From Substitution Box To Threshold

Anubhab Baksi[1], Sylvain Guilley[2], Ritu-Ranjan Shrivastwa[2], and Sofiane Takarabt[2]

[1] Nanyang Technological University, Singapore

[2] Secure-IC, Rennes, France

anubhab001@e.ntu.edu.sg, sylvain.guilley@secure-ic.com,
ritu-ranjan.shrivastwa@secure-ic.com, sofiane.takarabt@secure-ic.com

Abstract. With the escalating demand for lightweight ciphers as well as side channel protected implementation of those ciphers in recent times, this work focuses on two related aspects. First, we present a tool for automating the task of finding a Threshold Implementation (TI) of a given Substitution Box (SBox). Our tool returns 'with decomposition' and 'without decomposition' based TI. The 'with decomposition' based implementation returns a combinational SBox; whereas we get a sequential SBox from the 'without decomposition' based implementation. Despite being high in demand, it appears that this kind of tool has been missing so far. In the process, we report new decomposition for the PRESENT SBox (improving from Poschmann et al.'s JoC'11 paper) and that of the GIFT SBox (improving from Jati et al.'s TIFS'20 paper). Second, we show an algorithmic approach where a given cipher implementation can be tweaked (without altering the cipher specification) so that its TI cost can be significantly reduced. We take the PRESENT cipher as our case study (our methodology can be applied to other ciphers as well). Indeed, we show over 31 percent reduction in area and over 52 percent reduction in depth compared to the basic threshold implementation.

Keywords: Lightweight Cryptography · Block Cipher · SBox · Side Channel Countermeasure · Threshold Implementation · PRESENT

1 Introduction

Over the last few years, we have observed a surge of research works dedicated to finding new lightweight ciphers and/or low-cost implementation of those ciphers. Recently we have also seen side channel protected implementation of the lightweight ciphers gaining traction [15,23]. While the community is proactive in advocating side channel protected implementation, surprisingly, a systematic and generic study on how to do that appears to be missing from the literature. There is an overall theory, but for the most part, a detailed study is required to be undertaken for better understanding of the context. The authors seem to come up with some ad-hoc approach (see, e.g., [23]) for the implementation. This further hinders the overdue tasks such as, finding proper algorithms, easy-to-use and publicly available tools, and various optimisations that can be employed to reduce the cost.

This work makes a humble attempt to look into the problem of finding a *Threshold Implementation* (TI) of a given SBox. The TI is a well-known concept that aims at protecting against the most common type of side channel attack which relies on power or electromagnetic leakage [9,23,26]. Such SBoxes are probably the most common choice for the non-linear component in the modern lightweight ciphers, thus an automated tool to find TI of those SBoxes is of prime importance.

The first author would like to thank Robert Hines, Bijoy Das (IIT Bhilai, India), Gan Peizhou (NTU, Singapore) and Aneesh Kandi (IIT Madras, India). This is an extended version of the paper with the same title accepted in Indocrypt 2023.

Further, we observe that with a slight modification in the implementation, it is possible to reduce the cost for a TI of cipher. This does not alter the actual description of the cipher, hence there is no need to redo the security analysis. We show how to adopt a systematic approach with the lightweight cipher PRESENT [11], and how it successfully reduces the TI cost, by more than 31% in terms of area or by more of 52% in terms of depth. This methodology is generic, hence it can be applied to other ciphers that use a similar structure.

Our Contributions

The side channel attack is considered among the major threats, though, the field of studying the protected implementations seems less than developed at multiple levels. While working on this general area, we find ourselves in a situation where we need/want to find threshold implementation of a fairly large pool of SBoxes. Effectively, we look for automated tools to do the batch processing, but to our dismay, cannot find anything suitable (except for the tool by Petkova-Nikova mentioned in Section 4). Parallel to this, the problem of finding some optimisations at the algorithmic level is another interesting direction, which seems underdeveloped too.

Ultimately, we decide to make our own tool for this purpose. As we go along, we discover a proper algorithm is missing. For the most part, the previous authors such as [23, Section III] or [9, Chapter 3] convey the idea behind threshold through examples, instead of a well-formed algorithm. Naturally, several pertinent issues remain unexplored, such as automating the process or dealing with the corner cases. We therefore look further down with adequate clarity to come up with algorithms (as well as release our codes as open-source[1]). Our tool has two segments, one segment returns 'without decomposition' based threshold implementation (Section 3), and the other segment returns 'with decomposition' based threshold implementation (Section 4). The idea for decomposition presented here, while being similar to that one used in [23, 39], is simplified. On top of that, we show decomposition for the PRESENT and GIFT SBoxes, which improve in terms of ASIC cost compared to the currently best-known results from [39] (for PRESENT) and [23] (for GIFT).

