal{Q}_{12}^4$ and $\mathcal{Q}_{300}^4$ (in Table 10-12) correspondingly.*

Proof Starting from a 3-bit permutation S and adding a new variable we can obtain a 4-bit permutation $\tilde{S}$. Namely, the transformation is defined as follows: let $S(y, z, w) = (a, b, c)$ and define $\tilde{S}(x, y, z, w) = (a, b, c, x)$. It is easy to check that this transformation maps the first 3 classes into the other 3 classes. □

The relation from Lemma 3 explains why if we have a TI for a class in $\mathcal{F}_2^3$ we also obtain a TI for the corresponding class in $\mathcal{F}_2^4$ and vice versa, i.e., if we cannot implement a class with TI then the corresponding class cannot be implemented with TI either.

Note that S^{-1}, the inverse permutation, is not necessarily affine equivalent to S and in this case may not have the same algebraic degree. We know however, that the inverse of an affine permutation is always an affine permutation. In the case of 3-bit permutations it follows that the inverse of a quadratic permutation is again a quadratic permutation. Moreover, it can be shown that the 3 quadratic classes in S_8 are self-inverse, i.e. S^{-1} belongs to the same class as S. In the case $n = 4$, we can apply the following lemma.

Lemma 4 ([14]) *Let F be a permutation of $GF(2^n)$, then deg$(F^{-1}) = n - 1$ if and only if deg$(F) = n - 1$.*

Since the inverse of an affine permutation is affine, and, when $n = 4$, the inverse of a cubic permutation is cubic, it follows that in this case the inverse of a quadratic permutation is quadratic. The Keccak S-box ($n = 5$) [5], which is a permutation, is as an example where the algebraic degree of the inverse S-box (3) is different from the algebraic degree of the S-box itself (2).

We have observed that there are 172 self-inverse classes in the symmetric group S_{16}. The remaining 130 classes form 65 pairs, i.e., any permutation S of the first class has an inverse permutation S^{-1} in the second class (and vice versa). Table 1 gives the list of the pairs of inverse classes.

3.1 Direct Sharing

The most difficult property of TI to be satisfied when the function is shared is the uniformity. Assume that we want to construct a TI for the function $F(x, y, z)$ with 3 shares. It is easy to produce a sharing which satisfies the correctness and the

Table 1: Pairs of inverse classes

65 pairs of inverse classes; the remaining 172 classes are self-inverse
$(\mathcal{C}^4_{29},\mathcal{C}^4_{30}),(\mathcal{C}^4_{33},\mathcal{C}^4_{34}),(\mathcal{C}^4_{39},\mathcal{C}^4_{40}),(\mathcal{C}^4_{43},\mathcal{C}^4_{44}),(\mathcal{C}^4_{47},\mathcal{C}^4_{48}),(\mathcal{C}^4_{49},\mathcal{C}^4_{50}),(\mathcal{C}^4_{52},\mathcal{C}^4_{53}),(\mathcal{C}^4_{58},\mathcal{C}^4_{59}),(\mathcal{C}^4_{60},\mathcal{C}^4_{61}),$ $(\mathcal{C}^4_{63},\mathcal{C}^4_{64}),(\mathcal{C}^4_{66},\mathcal{C}^4_{67}),(\mathcal{C}^4_{68},\mathcal{C}^4_{69}),(\mathcal{C}^4_{70},\mathcal{C}^4_{71}),(\mathcal{C}^4_{73},\mathcal{C}^4_{74}),(\mathcal{C}^4_{79},\mathcal{C}^4_{80}),(\mathcal{C}^4_{85},\mathcal{C}^4_{86}),(\mathcal{C}^4_{87},\mathcal{C}^4_{88}),(\mathcal{C}^4_{90},\mathcal{C}^4_{91}),$ $(\mathcal{C}^4_{93},\mathcal{C}^4_{94}),(\mathcal{C}^4_{95},\mathcal{C}^4_{96}),(\mathcal{C}^4_{97},\mathcal{C}^4_{98}),(\mathcal{C}^4_{103},\mathcal{C}^4_{104}),(\mathcal{C}^4_{105},\mathcal{C}^4_{106}),(\mathcal{C}^4_{108},\mathcal{C}^4_{109}),(\mathcal{C}^4_{110},\mathcal{C}^4_{111}),(\mathcal{C}^4_{112},\mathcal{C}^4_{113}),$ $(\mathcal{C}^4_{114},\mathcal{C}^4_{115}),(\mathcal{C}^4_{116},\mathcal{C}^4_{117}),\ (\mathcal{C}^4_{120},\mathcal{C}^4_{121}),(\mathcal{C}^4_{123},\mathcal{C}^4_{124}),(\mathcal{C}^4_{126},\mathcal{C}^4_{127}),(\mathcal{C}^4_{128},\mathcal{C}^4_{129}),(\mathcal{C}^4_{130},\mathcal{C}^4_{131}),$ $(\mathcal{C}^4_{132},\mathcal{C}^4_{133}),(\mathcal{C}^4_{143},\mathcal{C}^4_{144}),(\mathcal{C}^4_{147},\mathcal{C}^4_{148}),(\mathcal{C}^4_{150},\mathcal{C}^4_{151}),(\mathcal{C}^4_{152},\mathcal{C}^4_{153}),(\mathcal{C}^4_{154},\mathcal{C}^4_{155}),(\mathcal{C}^4_{156},\mathcal{C}^4_{157}),$ $(\mathcal{C}^4_{158},\mathcal{C}^4_{159}),(\mathcal{C}^4_{161},\mathcal{C}^4_{162}),(\mathcal{C}^4_{164},\mathcal{C}^4_{165}),(\mathcal{C}^4_{166},\mathcal{C}^4_{167}),(\mathcal{C}^4_{169},\mathcal{C}^4_{170}),(\mathcal{C}^4_{171},\mathcal{C}^4_{172}),(\mathcal{C}^4_{181},\mathcal{C}^4_{182}),$ $(\mathcal{C}^4_{183},\mathcal{C}^4_{184}),(\mathcal{C}^4_{185},\mathcal{C}^4_{186}),(\mathcal{C}^4_{190},\mathcal{C}^4_{191}),(\mathcal{C}^4_{199},\mathcal{C}^4_{200}),(\mathcal{C}^4_{201},\mathcal{C}^4_{202}),(\mathcal{C}^4_{203},\mathcal{C}^4_{204}),(\mathcal{C}^4_{206},\mathcal{C}^4_{207}),$ $(\mathcal{C}^4_{209},\mathcal{C}^4_{210}),(\mathcal{C}^4_{211},\mathcal{C}^4_{212}),(\mathcal{C}^4_{214},\mathcal{C}^4_{215}),(\mathcal{C}^4_{226},\mathcal{C}^4_{227}),(\mathcal{C}^4_{229},\mathcal{C}^4_{230}),(\mathcal{C}^4_{233},\mathcal{C}^4_{234}),(\mathcal{C}^4_{241},\mathcal{C}^4_{242}),$ $(\mathcal{C}^4_{243},\mathcal{C}^4_{244}),(\mathcal{C}^4_{256},\mathcal{C}^4_{257}),(\mathcal{C}^4_{259},\mathcal{C}^4_{260}),(\mathcal{C}^4_{296},\mathcal{C}^4_{297}).$

non-completeness requirements and is rotation symmetric, by means of a method that we call the *direct sharing method*, and that we now describe for 3 shares. First, we replace every input variable by the sum of 3 shares. The correctness is satisfied if we ensure that

$$F_1 + F_2 + F_3 = F(x_1 + x_2 + x_3, y_1 + y_2 + y_3, z_1 + z_2 + z_3).$$

In order to satisfy non-completeness, we have to divide the terms of the right hand side over the three F_j in such a way that F_j does not contain the terms x_j, y_j and z_j where $j \in \{1, 2, 3\}$. We achieve this by assigning the linear terms containing an index j to F_{j-1}, the quadratic terms containing indices j and $j+1$ to F_{j-1} and the quadratic terms containing indices j only to F_{j-1}. For example,

$$\begin{aligned} F(x, y, z) &= x + yz, \\ &\text{gives:} \\ &F_1 = x_2 + y_2z_2 + y_2z_3 + y_3z_2 \\ &F_2 = x_3 + y_3z_3 + y_3z_1 + y_1z_3 \\ &F_3 = x_1 + y_1z_1 + y_1z_2 + y_2z_1. \end{aligned} \tag{2}$$

The method can easily be generalized for larger number of shares and higher degrees.

Note that the uniformity of a sharing produced in this way is not guaranteed. It has to be verified separately. On the other hand, it is enough to find a uniform direct sharing for one permutation within the class to judge if a permutation that belongs to the same class has a TI by Theorem 1. Therefore, we run an algorithm that goes through all permutations within a class to find a permutation that has direct uniform sharing. With this method using 3 shares, we were able to find TIs for permutations of $\mathcal{Q}^3_1$, but none of $\mathcal{Q}^3_2$ and $\mathcal{Q}^3_3$. We were also able to find TIs for permutations of $\mathcal{Q}^4_4$, $\mathcal{Q}^4_{294}$ and $\mathcal{Q}^4_{299}$, but none of $\mathcal{Q}^4_{12}$, $\mathcal{Q}^4_{293}$ and $\mathcal{Q}^4_{300}$. So, unfortunately half of the quadratic permutations does not have a TI when shared directly with 3 shares. We can increase the number of shares to find a uniform sharing for quadratic or cubic permutations. When we use 4 shares, we observe that all quadratic classes have at least one permutation that has a TI with direct sharing. We have also found TIs for the permutations in cubic classes $\mathcal{C}^4_1$, $\mathcal{C}^4_3$, $\mathcal{C}^4_{13}$ and $\mathcal{C}^4_{301}$ from $S_{16} \backslash A_{16}$ using direct sharing with 4 shares. On the other hand, with

5 shares we can find a TI for at least one permutation for all classes when shared directly which is a big improvement compared to the situation with 4 shares.

3.2 Sharing Using Correction Terms

Since direct sharing does not always result in a uniform sharing the use of *correction terms* (CT) has been proposed [38, 39]. Correction terms are terms that can be added in pairs to more than one share, such that they satisfy the non-completeness rule. Since the terms in a pair cancel each other, the sharing still satisfies the correctness.

By varying the CT one can obtain all possible sharings of a given function. Consider a Boolean quadratic function with m variables (1 output bit), which we want to share with 3 shares. Note that the only terms which can be used as CT without increasing the degree of the component functions are x_i or x_iy_i for $i = 1, 2, 3$. Indeed terms like x_iy_j for $i \neq j$ cannot be used in the i-th and j-th share of the function because of the non-completeness rule, i.e. such a term can be used in only 1 out of 3 shares, hence it cannot be used as a CT. We can also use higher degree terms such as $x_iy_iz_i \dots$ as CT if we do not limit the CT to the function degree.

Counting the linear, quadratic and cubic CT and ignoring the constant terms, which will not influence the uniformity, we obtain $3(m+\binom{m}{2}+\binom{m}{3})$ CT. Taking into account all possible positions for the CT we get $2^{3(m+\binom{m}{2}+\binom{m}{3})}$ different sharings. For example, for a quadratic function of 3 variables there are 2^{21} possible CT and therefore for a 3×3 S-box, the space of CT will be of size 2^{63}. This makes the exhaustive search (to find a single good solution) over the set of defined CT unpractical, even for small S-boxes. For a sharing with 4 shares even more terms can be used as CT. In [11], we left the problem of having an efficient algorithm to search through CT as an open question. Here we describe an algorithm that can provide a negative result if there does not exist any TI with correction terms, with a complexity less than the exhaustive search.

Consider a TI with the s-sharing $\mathbf{F}$ of an n-bit permutation F (if it exists). The uniformity property implies that the vectorial Boolean function $\mathbf{F} : \mathbb{F}_2^{ns} \to \mathbb{F}_2^{ns}, \mathbf{F} = (F_1, ..., F_s)$ is a balanced function. Recall some properties of the balanced vectorial functions [16].

Lemma 5 *Let $F = (f_1, ..., f_n)$ be a vectorial Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. F is a bijection if and only if for any k $(1 \leqslant k \leqslant n)$ and for any tuple of indices $i_1, ..., i_k$ $(1 \leqslant i_1 \leqslant \ldots \leqslant i_k \leqslant n)$ the vectorial Boolean function $(f_{i_1}, ..., f_{i_k})$ is balanced.*

Let $F = (f_1, ..., f_n)$ be a vectorial Boolean function such that $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $\mathbf{F} = (f_{11}, ..., f_{n1}, ..., f_{1s}, ..., f_{ns})$ be the direct sharing of F with s shares $F_i = (f_{1i}, ..., f_{ni})$. We say that the function $\mathbf{C_F} = (c_{11}, ..., c_{n1}, ..., c_{1s}, ..., c_{ns})$, where $c_{ij} : \mathbb{F}_2^{ns} \to \mathbb{F}_2$ are CT, is a *correction function* for $\mathbf{F}$, if the function $\mathbf{F} + \mathbf{C_F}$ satisfies all the properties of a TI.

Let $k \in \{1, \ldots, ns\}$ and let $(i_1j_1, ..., i_kj_k)$ be a k-tuple from the set $\{11, \ldots, n1, \ldots, 1s, \ldots, ns\}$. Denote the set

$$C^k_{i_1j_1,...,i_kj_k} = \{\mathbf{C_F} \mid (f_{i_1j_1}+c_{i_1j_1}, ..., f_{i_kj_k}+c_{i_kj_k}) \text{ is a balanced function from } \mathbb{F}_2^{ns} \text{ to } \mathbb{F}_2^k\}.$$

Then consider the set

$$\mathbf{C} = \bigcap_{k} \bigcap_{i_1j_1,\dots,i_kj_k} C^k_{i_1j_1,\dots,i_kj_k}.$$

Theorem 2 *The function* $\mathbf{F} + \mathbf{C_F}$ *is a bijection if and only if* $\mathbf{C_F} \in \mathbf{C}$.

Proof (Sufficient condition) By the definition of $\mathbf{C}$, for any function $\mathbf{C_F} \in \mathbf{C}$, the function $\mathbf{F} + \mathbf{C_F}$ is such that for any $k \in \{1, \dots, ns\}$ and for any tuple $(i_1j_1, ..., i_kj_k)$, from $\{11, \dots, n1, \dots, 1s, \dots, ns\}$ the vectorial sub-function $(f_{i_1j_1} + c_{i_1j_1}, ..., f_{i_kj_k} + c_{i_kj_k})$ is balanced. Hence, it follows from the lemma that the function $\mathbf{F} + \mathbf{C_F} = (f_{11} + c_{11}, ..., f_{n1} + c_{n1}, ..., f_{1s} + c_{1s}, ..., f_{ns} + c_{ns})$ is a bijection.

(Necessary condition) Let the function $\mathbf{F} + \mathbf{C_F}$ be a bijection. Then for any $k \in \{1, \dots, ns\}$, for any tuple $(i_1j_1, ..., i_kj_k)$ from the set $\{11, \dots, n1, \dots, 1s, \dots, ns\}$, the vectorial function $(f_{i_1j_1} + c_{i_1j_1}, ..., f_{i_kj_k} + c_{i_kj_k})$ is balanced. Hence, the function $\mathbf{C_F}$ belongs to the set $C^k_{i_1j_1,\dots,i_kj_k}$ by construction. Therefore, due to the arbitrariness of k and $(i_1j_1, ..., i_kj_k)$, the function $\mathbf{C_F}$ belongs to the set $\mathbf{C} = \bigcap_k \bigcap_{i_1j_1,\dots,i_kj_k} C^k_{i_1j_1,\dots,i_kj_k}$. □

This theorem allows us to use the following algorithm to search for a TI for the permutation $F = (f_1, ..., f_n)$ s.t. $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

```
Input: Direct sharing $\mathbf{F} = (f_{11}, ..., f_{1s}, ..., f_{n1}, ..., f_{ns})$ of $F$ with $s$ shares
       $F_i = (f_{1i}, ..., f_{ni})$ and the sets $J_k$ containing all $k$-tuples $(i_1j_1, ..., i_kj_k)$, from
       $\{11, \dots, n1, \dots, 1s, \dots, ns\}$ for $1 \le k \le ns$.
Output: The set $\mathbf{C}$ s.t. for each function $\mathbf{C_F} \in \mathbf{C}$ the sharing
       $F'_1 = (f_{11} + c_{11}, ..., f_{n1} + c_{n1}), ..., F'_s = (f_{1s} + c_{1s}, ..., f_{ns} + c_{ns})$ is uniform.
for $k = 1$ to $ns$ do
    while $J_k \neq \emptyset$ do
        Choose a tuple of indices $(i_1j_1, ..., i_kj_k) \in J_k$.
        Assign $J_k := J_k \backslash (i_1j_1, ..., i_kj_k)$.
        Construct the set
        $C^k_{i_1j_1,\dots,i_kj_k} = \{\mathbf{C_F} \mid (f_{i_1j_1} + c_{i_1j_1}, ..., f_{i_kj_k} + c_{i_kj_k})$ is balanced function$\}$.
        $\mathbf{C} := \bigcap_k \bigcap_{i_1j_1,\dots,i_kj_k} C^k_{i_1j_1,\dots,i_kj_k}$.
        if $\mathbf{C} \neq \emptyset$ then
            break;
        end
    end
    if $\mathbf{C} = \emptyset$ then
        break;
    end
end
```

With this algorithm, we can conclude that there does not exist a TI with correction terms if $\mathbf{C} = \emptyset$, which is likely to happen faster than the exhaustive search. Observe also that for any k, if the set $C^k_{i_1j_1,\dots,i_kj_k} = \emptyset$ then $\mathbf{C} = \emptyset$. Moreover, since the goal is to satisfy $\mathbf{C} \neq \emptyset$ for $k = ns$, we can start the *for* loop from any k s.t. $1 \le k \le ns$. If for any k, we have $\mathbf{C} \neq \emptyset$ at the end of the *while* loop, it implies that $\mathbf{C} \neq \emptyset$ for all $k' < k$.

We consider the permutation $F = (xy+yz+xz, x+y+xy+yz, x+z+yz)$ from the class $\mathcal{Q}^3_3$ and applied the above algorithm with initialization of $k = 4$. We choose the tuple of indices as $(11, 12, 13, 21) \in J_k$ and construct the set $C_1 = C^4_{11,12,13,21}$.

We obtaine that the set C_1 is empty. Therefore, the set C from the theorem is empty, hence, there is no uniform sharing for the given permutation. The algorithm terminated after a computation with complexity 2^{35}. This proves the following:

Corollary 1 *There does not exist a uniform sharing with 3 shares for permutations from $\mathcal{Q}_3^3$.*

Recall that by Lemma 3, the class $\mathcal{Q}_3^3$ corresponds to the class $\mathcal{Q}_{300}^4$. Moreover, if there is no uniform sharing for 3-bit permutations from a class, then there does not exist a uniform sharing for 4-bit permutations from the corresponding class.

Corollary 2 *There does not exist a uniform sharing with 3 shares for permutations from $\mathcal{Q}_{300}^4$.*

Recall that with a direct sharing, we could not find a uniform sharing for many classes. However by using CT and sharing linear and quadratic terms as in Equation 3, we found sharings for classes $\mathcal{Q}_2^3$, $\mathcal{Q}_{12}^4$, $\mathcal{Q}_{293}^4$ with three shares.

$$\begin{aligned} F(x,y,z) &= x + yz \\ &\text{gives:} \\ F_1 &= x_3 + z_2y_2 + z_2y_3 + z_3y_2 \\ F_2 &= x_1 + z_3y_3 + z_3y_1 + z_1y_3 \\ F_3 &= x_2 + z_1y_1 + z_1y_2 + z_2y_1. \end{aligned} \tag{3}$$

So all 3 and 4-bit quadratic classes except $\mathcal{Q}_3^3$ and $\mathcal{Q}_{300}^4$ have a TI with 3 shares.

3.3 Sharing Using Decomposition

In case we can not find a uniform sharing for a given S-box with a particular number of shares, one option to satisfy the uniformity property is to *re-mask* the output of the S-box as proposed by Moradi et al. [36]. That is, given $F(x, y, \ldots)$, its sharing $\mathbf{F} = (F_1, F_2, F_3)$ that is not uniform and two random masks m_1, m_2, the sharing **F'** provided in Equation 4 is uniform. This re-masking can be extended to any number of shares to generate uniform sharings when it is not possible to find uniform sharings otherwise. However, this should not be considered as a straight-forward approach since generation of good masks can be a burden.

$$\begin{aligned} F_1' &= F_1 + m_1 \\ F_2' &= F_2 + m_2 \\ F_3' &= F_3 + m_1 + m_2. \end{aligned} \tag{4}$$

Another option is to increase the number of shares. However, the area requirements of an implementation of a cryptographic algorithm increase with the number of shares, therefore it is desirable to keep the number of shares as low as possible. To overcome these issues, we can decompose an S-box into other S-boxes that have uniform sharings and/or lower degree. Examples of such decompositions have been presented for NOEKEON [39,40] and for PRESENT [43]. These three realizations decompose the S-box with algebraic degree three into two permutations with algebraic degree two. For the PRESENT S-box, decompositions $S(x) = F(G(x))$ with $G(0) = 0$ have been found [43] where $F(x)$ and $G(x)$ are quadratic permutations. By varying the constant term $G(0)$ the authors found all possible decompositions of $S(X) = F(G(X))$. Both S-boxes, $F(x)$ and $G(x)$, have been shared with direct sharing with 3 shares, (F_1, F_2, F_3) and (G_1, G_2, G_3), that are correct, non-complete and uniform. Moreover, the output of G is stored in the registers before it is taken as input to F to force the non-completeness property when the functions are cascaded as discussed in 2.3. Figure 1 illustrates this approach.

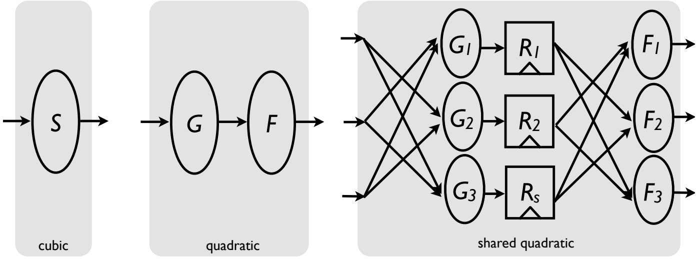

Fig. 1: Decomposition approach

Now we consider all 4-bit permutations, and investigate when a cubic permutation from S_{16} can be decomposed as a *composition of quadratic permutations.* We will refer to the minimum number of quadratic permutations in such a decomposition as *decomposition length.*

Lemma 6 *If a permutation S can be decomposed into a sequence of t quadratic permutations, then all permutations which are affine equivalent to S can be decomposed into a sequence of t quadratic permutations.*

Proof Let S be a cubic permutation which can be decomposed as a composition of quadratic permutations $Q_1 \circ Q_2 \circ \ldots \circ Q_{t-1} \circ Q_t$ with length t. Let W be a permutation which is affine equivalent to S. By definition, there exist affine permutations A and B s.t. $W = B \circ S \circ A$, therefore $W = B \circ Q_1 \circ Q_2 \circ \ldots \circ Q_{t-1} \circ Q_t \circ A$. Now, by defining two quadratic permutations $Q'_1 = B \circ Q_1$ and $Q'_t = Q_t \circ A$ we obtain that $W = Q'_1 \circ Q_2 \circ \ldots \circ Q_{t-1} \circ Q'_t$ has a decomposition with quadratic permutations and that its length is t. □

Lemma 7 *All 4-bit quadratic permutations belong to the alternating group A_{16}.*

Proof Since all affine permutations are in the alternating group (Lemma 1), two permutations which are affine equivalent, are either both even or both odd. We have taken one representative of each of the 6 quadratic classes $\mathcal{Q}_i^4$ for $i \in \{4, 12, 293, 294, 299, 300\}$ [21] and have verified that their parities are even. □

Now we investigate which permutations we can generate by combining the affine and the quadratic permutations. We start with the following lemma.

Lemma 8 *Let Q_i be 6 arbitrarily selected representatives of the 6 quadratic classes $\mathcal{Q}_i^4$ (hence $i \in \{4, 12, 293, 294, 299, 300\}$). Then all cubic permutations S that have decomposition length 2, are affine equivalent to one of the cubic permutation that can be written as*

$$\tilde{S}_{i \times j} = Q_i \circ A \circ Q_j, \tag{5}$$

where A is an affine permutation and $i, j \in \{4, 12, 293, 294, 299, 300\}$.

Proof Assume that $S = Q_a \circ Q_b$. Then we know that there are affine permutations A_a, B_a, A_b, B_b such that $S = (B_a \circ Q_i \circ A_a) \circ (B_b \circ Q_j \circ A_b)$, where Q_i, Q_j are two of the representatives defined above. We choose $A = A_a \circ B_b$ and $\tilde{S}_{i \times j} = {B_a}^{-1} \circ S \circ {A_b}^{-1}$. □

It follows that we can construct all cubic classes of decomposition length 2 by running through the 36 possibilities of $i \times j$ and the 322560 invertible affine transformations in Equation (5). This approach produces 30 cubic classes. In the remainder, we will denote the permutations $\tilde{S}_{i \times j}$ by $i \times j$ and refer to them as the *simple solutions*. Table 15 in the Appendix lists the simple solutions for all 30 decompositions with length 2. Note that if $Q_i \circ A \circ Q_j = S$, i.e. S can be decomposed as a product of $i \times j$, then $Q_j^{-1} \circ A^{-1} \circ Q_i^{-1} = S^{-1}$. Since for $n = 4$ all quadratics are affine equivalent to their inverse, it follows that S^{-1} is decomposed as a product of $j \times i$. Thus any self-inverse class has decomposition $i \times j$ and $j \times i$ as well. For the pairs of inverse classes we conclude that if $i \times j$ belongs to the first class then $j \times i$ belongs to the second class.

To obtain all decompositions with length 3 we use similar approach as for length 2 but the first permutation Q_i is cubic (instead of quadratic) and belongs to the already found list of cubic classes decomposable with length 2. It turns out that we can generate in this way the 114 remaining elements of A_{16}.

Summarizing, we can prove the following Theorem and Lemma (stated without proof in [22]).

Theorem 3 *A 4-bit permutation can be decomposed using 4-bit quadratic permutations if and only if it belongs to the alternating group A_{16} (151 classes).*

Proof (⇒) Let S be a permutation which can be decomposed with quadratic permutations say $Q_1 \circ Q_2 \circ \ldots \circ Q_t$. Since all $Q_i \in A_{16}$ (Lemma 7) and the alternating group is closed it follows that $S \in A_{16}$.
(⇐) Lemma 6, Lemma 8 and the discussion following it imply that we can generate all elements of the alternating group using quadratic permutations. □

The left-hand-side columns of Table 2 list the number of classes with a decomposition from a given length for all 4-bit permutations. Theorem 3 implies that the classes which are not in the alternative group i.e. in $S_{16} \setminus A_{16}$, cannot be decomposed as a product of quadratic classes. Now we make the following simple observation:

Lemma 9 *Let $\tilde{S}$ be a fixed permutation in $S_{16} \setminus A_{16}$ then any cubic permutation from $S_{16} \setminus A_{16}$ can be presented as a product of $\tilde{S}$ and a permutation from A_{16}.*

Proof By definition, all permutations in $S_{16} \setminus A_{16}$ are odd permutations, and if $\tilde{S} \in S_{16} \setminus A_{16}$, then $\tilde{S}^{-1} \in S_{16} \setminus A_{16}$. Since the product of two odd permutations is even, we have: $\forall S \in S_{16} \setminus A_{16} : S \circ \tilde{S}^{-1} \in A_{16}$. It follows that $\exists T \in A_{16} : S \circ \tilde{S}^{-1} = T$, i.e. $S = T \circ \tilde{S}$. □

In [29], Kutzner et al. introduce *factorization* instead of decomposition in order to enable 3-share TI for the invertible 4x4 S-boxes not belonging to A_{16}. In fact factorization uses a combination of decomposition and XOR of quadratic permutations. Although keeping three shares appears to be an interesting approach, the authors did not note the cost of their approach. When a single permutation is considered, the factorization method with 3 shares requires more registers than the 4-share approach for the permutations in $S_{16} \setminus A_{16}$. Implementations in [11] show that in certain cases it can be even more efficient to use 4 or 5-share TI instead of 3-share TI. In this paper we will not use the factorization approach.

Now that we know the possible decompositions of 3 and 4-bit permutations, we can try again to find TIs for these permutations by using less shares trading off with the decomposition length. However, this problem is more restrained than the basic problem, since we can use only the quadratic permutations for which we already have a uniform sharing. It turns out that the decompositions for $\mathcal{Q}^3_3$ are 1×2 and 2×1, i.e., we obtain a sharing with 3 shares for $\mathcal{Q}^3_3$ at the cost of decomposition length two (instead of length one). Similarly $\mathcal{Q}^4_{300}$ can be decomposed as 4×12, 4×293, 12×4, 12×294, 293×4, 293×294, 294×12 and 294×293. So, again we obtain a sharing with 3 shares with length two. With this result, we find TI for all 3 and 4-bit quadratic permutations with 3 shares.

$$
\begin{aligned}
&F(x,y,z) = x + yz + xyz \\
&\quad \text{gives:} \\
&\qquad F_1 = x_2 + (y_2+y_3+y_4)(z_2+z_3+z_4) \\
&\qquad\quad + (x_2+x_3+x_4)(y_2+y_3+y_4)(z_2+z_3+z_4) \\
&\qquad F_2 = x_3 + y_1(z_3+z_4) + z_1(y_3+y_4) + y_1z_1 + x_1(y_3+y_4)(z_3+z_4) \\
&\qquad\quad + y_1(x_3+x_4)(z_3+z_4) + z_1(x_3+x_4)(y_3+y_4) + x_1y_1(z_3+z_4) \\
&\qquad\quad + x_1z_1(y_3+y_4) + y_1z_1(x_3+x_4) + x_1y_1z_1 \\
&\qquad F_3 = x_4 + y_1z_2 + y_2z_1 + x_1y_1z_2 + x_1y_2z_1 + x_2y_1z_1 + x_1y_2z_2 + x_2y_1z_2 \\
&\qquad\quad + x_2y_1z_1 + x_1y_2z_4 + x_2y_1z_4 + x_1y_4z_2 + x_2y_4z_1 + x_4y_1z_2 + x_4y_2z_1 \\
&\qquad F_4 = x_1 + x_1y_2z_3 + x_1y_3z_2 + x_2y_1z_3 + x_2y_3z_1 + x_3y_1z_2 + x_3y_2z_1.
\end{aligned}
\tag{6}
$$

Recall that one can find decompositions into quadratic permutations for cubic permutations in the alternating group. Therefore these permutations have a TI with 3 shares. However the permutations outside the alternating group do not have a TI with three shares. By using Lemma 9 and Equation 3 or 6 (cubic term when necessary), we obtain a TI with 4 shares for all 4-bit permutations. The total length of the sharing depends on the class we use ($\mathcal{C}^4_1$, $\mathcal{C}^4_3$, $\mathcal{C}^4_{13}$ and $\mathcal{C}^4_{301}$) and also on the class from the alternating group, which is used for the decomposition. For example, class $\mathcal{C}^4_7$ can be decomposed using $\mathcal{C}^4_1$ with length four but with classes $\mathcal{C}^4_3$ and $\mathcal{C}^4_{13}$ it can be decomposed with length three. Note also that the number of solutions differs. We have found 10, 31 and 49 solutions when using $\mathcal{C}^4_1$, $\mathcal{C}^4_3$ and $\mathcal{C}^4_{13}$ classes, correspondingly. Unfortunately, we could not present the decompositions for all 4-bit permutations, which are provided in [10], in this paper due to page limitations. Table 2 summarizes these results.

Table 2: Overview of the numbers of classes of 4-bit permutations that can be decomposed and shared using 3 shares, 4 shares and 5 shares uniformly. The numbers are split up according to the decomposition length of the permutations (1, 2, 3, or 4), respectively their shares.

unshared			3 shares				4 shares			5 shares	remark
1	2	3	1	2	3	4	1	2	3	1	
6			5	1			6			6	quadratics
	30			28	2			30		30	cubics in A_{16}
		114			113	1			114	114	cubics in A_{16}
	–				–		4	22	125	151	cubics in $S_{16} \backslash A_{16}$

An open question is why for all 4-bit permutations, the TIs with 4 shares do not improve the results significantly compared to 3 shares and suddenly with 5 shares we can share all classes uniformly with length 1.

Recall that for the PRESENT S-box, decompositions $S(x) = F(G(x))$ have been found in [43]. The authors also made an observation that exactly $\frac{3}{7}$ sharings out of the decompositions automatically satisfy the uniformity condition (i.e. without any correction terms). Recall that with the direct sharing method without CT we (as well as the authors of [43]) were able to share only 3 quadratic classes: $\mathcal{Q}^4_4$, $\mathcal{Q}^4_{294}$ and $\mathcal{Q}^4_{299}$. The Present S-box belongs to $\mathcal{C}^4_{266}$ and has 7 simple solutions (see Table 15) but only 3 of them can be shared uniformly with direct sharing, namely 294 × 299, 299 × 294, 299 × 299, which explains the authors' observation.

In Tables 9–12, the column *Sharing* describes the length of the found TI with 3 and with 4 shares, separated by a comma. Since all classes can be shared with 5 shares uniformly with length 1 we omit this fact in these tables. Recall that for the permutations in $S_{16} \setminus A_{16}$ no solution with 3 shares exist which is indicated in the table by a $-$.

4 Permutations of Size $n = 5, 6$

S-boxes of size 5×5 and 6×6 have also been used in cryptographic primitives. One important example of a 5-bit S-box is the Keccak S-box [5], which has algebraic degree two. It has been shown in [7] that the direct sharing with 3 shares of this S-box does not satisfy the uniformity condition of a TI. Unfortunately, even with the algorithm provided in Section 3.2, it is not feasible to search through all possible CT to find a uniform sharing. On the other hand, TI with three shares with 4-bit extra fresh randomness per round or with four shares and CT is possible [7].

There are approximately 2^{61} (resp. 2^{215}) distinct affine equivalent classes of 5-bit (resp. 6-bit) permutations [21]. The set of almost bent permutations and almost perfect nonlinear permutations are the most well studied since they have a particular importance in cryptography.

Definition 2 ([17]) The permutation S is said to be *almost perfect nonlinear (APN)* if all the equations

$$S(x) + S(x + a) = b, \; a, b \in GF(2^n), a \neq 0,$$

have either 0 or 2 solutions.

Definition 3 ([17]) The permutation S is said to be *almost bent (AB)* if the numbers

$$\mu_S(a,b) = \sum_{x \in GF(2^n)} (-1)^{\langle b, S(x)\rangle + \langle a,x\rangle} \in V_m, a \neq 0,$$

are equal to either 0 or $\pm 2^{\frac{m+1}{2}}$ when $a, b \in GF(2^n)$ and $(a, b) \neq (0, 0)$.

It is known that all AB permutations are also APN. An APN permutation provides optimum resistance only against differential cryptanalysis. While an AB permutation provides optimum resistance against both differential and linear cryptanalysis [17]. However, AB permutations exist only when n is odd [17].

Up to affine equivalence there are only four AB permutations of dimension five. All can be represented as a power function [15]. A representative of each class is provided in Table 3. $AB4$ and $AB3$ are the inverse of $AB1$ and $AB2$, respectively.

Table 3: Representatives of AB permutations in $GF(2^5)$ [15].

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$AB1$	0	1	2	4	3	8	16	28	5	10	25	17	18	23	31	29	6
$AB2$	0	1	2	4	3	8	16	28	5	10	26	18	17	20	31	29	6
$AB3$	0	1	2	4	3	8	13	16	5	17	28	27	30	14	24	10	6
$AB4$	0	1	2	4	3	8	13	16	5	11	21	31	23	15	19	30	6
	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	deg.	pow.
$AB1$	20	13	24	19	11	9	22	27	7	14	21	26	12	30	15	2	x^3
$AB2$	21	24	12	22	15	25	7	14	19	13	23	9	30	27	11	2	x^5
$AB3$	19	11	20	31	29	12	21	18	26	15	25	7	22	23	9	3	x^7
$AB4$	28	29	9	24	27	14	18	10	17	12	26	7	25	20	22	3	x^{11}

Similar to the case of KECCAK S-box, with our current methods it is not feasible to find a TI with 3 shares for $AB1$ and $AB2$ even though they have algebraic degree two. However, it is possible to find TIs with 4 shares with CT for those permutations. The 4-share implementation for linear and quadratic terms are provided in Equation 6. Unfortunately, for $AB3$ and $AB4$, which have algebraic degree 3, with our current methods it is not feasible to find a uniform sharing up to five shares. However it is possible to have TIs with 4 shares with the cost of re-masking.

Up to affine equivalence, there is only one known APN permutation of dimension six [24]. Dillon shows in [24] that this APN permutation which has algebraic degree four, can be decomposed into two permutations of degree three and two. An example of an APN permutation with the decomposition $APN = F(G(x))$, where F has degree three and G has degree two is provided in Table 4. Unfortunately, with our current methods it is not feasible to find uniform sharing for F and G. However, with this decomposition, it is possible to have a 4-share implementation with re-masking.

As the S-box becomes bigger and more complicated, we observe that finding a uniform sharing becomes much more difficult and the search for CT becomes unfeasible. At this point, we leave the problem of finding TI for APN permutations of size six and AB permutations of degree three of size five as an open question.

Table 4: Representative of the known APN permutation in $GF(2^6)$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
APN	0	16	60	54	17	14	23	59	29	62	63	10	39	8	49	51
F	0	13	63	50	2	15	48	61	54	58	22	26	38	42	11	7
G	0	48	37	8	19	18	41	42	39	21	2	45	26	40	17	33
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
APN	45	37	61	48	47	5	12	20	36	57	40	46	26	56	43	55
F	46	49	14	17	33	62	12	19	24	6	39	57	5	27	55	41
G	32	60	7	6	51	28	22	59	43	27	61	16	11	57	46	30
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
APN	11	31	24	6	27	13	53	19	15	30	1	4	33	34	28	35
F	45	51	1	31	34	60	3	29	8	23	59	36	21	10	43	52
G	14	35	24	25	29	1	56	23	5	53	34	62	20	36	49	50
	48	49	50	51	52	53	54	55	56	57	58	9	60	61	62	63
APN	21	52	58	3	9	7	18	32	25	22	41	50	44	2	38	42
F	16	28	35	47	18	30	44	32	53	56	25	20	37	40	4	9
G	44	47	9	38	63	15	52	55	58	10	31	3	54	4	12	13

The only known examples of using AB permutation of size five or APN permutation of size six are the authenticated encryption algorithms FIDES [6] (AB1 and APN) and PRIMATEs [2] (AB1), which are designed to provide provable security against first-order side-channel analysis attacks.

Another example of using S-boxes with 6-bit inputs is the Data Encryption Standard (DES) [23]. A 6×4 S-box used by DES can be implemented as four 4×4 invertible S-boxes followed by a cubic selection function (4-to-1 multiplexer). The selection function can be implemented uniformly with direct sharing with 4 shares and we have TI for all 4×4 invertible S-boxes. However, the distribution of the input of the selection function, which combines the outputs of 4×4 S-boxes as input is not necessarily uniform. To avoid first-order leakage, the input of the selection function should be checked carefully and should be re-masked when necessary.

5 Hardware Implementations

Our aim is to provide a fair comparison and prediction what the cost (ratio of area to a NAND gate referred to as GE) is for a TI of a permutation in a specified library. For our investigations we use the open cell library NANGATE [37] and the Synopsis development tool. We used *compile ultra* command to optimize each component function (resp. unshared function) then combined these to get the cost of TI (resp. permutation).

The area results listed consider a set of input or output registers. The cost of a permutation is the cost of the combinational logic required to calculate the (shared) permutation, the cost of the pipelining registers if it is required (i.e. if it is decomposed) as described in Section 3.3 and the cost of the n-bit (resp. $n \times s$-bit if shared) register. The formulas for 3 shares are generated using the Equations 2 and 3. For 4 shares, we used a variation of the same equations for quadratic functions. For cubic permutations with 4 shares and for all permutations with 5 shares, we used a variation of the Equation 6.

For 3-bit permutations, we choose a permutation randomly from each class and observed the cost as shown in Table 5. We observe that for these small permutations, the sharing used has a big effect on area. That is why 4-share implementations that use the Equation 3 are bigger than the 5-share implementations that use the Equation 6 even though they require less registers.

Table 5: Area comparison for randomly selected quadratic permutations in S_{16}

3-bit	Sharing	Unshared		Shared		
Permutations	Length	Original	Decomp.	3 shares	4 shares	5 shares
Class #	(L)	1 reg	L reg	L reg	1 reg	1 reg
$\mathcal{Q}_1^3$	1	23	-	120	189	176
$\mathcal{Q}_2^3$	1	24	-	129	193	184
$\mathcal{Q}_3^3$	2	24	67	243	196	190

Since the wide-range of S-boxes used in cryptography are from the set of 4-bit permutations, we deepen our research for these permutations. Moreover, since we use quadratic or cubic classes with length 1 for the decompositions, we concentrated our efforts on these classes and implemented 1000 permutations chosen randomly per each class. Area distributions in Figure 2 and the average areas in Table 6 show that it is advantageous to use permutations from $\mathcal{Q}_4^4$ and avoid using permutations from $\mathcal{Q}_{299}^4$ if possible for a more area efficient implementation.

A similar argument can also be made for cubic permutations with decomposition length 1 with 4 shares. Namely, by Figure 3 and Table 6, we can conclude that it is more efficient to use $\mathcal{C}_1^4$ especially compared to $\mathcal{C}_{301}^4$ where the difference becomes more visible.

For classes with decomposition length more than one (Table 7), we randomly select a class representative i.e. a permutation. Then we implement the smallest amongst all possible decompositions of this permutation, namely the one which gives minimum GE. We saw that, classes $\mathcal{Q}_3^3$, $\mathcal{Q}_{300}^4$, $\mathcal{C}_{150}^4$, $\mathcal{C}_{151}^4$, $\mathcal{C}_{130}^4$, $\mathcal{C}_{131}^4$, $\mathcal{C}_{24}^4$, $\mathcal{C}_{204}^4$, $\mathcal{C}_{257}^4$ and $\mathcal{C}_{210}^4$ give relatively small results when implemented as 2×1, 12×4, 12×293, 293×12, $12 \times 4 \times 299$, $299 \times 12 \times 4$, $299 \times 12 \times 4 \times 299$, 3×294, 3×12 and $3 \times 293 \times 12$ respectively. Observe that we use permutations from $\mathcal{Q}_{299}^4$ to decompose permutations from $\mathcal{C}_{130}^4$, $\mathcal{C}_{131}^4$ and $\mathcal{C}_{24}^4$ since there does not exist a decomposition with the same length that does not require a permutation from $\mathcal{Q}_{299}^4$. On the other hand, several possibilities for decomposing a permutation $\mathcal{Q}_{300}^4$ exist as described in Section 3.3 and the smallest decomposition we find also matches with the findings in Figure 2.

Table 7 also shows that depending on the decomposition length and the permutation class, a 5 share TI can give smaller results than a 3 or 4 share TI if only the S-box is considered. The optimal invertible S-boxes used in cryptography have

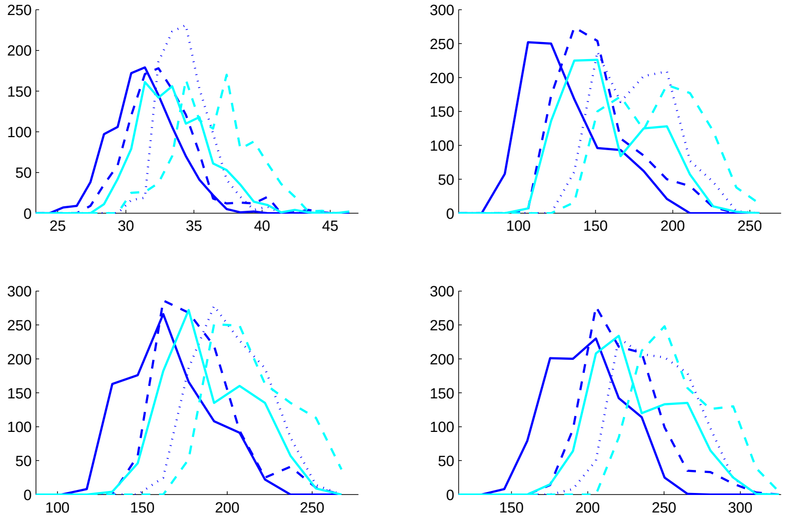

Fig. 2: Area (x-axis) distribution of permutations from Q^4_4 (—), Q^4_{12} (- -), Q^4_{293} (..), Q^4_{294} (—), Q^4_{299} (- -) with unshared (top, left), 3-share TI (top, right), 4-share TI (bottom, left) and 5-share TI (bottom, right) where y-axis refers to the number of permutations.

decomposition lengths (2,2), (3,3) or (-,3) for which an estimation can be deduced from the same table.

Remember that we cannot find a TI for cubic 5-bit AB permutations with four shares. Therefore we chose to implement only the quadratic AB permutations. For each of the class we randomly select a representative to implement. We observe that TI cost is a little bit over four times the cost of the unshared version.

On the other hand, we implement the APN permutation in Table 4 with the given decomposition even though the sharing is not uniform since that is the only known class. Therefore for that implementation, an extra cost of re-masking should be calculated. We observe that even the unshared implementation cost is high as a result of the high degree and with decomposition this cost might reduce. This 4-share implementation is $3,5$ times the cost of the unshared version. We summarize the results for permutations of size five and six in Table 8.

6 Extensions

In the basic approach of TI, the number of input and output shares and variables are the same. In this section, we propose two extensions to the basic approach.

Table 6: Average area comparison for quadratic permutations in A_{16} and cubic permutations in $S_{16}\backslash A_{16}$ which have decomposition length 1 with 3 and 4 shares respectively.

4-bit Permutations	Original	Shared		
Class #	S-box	3 shares	4 shares	5 shares
$\mathcal{Q}^4_4$	31	131	165	199
$\mathcal{Q}^4_{12}$	32	151	182	223
$\mathcal{Q}^4_{293}$	34	176	182	244
$\mathcal{Q}^4_{294}$	33	159	191	233
$\mathcal{Q}^4_{299}$	36	190	216	259
$\mathcal{C}^4_1$	31	-	254	322
$\mathcal{C}^4_3$	33	-	291	361
$\mathcal{C}^4_{13}$	34	-	285	349
$\mathcal{C}^4_{301}$	36	-	298	360

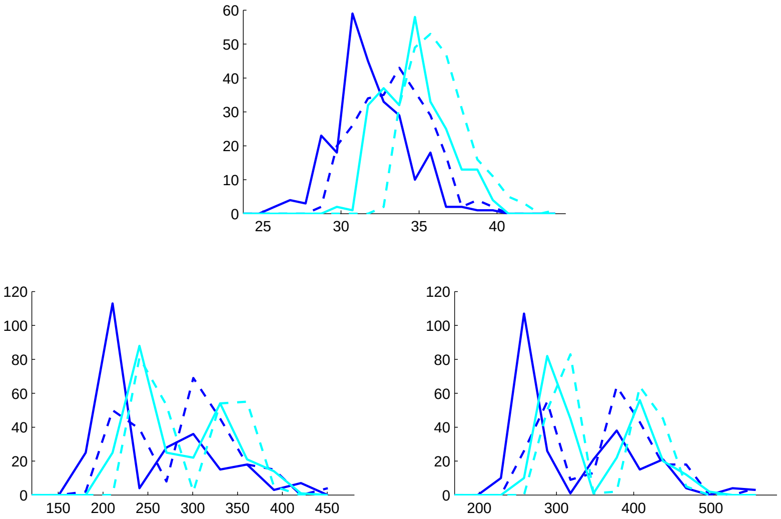

Fig. 3: Area (x-axis) distribution of permutations from C^4_1 (—), C^4_3 (- -), C^4_{13} (—), C^4_{301} (- -) with unshared (top), 4-share TI (bottom, left) and 5-share TI (bottom, right) where y-axis refers to the number of permutations.

6.1 Virtual Variables and Virtual Shares

For some Boolean functions with two inputs there is no sharing with three shares satisfying all TI requirements [38,39]. For example, direct sharing of multiplication of two variables with three shares does not satisfy the uniformity condition. On

Table 7: Area comparison for randomly selected quadratic and cubic permutations in A_{16} and cubic permutations in $S_{16} \backslash A_{16}$ which have decomposition more than one for 3 and 4 shares respectively.

4-bit	Sharing	Unshared		Shared		
Permutations	Length	Original	Decomp.	3 shares	4 shares	5 shares
Class #	(L, L')	1 reg	(L, L') reg	L reg	L' reg	1 reg
$\mathcal{Q}^4_{300} \in S_{16}$	2, 1	45	63, 41	301	266	314
$\mathcal{C}^4_{150} \in S_{16}$	2, 2	38	66, 66	284	408	399
$\mathcal{C}^4_{151} \in S_{16}$	2, 2	38	64, 64	267	396	490
$\mathcal{C}^4_{130} \in S_{16}$	3, 2	40	88, 65	375	360	506
$\mathcal{C}^4_{131} \in S_{16}$	3, 2	43	88, 62	370	404	517
$\mathcal{C}^4_{24} \in S_{16}$	4, 3	41	126,102	627	678	524
$\mathcal{C}^4_{204} \in S_{16} \backslash A_{16}$	-, 2	39	-, 65	-	495	466
$\mathcal{C}^4_{257} \in S_{16} \backslash A_{16}$	-, 2	41	-, 67	-	498	492
$\mathcal{C}^4_{210} \in S_{16} \backslash A_{16}$	-, 3	38	-, 115	-	750	518

Table 8: Quadratic AB and APN S-boxes sharing

Permutation	Sharing	Unshared		Shared
	Length	Original	Decomposed	4 shares
Class	(L)	1 reg	L reg	L reg
$AB1$	1	68	-	303
$AB2$	1	64	-	274
APN	2	224	192	795

the other hand, TI with three shares do exist for all quadratic Boolean functions with three inputs. This fact leads to an approach where we define extra input variables, *virtual variables* for the function that we want to find a sharing for. A *virtual variable* is hence an additional input to the function, whose value does not influence the output of the function. In the implementation however, it must be ensured that the attacker cannot predict the value of the virtual variable, i.e. it has to be random. Hence, the approach requires additional randomness as input. For example, assume that we want to construct a sharing for the function $F(x, y) = xy$. By adding a virtual variable z, we can share $F(x, y, z) = xy$ as follows:

$$\begin{aligned}
F_1 &= x_2y_2 + x_2y_3 + x_3y_2 + x_2z_2 + x_3z_3 + y_2z_2 + y_3z_3 \\
F_2 &= x_3y_3 + x_1y_3 + x_3y_1 + x_3z_3 + x_1z_1 + y_3z_3 + y_1z_1 \\
F_3 &= x_1y_1 + x_1y_2 + x_2y_1 + x_1z_1 + x_2z_2 + y_1z_1 + y_2z_2.
\end{aligned} \tag{7}$$

Without the virtual variable, we can share the product of two variables if we use four shares [38], hence in total $2 \times 4 = 8$ elements. With virtual variable, we obtain $3 \times 3 = 9$ elements, which is in fact not an improvement.

Since z in the previous example $F = xy$ was a virtual variable, its shares z_1, z_2 and z_3 can be called *virtual shares*. Instead of introducing all the 3 virtual shares, we can also introduce fewer of them. Since a virtual share is not related to a 'real' input of the function, it does not need to be taken into account when we check the non-completeness of the sharing. The previous example can be shared using only one virtual share as:

$$\begin{aligned} F_1 &= x_2y_2 + x_2y_3 + x_3y_2 + z \\ F_2 &= x_3y_3 + x_1y_3 + x_3y_1 + x_1z + y_1z \\ F_3 &= x_1y_1 + x_1y_2 + x_2y_1 + x_1z + y_1z + z. \end{aligned} \tag{8}$$

An alternative way to satisfy the uniformity is re-masking as described in Equation 4. If we consider the term x_1z+y_1z in Equation 8 as a second mask, this equation becomes equivalent to the Equation 4 where we use 8 elements. Therefore, using virtual variables can also be considered as a clever way of re-masking that requires only 7 elements. This improvement in the number of elements is very small when only one multiplication is considered. On the other hand, a cryptographic algorithm typically use much more than one multiplication.

6.2 Varying the Number of Shares

Until now we have considered the case when the inputs and the outputs of a function have to be shared with the same number of shares, e.g., s. In fact, it is possible to generalize the approach, such that the inputs are shared with s_i shares, the outputs (i.e., the function) with s_o shares providing that $s_i \geq s_o$ holds. We will shortly illustrate this approach by sharing the product xy, such that the input is shared with 4 shares and the output with 3 shares.

$$\begin{aligned} F_1 &= (x_2 + x_3 + x_4)(y_2 + y_3) + y_4 \\ F_2 &= (x_1 + x_3)(y_1 + y_4) + x_1y_3 + x_4 \\ F_3 &= (x_2 + x_4)(y_1 + y_4) + x_1y_2 + x_4 + y_4. \end{aligned}$$

This approach also gives flexibility when several blocks are combined with each other. An example of varying the number of shares is used in for implementing the multiplication in $GF(2^4)$ in the AES S-box [8].

7 Conclusions

We have considered the threshold implementation method, which is a method to construct implementations of cryptographic functions that are secure against a large class of side-channel attacks, even when the hardware technology is not glitch-free.

We have analyzed which basic permutations can be securely implemented using 3, 4 or 5 shares. We have constructed TIs for all 3 and 4-bit permutations. Moreover, we showed that 5-bit quadratic almost bent permutations can be implemented with 4 shares. Thus we have extended the threshold implementation method to secure implementations for any cryptographic algorithm which uses these permutations as S-boxes. Note that in many block ciphers the mixing layer of its round function is a linear operation and thus it is trivially shared even with

2 shares. Finally, we have implemented several of these shared S-boxes in order to investigate the cost of the sharing as well as the additional cost due to the pipelining stages separated by latches or registers.

Our results show that such secure implementations can also be efficient. As expected, the longer length a sharing has, the more costly it becomes (for 3 and 4 shares). Even though, there are several cases when using 5 shares reduces the cost by up to 25% compared to 3 and 4 shares with longer sharing length, in general 5 shares are about 30% more expensive than 4 shares.

An obvious conclusion is that the cost of the TI method heavily depends on the class the given S-box belongs to as well as the chosen number of shares and the associated sharing length. Therefore, in order to minimize the implementation cost the number of shares have to be carefully chosen.

We leave as an open question the existence of a uniform sharing of 5-bit quadratic AB S-boxes with 3 shares, cubic AB S-boxes with 4 shares and 6-bit APN S-box. A SW toolkit, which implements these techniques described in this paper can be found in [9].

Acknowledgements

We would like to thank the reviewers for their detailed comments, Christophe De Cannière for the fruitful discussions and for sharing part of his toolkit for affine equivalent classes with us, Georg Stütz for contributing to the proof of Theorem 1 and Anastasiya Gorodilova for kind assistance with APN permutations.

This work has been supported in part by the Research Council of KU Leuven (OT/13/071), B. Bilgin was partially supported by the FWO project G0B4213N, V. Nikov was supported by the European Commission (FP7) within the Tamper Resistant Sensor Node (TAMPRES) project with contract number 258754 and N. Tokareva and V. Vitkup were supported by the Russian Foundation for Basic Research (project 12–01–31097) and by Grant NSh–1939.2014.1 of President of Russia for Leading Scientific Schools.

References

1. Mehdi-Laurent Akkar and Christophe Giraud. An implementation of DES and AES, secure against some attacks. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer Berlin Heidelberg, 2001.
2. Elena Andreeva, Atul Luykx Florian Mendel Bart Mennink Nicky Mouha Qingju Wang Begül Bilgin, Andrey Bogdanov, and Kan Yasuda. PRIMATEs. Available at `http://competitions.cr.yp.to/round1/primatesv1.pdf`.
3. Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24(2):269–291, 2011.
4. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Building power analysis resistant implementations of Keccak. Second SHA-3 candidate conference, August 2010.
5. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, January 2011.
6. Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In Guido Bertoni and Jean-Sebastien Coron, editors, *Cryptographic Hardware and*

Embedded Systems - CHES 2013, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer Berlin Heidelberg, 2013.

7. Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order DPA resistant implementations of KECCAK. to appear in CARDIS 2013, 2013.
8. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A more efficient AES threshold implementation. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer International Publishing, 2014.
9. Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. TI toolkit, 2013. `http://homes.esat.kuleuven.be/~snikova/ti_tools.html`.
10. Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. List of decompositions of 4-bit permutations, 2014. `http://homes.esat.kuleuven.be/~bbilgin/other.html`.
11. Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold implementations of all 3x3 and 4x4 s-boxes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer Berlin Heidelberg, 2012.
12. Alex Biryukov, Christophe De Cannièere, An Braeken, and Bart Preneel. A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 33–50, Berlin, Heidelberg, 2003. Springer-Verlag.
13. Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably secure masking of AES. In Helena Handschuh and M.Anwar Hasan, editors, *Proceedings of the 11th International Conference on Selected Areas in Cryptography - SAC'04*, Lecture Notes in Computer Science, pages 69–83, Berlin, Heidelberg, 2005. Springer-Verlag.
14. Christina Boura and Anne Canteaut. On the influence of the algebraic degree of f^{-1} on the algebraic degree of $g \circ f$. Cryptology ePrint Archive, Report 2011/503, 2011. `http://eprint.iacr.org/`.
15. Marcus Brinkmann and Gregor Leander. On the classification of APN functions up to dimension five. *Designs, Codes and Cryptography*, 49(1-3):273–288, 2008.
16. Claude Carlet. Vectorial boolean functions for cryptography. In Yves Crama and Peter L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge University Press, New York, NY, USA, 2010.
17. Claude Carlet, Pascale Charpin, and Victor Zinoviev. Codes, bent functions and permutations suitable for DES-like cryptosystems. *Designs, Codes and Cryptography*, 15(2):125–156, November 1998.
18. Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer Berlin Heidelberg, 2014.
19. Joan Daemen, René Govaerts, and Joos Vandewalle. A new approach to block cipher design. In Ross Anderson, editor, *Fast Software Encryption*, volume 809 of *Lecture Notes in Computer Science*, pages 18–32. Springer Berlin Heidelberg, 1994.
20. Joan Daemen, Michael Peeters, and Gilles Assche. Bitslice ciphers and power analysis attacks. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 134–149. Springer Berlin Heidelberg, 2001.
21. Christophe De Canniere. *Analysis and Design of Symmetric Encryption Algorithms*. PhD thesis, 2007.
22. Christophe De Canniere, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. S-box decompositions for SCA-resisting implementations, 2012. Poster presented at CHES 2011, Nara, Japan.
23. DES. Data encryption standard. In *In FIPS PUB 46, Federal Information Processing Standards Publication*, pages 46–2, 1977.
24. John F. Dillon. APN polynomials: An update, 2009.
25. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer Berlin Heidelberg, 2014.
26. Jovan D. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In Burton S. Kaliski, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer Berlin Heidelberg, 2003.

27. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer Berlin Heidelberg, 2003.
28. Lars Knudsen, Gregor Leander, Axel Poschmann, and Matthew J.B. Robshaw. PRINTcipher: A block cipher for IC-printing. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 16–32. Springer Berlin Heidelberg, 2010.
29. Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann. Enabling 3-share threshold implementations for any 4-bit S-box. Cryptology ePrint Archive, Report 2012/510, 2012. `http://eprint.iacr.org/`.
30. Gregor Leander and Axel Poschmann. On the classification of 4 bit S-Boxes. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer Berlin Heidelberg, 2007.
31. Rudolf Lidl and Harald Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1997.
32. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
33. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer Berlin Heidelberg, 2005.
34. Amir Moradi. Statistical tools flavor side-channel collision attacks. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2012.
35. Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-enhanced power analysis collision attack. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.
36. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In KennethG. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer Berlin Heidelberg, 2011.
37. NANGATE. The NanGate 45nm Open Cell Library. Available at `http://www.nangate.com`.
38. Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer Berlin Heidelberg, 2006.
39. Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of non-linear functions in the presence of glitches. In Pil Joong Lee and Jung Hee Cheon, editors, *Information Security and Cryptology - ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 218–234. Springer Berlin Heidelberg, 2009.
40. Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24:292–321, 2011.
41. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A side-channel analysis resistant description of the AES s-box. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer Berlin Heidelberg, 2005.
42. Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer Berlin Heidelberg, 2005.
43. Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2,300 GE. *Journal of Cryptology*, 24(2):322–345, 2011.
44. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems - CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer Berlin Heidelberg, 2010.

45. Joseph Jonah Rotman. *An introduction to the theory of groups.* Springer-Verlag, New York; Berlin; Heidelberg [etc.], 1999.
46. Markku-JuhaniO. Saarinen. Cryptographic analysis of all 4 x 4-bit s-boxes. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 118–133. Springer Berlin Heidelberg, 2012.
47. Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '04, pages 10246–, Washington, DC, USA, 2004. IEEE Computer Society.
48. Elena Trichina, Tymur Korkishko, and KyungHee Lee. Small size, low power, side channel-immune AES coprocessor: Design and synthesis results. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 113–127. Springer Berlin Heidelberg, 2005.
49. Ralph Wernsdorf. The round functions of rijndael generate the alternating group. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Science*, pages 143–148. Springer Berlin Heidelberg, 2002.

A Appendix - Tables

Table 9: The 4 classes of 3-bit permutations

Class	Truth table	Sharing
$\mathcal{A}_0^3$	01234567	1,1
$\mathcal{Q}_1^3$	01234576	1,1
$\mathcal{Q}_2^3$	01234675	1,1
$\mathcal{Q}_3^3$	01243675	2,2

Table 10: The 302 classes of 4-bit permutations

Class	Truth table	Sharing	Class	Truth table	Sharing
$\mathcal{A}_0^4$	0123456789ABCDEF	1,1	$\mathcal{C}_{36}^4$	0123456879CEAFBD	3,3
$\mathcal{C}_1^4$	0123456789ABCDFE	-,1	$\mathcal{C}_{37}^4$	0123456879ACDEFB	-,3
$\mathcal{C}_2^4$	0123456789ABCEFD	3,3	$\mathcal{C}_{38}^4$	0123456879ABDEFC	3,3
$\mathcal{C}_3^4$	0123456789ABDEFC	-,1	$\mathcal{C}_{39}^4$	012345768A9CBEFD	-,3
$\mathcal{Q}_4^4$	0123456789ABDCFE	1,1	$\mathcal{C}_{40}^4$	012345768A9CBFDE	-,2
$\mathcal{C}_5^4$	0123456789ACDBFE	-,2	$\mathcal{C}_{41}^4$	012345768A9CBFED	3,3
$\mathcal{C}_6^4$	0123456789ACBDFE	3,3	$\mathcal{C}_{42}^4$	012345786ACBED9F	-,3
$\mathcal{C}_7^4$	0123456789ACBEFD	-,3	$\mathcal{C}_{43}^4$	012345786ABCF9DE	3,3
$\mathcal{C}_8^4$	0123456789ACDEFB	3,3	$\mathcal{C}_{44}^4$	012345786AC9BFED	3,3
$\mathcal{C}_9^4$	0123456789ACDEBF	-,3	$\mathcal{C}_{45}^4$	012345786A9CFBDE	-,3
$\mathcal{C}_{10}^4$	0123456789BCAEFD	3,3	$\mathcal{C}_{46}^4$	012345786ABCDEF9	3,3
$\mathcal{C}_{11}^4$	0123456789BCEFDA	-,2	$\mathcal{C}_{47}^4$	012345786AC9DEBF	-,3
$\mathcal{Q}_{12}^4$	0123456789CDEFAB	1,1	$\mathcal{C}_{48}^4$	012345786AC9EDFB	-,3
$\mathcal{C}_{13}^4$	0123456789CDEFBA	-,1	$\mathcal{C}_{49}^4$	012345786A9CDEBF	3,3
$\mathcal{C}_{14}^4$	0123456879CDEFBA	3,3	$\mathcal{C}_{50}^4$	012345786A9CFDBE	3,3
$\mathcal{C}_{15}^4$	012345687A9CBEFD	-,3	$\mathcal{C}_{51}^4$	012345786ABCDE9F	-,3
$\mathcal{C}_{16}^4$	012345687A9CDFBE	3,3	$\mathcal{C}_{52}^4$	012345786ACBDE9F	3,3
$\mathcal{C}_{17}^4$	0123456879CDEFAB	-,2	$\mathcal{C}_{53}^4$	012345786ACBDFE9	3,3
$\mathcal{C}_{18}^4$	0123456879ACDBFE	3,3	$\mathcal{C}_{54}^4$	012345786A9BCEFD	-,2
$\mathcal{C}_{19}^4$	0123456879ACDFBE	-,3	$\mathcal{C}_{55}^4$	012345786AB9CFDE	3,3
$\mathcal{C}_{20}^4$	0123456879ACDEBF	3,3	$\mathcal{C}_{56}^4$	012345786AC9BFDE	-,3
$\mathcal{C}_{21}^4$	0123456879ACBDFE	-,3	$\mathcal{C}_{57}^4$	012345786A9CBEFD	3,3
$\mathcal{C}_{22}^4$	0123456879ACFEDB	3,3	$\mathcal{C}_{58}^4$	012345786ACFDE9B	-,3
$\mathcal{C}_{23}^4$	0123456879BCEFAD	-,3	$\mathcal{C}_{59}^4$	012345786ACEDFB9	-,2
$\mathcal{C}_{24}^4$	012345687A9CFBDE	4,3	$\mathcal{C}_{60}^4$	012345786ACFB9DE	3,3
$\mathcal{C}_{25}^4$	0123456879ABCEFD	-,3	$\mathcal{C}_{61}^4$	012345786ACFDEB9	3,3
$\mathcal{C}_{26}^4$	0123456879BCDEFA	3,3	$\mathcal{C}_{62}^4$	012345786A9CBFED	-,3
$\mathcal{C}_{27}^4$	012345687ABCDEF9	-,3	$\mathcal{C}_{63}^4$	012345786AC9DEFB	3,3
$\mathcal{C}_{28}^4$	0123456879BCEAFD	3,3	$\mathcal{C}_{64}^4$	012345786ABCED9F	3,3
$\mathcal{C}_{29}^4$	012345687ABCEFD9	-,3	$\mathcal{C}_{65}^4$	012345786A9CFDEB	-,3
$\mathcal{C}_{30}^4$	012345687ABCE9FD	-,3	$\mathcal{C}_{66}^4$	012345786ACB9EFD	3,3
$\mathcal{C}_{31}^4$	0123456879ACBEFD	3,3	$\mathcal{C}_{67}^4$	012345786ACF9DBE	3,3
$\mathcal{C}_{32}^4$	0123456879ACFBDE	-,3	$\mathcal{C}_{68}^4$	0123457869ACDFEB	-,3
$\mathcal{C}_{33}^4$	0123456879BCEFDA	3,3	$\mathcal{C}_{69}^4$	0123457869ACDEBF	-,3
$\mathcal{C}_{34}^4$	0123456879BCFEAD	3,3	$\mathcal{C}_{70}^4$	012345786ACBF9ED	3,3
$\mathcal{C}_{35}^4$	0123456879CEAFDB	-,3	$\mathcal{C}_{71}^4$	012345786ACEBD9F	3,3

Table 11: The 302 classes of 4-bit permutations

Class	Truth table	Sharing	Class	Truth table	Sharing
$\mathcal{C}^4_{72}$	012345786ACDF9EB	-,3	$\mathcal{C}^4_{126}$	012345786AC9EDBF	3,3
$\mathcal{C}^4_{73}$	012345786ACDF9BE	3,3	$\mathcal{C}^4_{127}$	012345786ABC9FED	3,3
$\mathcal{C}^4_{74}$	012345786ACDE9FB	3,3	$\mathcal{C}^4_{128}$	0123458A6B9CDE7F	-,2
$\mathcal{C}^4_{75}$	012345786AC9FBED	-,3	$\mathcal{C}^4_{129}$	0123458A6BC7F9ED	-,3
$\mathcal{C}^4_{76}$	012345786ACEBFD9	3,3	$\mathcal{C}^4_{130}$	0123458A6CBDE79F	3,2
$\mathcal{C}^4_{77}$	012345786A9CEFDB	-,3	$\mathcal{C}^4_{131}$	0123458A6CE9BDF7	3,2
$\mathcal{C}^4_{78}$	0123457869ACBEDF	3,3	$\mathcal{C}^4_{132}$	0123458A6CBD7E9F	-,3
$\mathcal{C}^4_{79}$	0123457869ACBFDE	-,3	$\mathcal{C}^4_{133}$	0123458A6C9FBD7E	-,3
$\mathcal{C}^4_{80}$	0123457869ACBEFD	-,3	$\mathcal{C}^4_{134}$	0123458A69C7DEBF	3,3
$\mathcal{C}^4_{81}$	0123457869ACEFDB	3,3	$\mathcal{C}^4_{135}$	0123458A69CDE7FB	-,3
$\mathcal{C}^4_{82}$	0123457869ACEBDF	-,3	$\mathcal{C}^4_{136}$	0123458A69C7FBED	3,3
$\mathcal{C}^4_{83}$	0123457869ACEBFD	3,3	$\mathcal{C}^4_{137}$	0123458967CEAFBD	-,3
$\mathcal{C}^4_{84}$	012345786ACF9EBD	-,3	$\mathcal{C}^4_{138}$	0123458967CEAFDB	3,3
$\mathcal{C}^4_{85}$	012345786A9CEBDF	3,3	$\mathcal{C}^4_{139}$	0123456879BCAEFD	-,3
$\mathcal{C}^4_{86}$	012345786A9CFBED	3,3	$\mathcal{C}^4_{140}$	012345687ABC9FDE	3,3
$\mathcal{C}^4_{87}$	012345786ACD9EFB	-,3	$\mathcal{C}^4_{141}$	0123458967CEBFDA	-,3
$\mathcal{C}^4_{88}$	012345786ACD9FBE	-,2	$\mathcal{C}^4_{142}$	012345786ACD9FEB	3,3
$\mathcal{C}^4_{89}$	012345786ACD9EBF	3,3	$\mathcal{C}^4_{143}$	0123458A69CFB7DE	-,3
$\mathcal{C}^4_{90}$	012345786ABCF9ED	-,3	$\mathcal{C}^4_{144}$	0123458A69CFDEB7	-,3
$\mathcal{C}^4_{91}$	012345786ACFBD9E	-,3	$\mathcal{C}^4_{145}$	0123458A69BCF7ED	3,3
$\mathcal{C}^4_{92}$	012345786ABC9EDF	3,3	$\mathcal{C}^4_{146}$	0123458A69CB7FDE	-,3
$\mathcal{C}^4_{93}$	012345786ABC9EFD	-,3	$\mathcal{C}^4_{147}$	012345786ABCFDE9	3,3
$\mathcal{C}^4_{94}$	012345786ACED9FB	-,3	$\mathcal{C}^4_{148}$	012345786ABCE9FD	3,3
$\mathcal{C}^4_{95}$	012345786A9CDFEB	3,3	$\mathcal{C}^4_{149}$	012345786ABCFD9E	-,3
$\mathcal{C}^4_{96}$	012345786A9CEDFB	3,3	$\mathcal{C}^4_{150}$	0123458A6BCFDE97	2,2
$\mathcal{C}^4_{97}$	0123458A6BCEDF97	-,3	$\mathcal{C}^4_{151}$	0123458A6BCF97DE	2,2
$\mathcal{C}^4_{98}$	0123458A6BCF97ED	-,3	$\mathcal{C}^4_{152}$	0123458A6BCF7E9D	-,3
$\mathcal{C}^4_{99}$	0123458A6BC97FDE	3,3	$\mathcal{C}^4_{153}$	0123458A6B9CEDF7	-,3
$\mathcal{C}^4_{100}$	0123458A6B9CF7ED	-,3	$\mathcal{C}^4_{154}$	0123467859CFBEAD	3,3
$\mathcal{C}^4_{101}$	0123458A6BCFED79	3,3	$\mathcal{C}^4_{155}$	0123467859CFEBDA	3,3
$\mathcal{C}^4_{102}$	012345786A9CDBEF	-,3	$\mathcal{C}^4_{156}$	0123458A69CFE7BD	-,3
$\mathcal{C}^4_{103}$	0123458A69C7DFEB	3,3	$\mathcal{C}^4_{157}$	0123458A69CEFB7D	-,3
$\mathcal{C}^4_{104}$	0123458A69C7FDBE	3,3	$\mathcal{C}^4_{158}$	0123458A6BCF7D9E	2,2
$\mathcal{C}^4_{105}$	0123458A697CBEFD	-,3	$\mathcal{C}^4_{159}$	0123458A6BCED79F	2,2
$\mathcal{C}^4_{106}$	0123458A697CBFDE	-,3	$\mathcal{C}^4_{160}$	0123468B59CED7AF	-,3
$\mathcal{C}^4_{107}$	0123458A69CE7FDB	3,3	$\mathcal{C}^4_{161}$	0123458A6B7CEDF9	3,3
$\mathcal{C}^4_{108}$	0123458A6C9FEB7D	-,2	$\mathcal{C}^4_{162}$	0123458A6B7CDFE9	3,3
$\mathcal{C}^4_{109}$	0123458A6CB9F7ED	-,3	$\mathcal{C}^4_{163}$	0123468C59BDE7AF	-,3
$\mathcal{C}^4_{110}$	0123458A69CFD7BE	3,3	$\mathcal{C}^4_{164}$	0123458A6B7C9FDE	3,3
$\mathcal{C}^4_{111}$	0123458A69BC7FDE	3,3	$\mathcal{C}^4_{165}$	0123458A6B7C9EFD	3,3
$\mathcal{C}^4_{112}$	0123458A6C7EBFD9	-,3	$\mathcal{C}^4_{166}$	012345896ABCE7DF	-,2
$\mathcal{C}^4_{113}$	0123458A6C7FBE9D	-,3	$\mathcal{C}^4_{167}$	0123458A67BC9EFD	-,3
$\mathcal{C}^4_{114}$	012345786ACFBDE9	3,3	$\mathcal{C}^4_{168}$	0123458A6CBFE7D9	2,2
$\mathcal{C}^4_{115}$	012345786ACBE9DF	3,3	$\mathcal{C}^4_{169}$	012345786ACFB9ED	-,3
$\mathcal{C}^4_{116}$	0123458A6C9D7FBE	-,2	$\mathcal{C}^4_{170}$	012345786ACEB9DF	-,2
$\mathcal{C}^4_{117}$	0123458A6C9D7EFB	-,3	$\mathcal{C}^4_{171}$	0123458A6CBF7E9D	2,2
$\mathcal{C}^4_{118}$	0123458A6C9FDB7E	3,3	$\mathcal{C}^4_{172}$	0123458A6C9DBF7E	2,2
$\mathcal{C}^4_{119}$	012345786ACB9FED	-,3	$\mathcal{C}^4_{173}$	012345786A9CBDFE	-,3
$\mathcal{C}^4_{120}$	0123458A6C7EBDF9	3,3	$\mathcal{C}^4_{174}$	0123458A69CF7EBD	3,3
$\mathcal{C}^4_{121}$	0123458A6C7FBD9E	3,3	$\mathcal{C}^4_{175}$	012345786ACDE9BF	-,3
$\mathcal{C}^4_{122}$	0123458A6BCE79FD	-,3	$\mathcal{C}^4_{176}$	0123457869ACFEBD	3,3
$\mathcal{C}^4_{123}$	0123458A69BCE7DF	3,3	$\mathcal{C}^4_{177}$	0123457869BCEAFD	-,3
$\mathcal{C}^4_{124}$	0123458A69CEBDF7	3,3	$\mathcal{C}^4_{178}$	0123458A6C7DBFE9	3,3
$\mathcal{C}^4_{125}$	0123458A69CB7EFD	-,3	$\mathcal{C}^4_{179}$	012345786A9CEDBF	-,3

Table 12: The 302 classes of 4-bit permutations

Class	Truth table	Sharing	Class	Truth table	Sharing
$\mathcal{C}^4_{180}$	0123458A6C9D7FEB	3,3	$\mathcal{C}^4_{225}$	0123456879CEBFDA	3,3
$\mathcal{C}^4_{181}$	012345896ABC7FDE	-,3	$\mathcal{C}^4_{226}$	012345786ABC9FDE	-,3
$\mathcal{C}^4_{182}$	0123458A67BC9FDE	-,3	$\mathcal{C}^4_{227}$	012345786ACFD9BE	-,3
$\mathcal{C}^4_{183}$	012345896ACF7BED	3,3	$\mathcal{C}^4_{228}$	0123458A69BCEDF7	3,3
$\mathcal{C}^4_{184}$	0123458A67CF9BED	3,3	$\mathcal{C}^4_{229}$	0123458A6C9DBFE7	-,3
$\mathcal{C}^4_{185}$	012345896ACE7BFD	-,3	$\mathcal{C}^4_{230}$	0123458A6CEB7FD9	-,3
$\mathcal{C}^4_{186}$	0123458A67CF9BDE	-,3	$\mathcal{C}^4_{231}$	0123468B59CEDA7F	3,3
$\mathcal{C}^4_{187}$	012345786ACEFB9D	3,3	$\mathcal{C}^4_{232}$	0123458A6C9FDBE7	-,3
$\mathcal{C}^4_{188}$	012345786ACFEB9D	-,3	$\mathcal{C}^4_{233}$	0123458A67B9CFDE	2,2
$\mathcal{C}^4_{189}$	0123457869CEFBDA	3,3	$\mathcal{C}^4_{234}$	012345896AB7CFDE	2,2
$\mathcal{C}^4_{190}$	0123458A6C7DBEF9	-,3	$\mathcal{C}^4_{235}$	0123458A69B7CEFD	-,3
$\mathcal{C}^4_{191}$	0123458A6C7FB9DE	-,3	$\mathcal{C}^4_{236}$	0123458A6B97CFDE	2,2
$\mathcal{C}^4_{192}$	0123458A6C7FBED9	3,3	$\mathcal{C}^4_{237}$	0123458A69B7CFDE	-,3
$\mathcal{C}^4_{193}$	0123458A6C7FDB9E	-,3	$\mathcal{C}^4_{238}$	0123457689CEAFBD	2,2
$\mathcal{C}^4_{194}$	012345786ACFED9B	3,3	$\mathcal{C}^4_{239}$	0123457689CEAFDB	-,3
$\mathcal{C}^4_{195}$	0123458A6BC7DE9F	-,3	$\mathcal{C}^4_{240}$	012345768A9CDEFB	3,3
$\mathcal{C}^4_{196}$	0123468C59BDEA7F	3,3	$\mathcal{C}^4_{241}$	012345768A9CDEBF	-,2
$\mathcal{C}^4_{197}$	0123458A6CBDE97F	-,3	$\mathcal{C}^4_{242}$	012345768A9CDFEB	-,3
$\mathcal{C}^4_{198}$	0123458A69C7BEFD	3,3	$\mathcal{C}^4_{243}$	012345768ACF9BDE	2,2
$\mathcal{C}^4_{199}$	0123458A6BCFD9E7	-,2	$\mathcal{C}^4_{244}$	012345768ACE9BFD	2,2
$\mathcal{C}^4_{200}$	0123458A6BCFD79E	-,3	$\mathcal{C}^4_{245}$	012345768ACF9BED	-,3
$\mathcal{C}^4_{201}$	012345786ACB9FDE	3,3	$\mathcal{C}^4_{246}$	0123456879BAEFDC	-,2
$\mathcal{C}^4_{202}$	012345786ACE9DFB	3,3	$\mathcal{C}^4_{247}$	012345687AB9DEFC	3,3
$\mathcal{C}^4_{203}$	012345786ACF9BDE	-,3	$\mathcal{C}^4_{248}$	0123456879CEFBDA	-,2
$\mathcal{C}^4_{204}$	012345786ACE9BFD	-,2	$\mathcal{C}^4_{249}$	0123458A69CFEB7D	3,3
$\mathcal{C}^4_{205}$	012345786ACDB9EF	3,3	$\mathcal{C}^4_{250}$	0123458A69CD7FEB	-,3
$\mathcal{C}^4_{206}$	012345896ABCEDF7	-,3	$\mathcal{C}^4_{251}$	0123458A69CEF7DB	-,3
$\mathcal{C}^4_{207}$	0123458A67BCEDF9	-,3	$\mathcal{C}^4_{252}$	0123458A69CEFBD7	2,2
$\mathcal{C}^4_{208}$	0123458A69C7BFDE	3,3	$\mathcal{C}^4_{253}$	0123458A69CE7FBD	-,3
$\mathcal{C}^4_{209}$	0123468B59CF7DAE	-,3	$\mathcal{C}^4_{254}$	0123458A69BCFD7E	3,3
$\mathcal{C}^4_{210}$	0123468A5BCF7D9E	-,3	$\mathcal{C}^4_{255}$	012345786ABCEDF9	-,3
$\mathcal{C}^4_{211}$	0123458A69CED7FB	3,3	$\mathcal{C}^4_{256}$	012345896ACF7BDE	-,3
$\mathcal{C}^4_{212}$	0123458A69BC7EFD	3,3	$\mathcal{C}^4_{257}$	012345896ABCFD7E	-,2
$\mathcal{C}^4_{213}$	012345896ABC7EFD	-,2	$\mathcal{C}^4_{258}$	012345896ACE7BDF	2,2
$\mathcal{C}^4_{214}$	0123458A67CEB9FD	2,2	$\mathcal{C}^4_{259}$	012345896ACEFDB7	2,2
$\mathcal{C}^4_{215}$	012345896ACEB7FD	2,2	$\mathcal{C}^4_{260}$	012345896AB7CEFD	2,2
$\mathcal{C}^4_{216}$	0123457869CDEFBA	-,2	$\mathcal{C}^4_{261}$	0123458A69CEB7FD	-,3
$\mathcal{C}^4_{217}$	012345687ABC9EFD	3,3	$\mathcal{C}^4_{262}$	0123458A6C7DB9FE	2,2
$\mathcal{C}^4_{218}$	0123457869BCDEFA	-,3	$\mathcal{C}^4_{263}$	0123458A6BC7EDF9	-,3
$\mathcal{C}^4_{219}$	012345786ACF9BED	3,3	$\mathcal{C}^4_{264}$	0123458A6C7DFEB9	2,2
$\mathcal{C}^4_{220}$	0123468A59CFDE7B	-,3	$\mathcal{C}^4_{265}$	0123458A6BCDE9F7	-,3
$\mathcal{C}^4_{221}$	0123457869CEAFDB	3,3	$\mathcal{C}^4_{266}$	0123468A5BCFED97	2,2
$\mathcal{C}^4_{222}$	0123467859CFEADB	-,3	$\mathcal{C}^4_{267}$	012345786ABCE9DF	-,3
$\mathcal{C}^4_{223}$	0123468A5BCFDE79	2,2	$\mathcal{C}^4_{268}$	0123458A69CFBED7	3,3
$\mathcal{C}^4_{224}$	0123457869CEBFDA	-,3	$\mathcal{C}^4_{269}$	0123458A69CEBFD7	-,3

Table 13: The 302 classes of 4-bit permutations

Class	Truth table	Sharing	Class	Truth table	Sharing
$\mathcal{C}^4_{270}$	0123468B5C9DEA7F	3,3	$\mathcal{C}^4_{286}$	0123458967CEFBDA	2,2
$\mathcal{C}^4_{271}$	0123468B5C9DAFE7	-,3	$\mathcal{C}^4_{287}$	012345768A9CDFBE	3,3
$\mathcal{C}^4_{272}$	0123468B5CD79FAE	-,3	$\mathcal{C}^4_{288}$	0123456789CEFBDA	2,2
$\mathcal{C}^4_{273}$	0123458A6C7FEB9D	3,3	$\mathcal{C}^4_{289}$	0123456789CEBFDA	-,3
$\mathcal{C}^4_{274}$	0123458A6BCED97F	-,3	$\mathcal{C}^4_{290}$	0123456789BCEAFD	-,3
$\mathcal{C}^4_{275}$	0123458A6CF7BE9D	3,3	$\mathcal{C}^4_{291}$	012345768A9BCFED	-,3
$\mathcal{C}^4_{276}$	0123458A6CF7BD9E	-,3	$\mathcal{C}^4_{292}$	012345768A9BCEFD	2,2
$\mathcal{C}^4_{277}$	0123458A6BC9DE7F	3,3	$\mathcal{Q}^4_{293}$	0123457689CDEFBA	1,1
$\mathcal{C}^4_{278}$	0123468B5CD7AF9E	3,3	$\mathcal{Q}^4_{294}$	0123456789BAEFDC	1,1
$\mathcal{C}^4_{279}$	0123458A6BC7DFE9	-,3	$\mathcal{C}^4_{295}$	0123468C59DFA7BE	-,3
$\mathcal{C}^4_{280}$	0123457869ACEDBF	3,3	$\mathcal{C}^4_{296}$	0123468A5BCF7E9D	2,2
$\mathcal{C}^4_{281}$	0123457869ACFBDE	3,3	$\mathcal{C}^4_{297}$	0123468A5BCF79DE	2,2
$\mathcal{C}^4_{282}$	0123468B5CD7F9EA	-,3	$\mathcal{C}^4_{298}$	012345687ACEB9FD	-,2
$\mathcal{C}^4_{283}$	0123468B5C9DE7AF	-,3	$\mathcal{Q}^4_{299}$	012345678ACEB9FD	1,1
$\mathcal{C}^4_{284}$	0123458A6BCF9D7E	-,3	$\mathcal{Q}^4_{300}$	0123458967CDEFAB	2,1
$\mathcal{C}^4_{285}$	0123457869CEAFBD	-,2	$\mathcal{C}^4_{301}$	0123458967CDEFBA	-,1

Table 14: Known S-boxes and their classes

Class	Cipher	Class	Cipher
$\mathcal{C}^4_{39}$	DESL Row2, DESL Row3	$\mathcal{C}^4_{203}$	DESL Row1
$\mathcal{C}^4_{46}$	DES7 Row3	$\mathcal{C}^4_{204}$	DES2 Row2, DES3 Row2
$\mathcal{C}^4_{59}$	DES7 Row1	$\mathcal{C}^4_{206}$	Gost K7
$\mathcal{C}^4_{69}$	DES3 Row1, DES7 Row0	$\mathcal{C}^4_{208}$	Twofish q0 t1
$\mathcal{C}^4_{74}$	DES6 Row1	$\mathcal{C}^4_{209}$	Serpent4, Serpent5, HB2 S2, Optimal G_{15}
$\mathcal{C}^4_{80}$	DES8 Row2	$\mathcal{C}^4_{210}$	Clefia0, Twofish q0 t2, HB1 S0, HB2 S3, Optimal G_{14}
$\mathcal{C}^4_{85}$	DES1 Row0, DES1 Row1, DES1 Row2, DES8 Row3	$\mathcal{C}^4_{220}$	DES6 Row0
$\mathcal{C}^4_{97}$	DES8 Row0	$\mathcal{C}^4_{221}$	DES5 Row2
$\mathcal{C}^4_{108}$	Twofish q1 t1	$\mathcal{C}^4_{223}$	Noekeon, Luffa v1, Piccolo, Optimal G_8
$\mathcal{C}^4_{117}$	DES2 Row0, DES6 Row3	$\mathcal{C}^4_{229}$	Twofish q1 t2
$\mathcal{C}^4_{120}$	Twofish q0 t3	$\mathcal{C}^4_{231}$	JH S0, JH S1, Optimal G_{13}
$\mathcal{C}^4_{137}$	DES8 Row1	$\mathcal{C}^4_{253}$	Gost K3
$\mathcal{C}^4_{139}$	DES3 Row0, DES5 Row0	$\mathcal{C}^4_{254}$	DES5 Row1
$\mathcal{C}^4_{142}$	Twofish q1 t3	$\mathcal{C}^4_{257}$	DES3 Row3
$\mathcal{C}^4_{145}$	Gost K6	$\mathcal{C}^4_{266}$	Present, Serpent2, Serpent6, Luffa v2, Hamsi, Optimal G_1
$\mathcal{C}^4_{148}$	DES5 Row3	$\mathcal{C}^4_{267}$	Gost K4
$\mathcal{C}^4_{153}$	Twofish q1 t0	$\mathcal{C}^4_{270}$	Klein, KhazadP, KhazadQ, Iceberg G0, Iceberg G1, Puffin, Optimal G_4
$\mathcal{C}^4_{154}$	Gost K5	$\mathcal{C}^4_{272}$	Optimal G_6
$\mathcal{C}^4_{160}$	Serpent3, Serpent7, Clefia2, Clefia3, HB1 S1, HB1 S3, HB2 S0, Optimal G_9	$\mathcal{C}^4_{275}$	Gost K2
$\mathcal{C}^4_{163}$	Clefia1, HB1 S2, HB2 S1, Optimal G_{10}	$\mathcal{C}^4_{278}$	Optimal G_5
$\mathcal{C}^4_{166}$	DES2 Row1, DESL Row0	$\mathcal{C}^4_{279}$	DES2 Row3, DES4 Row0, DES4 Row1, DES4 Row2, DES4 Row3, DES7 Row2
$\mathcal{C}^4_{172}$	Gost K1	$\mathcal{C}^4_{281}$	DES6 Row2
$\mathcal{C}^4_{177}$	Gost K8	$\mathcal{C}^4_{282}$	Inversion in $GF(2^4)$, Optimal G_3 mCrypton S0,S1,S2,S3
$\mathcal{C}^4_{184}$	DES1 Row3	$\mathcal{C}^4_{283}$	Optimal G_{12}
$\mathcal{C}^4_{188}$	Lucifer S0	$\mathcal{C}^4_{295}$	Optimal G_{11}
$\mathcal{C}^4_{190}$	Twofish q0 t0	$\mathcal{C}^4_{296}$	Serpent1, Optimal G_0
$\mathcal{C}^4_{196}$	Optimal G_7	$\mathcal{C}^4_{297}$	Serpent0, Optimal G_2
$\mathcal{C}^4_{197}$	Lucifer S1		

Table 15: Quadratic decomposition length 2

Class # in A_{16}	Quadratic Decomposition length 2: quadratic × quadratic	# simple solutions
$\mathcal{C}^4_{130}$	300×299	1
$\mathcal{C}^4_{131}$	299×300	1
$\mathcal{C}^4_{150}$	$12 \times 293, 293 \times 300, 300 \times 12, 300 \times 300$	4
$\mathcal{C}^4_{151}$	$12 \times 300, 293 \times 12, 300 \times 293, 300 \times 300$	4
$\mathcal{C}^4_{158}$	299×293	1
$\mathcal{C}^4_{159}$	293×299	1
$\mathcal{C}^4_{168}$	$12 \times 300, 293 \times 293, 300 \times 12, 300 \times 300$	4
$\mathcal{C}^4_{171}$	$293 \times 12, 293 \times 300, 294 \times 293, 294 \times 300$	4
$\mathcal{C}^4_{172}$	$12 \times 293, 293 \times 294, 300 \times 293, 300 \times 294$	4
$\mathcal{C}^4_{214}$	$4 \times 299, 12 \times 12, 12 \times 294, 12 \times 299, 293 \times 4, 293 \times 12, 293 \times 294, 293 \times 299, 294 \times 12, 294 \times 294, 294 \times 299, 300 \times 4, 300 \times 12, 300 \times 294, 300 \times 299$	15
$\mathcal{C}^4_{215}$	$4 \times 293, 4 \times 300, 12 \times 12, 12 \times 293, 12 \times 294, 12 \times 300, 294 \times 12, 294 \times 293, 294 \times 294, 294 \times 300, 299 \times 4, 299 \times 12, 299 \times 293, 299 \times 294, 299 \times 300$	15
$\mathcal{C}^4_{223}$	$12 \times 293, 293 \times 293, 293 \times 294, 294 \times 293, 294 \times 294, 299 \times 12, 299 \times 299$	7
$\mathcal{C}^4_{233}$	$12 \times 12, 293 \times 293, 293 \times 300, 294 \times 12, 294 \times 300, 299 \times 12, 300 \times 293, 300 \times 300$	8
$\mathcal{C}^4_{234}$	$12 \times 12, 12 \times 294, 12 \times 299, 293 \times 293, 293 \times 300, 300 \times 293, 300 \times 294, 300 \times 300$	8
$\mathcal{C}^4_{236}$	$12 \times 12, 293 \times 293, 293 \times 294, 293 \times 300, 294 \times 293, 294 \times 294, 299 \times 299, 300 \times 293, 300 \times 300$	9
$\mathcal{C}^4_{238}$	$12 \times 300, 293 \times 293, 300 \times 12, 300 \times 300$	4
$\mathcal{C}^4_{243}$	$4 \times 293, 4 \times 294, 12 \times 4, 12 \times 293, 12 \times 294, 12 \times 299, 293 \times 12, 293 \times 294, 294 \times 4, 294 \times 12, 294 \times 293, 294 \times 294, 299 \times 4, 299 \times 293, 299 \times 294, 300 \times 12, 300 \times 294, 300 \times 299$	18
$\mathcal{C}^4_{244}$	$4 \times 12, 4 \times 294, 4 \times 299, 12 \times 293, 12 \times 294, 12 \times 300, 293 \times 4, 293 \times 12, 293 \times 294, 293 \times 300, 294 \times 4, 294 \times 12, 294 \times 293, 294 \times 294, 294 \times 299, 294 \times 300, 299 \times 12, 299 \times 300$	18
$\mathcal{C}^4_{252}$	$299 \times 300, 300 \times 299$	2
$\mathcal{C}^4_{258}$	$4 \times 12, 4 \times 300, 12 \times 4, 12 \times 12, 12 \times 293, 12 \times 294, 12 \times 299, 12 \times 300, 293 \times 12, 293 \times 294, 293 \times 299, 294 \times 12, 294 \times 293, 294 \times 299, 294 \times 300, 299 \times 12, 299 \times 293, 299 \times 294, 299 \times 300, 300 \times 4, 300 \times 12, 300 \times 294, 300 \times 299$	23
$\mathcal{C}^4_{259}$	$4 \times 12, 4 \times 300, 12 \times 12, 12 \times 293, 12 \times 294, 12 \times 299, 12 \times 300, 293 \times 4, 293 \times 12, 293 \times 294, 293 \times 299, 294 \times 4, 294 \times 12, 294 \times 293, 294 \times 294, 294 \times 300, 299 \times 12, 299 \times 293, 299 \times 294, 299 \times 300, 300 \times 12, 300 \times 294, 300 \times 299$	23
$\mathcal{C}^4_{260}$	$4 \times 293, 4 \times 294, 12 \times 4, 12 \times 12, 12 \times 293, 12 \times 294, 12 \times 299, 12 \times 300, 293 \times 12, 293 \times 294, 293 \times 299, 294 \times 12, 294 \times 293, 294 \times 294, 294 \times 299, 294 \times 299, 299 \times 12, 299 \times 293, 299 \times 300, 300 \times 4, 300 \times 12, 300 \times 294, 300 \times 299$	23
$\mathcal{C}^4_{262}$	$12 \times 299, 294 \times 299, 299 \times 12, 299 \times 294$	4
$\mathcal{C}^4_{264}$	$12 \times 294, 293 \times 293, 293 \times 300, 294 \times 12, 294 \times 300, 299 \times 299, 300 \times 293, 300 \times 294$	8
$\mathcal{C}^4_{266}$	$12 \times 12, 293 \times 300, 294 \times 299, 299 \times 294, 299 \times 299, 300 \times 293, 300 \times 300$	7
$\mathcal{C}^4_{286}$	$12 \times 293, 12 \times 300, 293 \times 12, 293 \times 300, 300 \times 12, 300 \times 293, 300 \times 300$	7
$\mathcal{C}^4_{288}$	$12 \times 12, 293 \times 300, 300 \times 293, 300 \times 300$	4
$\mathcal{C}^4_{292}$	$4 \times 4, 4 \times 12, 4 \times 294, 12 \times 4, 12 \times 12, 12 \times 293, 12 \times 294, 12 \times 300, 293 \times 12, 293 \times 294, 293 \times 299, 294 \times 4, 294 \times 12, 294 \times 293, 294 \times 294, 294 \times 299, 294 \times 300, 299 \times 293, 299 \times 294, 299 \times 300, 300 \times 12, 300 \times 294, 300 \times 299$	23
$\mathcal{C}^4_{296}$	$12 \times 299, 293 \times 293, 293 \times 300, 294 \times 12, 294 \times 300, 299 \times 294, 299 \times 299$	7
$\mathcal{C}^4_{297}$	$12 \times 294, 293 \times 293, 294 \times 299, 299 \times 12, 299 \times 299, 300 \times 293, 300 \times 294$	7