On the other hand, better understanding of TI motivates us to find other avenues for cost reduction. With the lightweight block cipher PRESENT [11], we show an optimisation strategy (Section 5) that uses less resources. The basic idea stems from the concept of *affine equivalence* (Section 2.2) of SBoxes, i.e., by opting for another SBox than specified by the designers. This SBox is chosen in such as a way that the netlist optimisation tool can leverage on the new SBox's algebraic properties. However, our approach does not change the overall cipher specification (so one does not need to carry out the usual security analysis). Thus, our analysis reveals, some alternative representations are indeed more efficient to implement than the naïve implementation (which is used as a baseline). Further, this strategy can be applied to any other cipher with a similar structure.

2 Background

2.1 Möbius Transform

The Möbius transform is used to find the coordinate function representation (in ANF) of a given (bijective) SBox. For the sake of better clarity, it is described here. Given an $n \times n$ SBox S, we first need to define the corresponding $2^n \times 2^n$ *Möbius matrix* (see Definition 1).

[1] Our codes can be accessed at https://github.com/anubhab001/sbox-threshold-public.

Definition 1 (Möbius Matrix). *The* $2^n \times 2^n$ *Möbius matrix* M_{2^n} *takes elements from* $\mathbb{F}_2$ *and is recursively defined as:* $M_{2^0} = \begin{bmatrix} 1 \end{bmatrix}$, $M_{2^1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, *and* $M_{2^n} = \begin{bmatrix} M_{2^{n-1}} & M_{2^{n-1}} \\ \mathbf{0} & M_{2^{n-1}} \end{bmatrix}$ *for* $n \geq 2$.

Remark 1. The Möbius matrix is self-inverse. □

Example 1. One of the most commonly used Möbius matrix that corresponds to a 4-bit SBox is given by:

$$M_{2^4} = \begin{pmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix}$$

□

Definition 2 (Coordinate Function). *Suppose* $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *is defined as* $F(x) = (f_0(x), \ldots, f_{n-1}(x))$ *for all* $x \in \mathbb{F}_2^n$, *where* $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ *for* $i = 0, \ldots, n-1$. *Then each* f_i *is called a coordinate function of* F.

We also use the following two terms, *bit-slice matrix* and *coordinate matrix*. Informally, these two can be defined as follows. The bit-slice matrix of an $n \times n$ SBox is a $n \times 2^n$ matrix where column i stores the i^{th} look-up entry as a binary column vector. The coordinate matrix is the $n \times 2^n$ binary matrix which is the result of post-multiplying the Möbius matrix M_{2^n} to the bit-slice matrix.

The coordinate matrix plays an important role in representing the SBox in its coordinate functions (Definition 2) in Algebraic Normal Form (ANF). The rows indicate the coordinate functions, while the columns indicate which product of the input variables is present. This is shown in Example 2 with respect to the SBox used in `PRESENT` [11].

Example 2 (`PRESENT` *SBox).* The bit-slice matrix of the `PRESENT` SBox (`C56B90AD3EF84712`) is given as follows:

$$B_{\texttt{PRESENT}} = \begin{pmatrix}
0&1&0&1&1&0&0&1&1&0&1&0&0&1&1&0\\
0&0&1&1&0&0&1&0&1&1&1&0&0&1&0&1\\
1&1&1&0&0&0&0&1&0&1&1&0&1&1&0&0\\
1&0&0&1&1&0&1&1&0&1&1&1&0&0&0&0
\end{pmatrix}$$

For instance, the last column of this matrix, $(0, 1, 0, 0)^\top$, is the last look-up entry (2) written as the binary column vector. After post-multiplying the Möbius matrix with this matrix, one gets the following as the coordinate matrix of the PRESENT SBox:

$$C_{\texttt{PRESENT}} = B_{\texttt{PRESENT}} \times M_{2^4} = \begin{pmatrix} 0&1&0&0&1&0&1&0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1&1&0&1&1&1&1&0&0 \\ 1&0&0&1&1&0&0&0&1&1&1&1&0&1&0&0 \\ 1&1&1&0&0&0&1&1&1&0&0&1&0&1&0&0 \end{pmatrix}$$

Following the conventional notation, we denote the input variables as x_0, x_1, x_2, x_3 and the coordinate functions as y_0, y_1, y_2, y_3. Then the rows of the $C_{\texttt{PRESENT}}$ denote (in sequence) y_0, y_1, y_2, y_3, and the columns denote (in sequence) whether or not $(1, x_0, x_1, x_0x_1, x_2, x_0x_2, x_1x_2, x_0x_1x_2, x_3, x_0x_3, x_1x_3, x_0x_1x_3, x_2x_3, x_0x_2x_3, x_1x_2x_3, x_0x_1x_2x_3)$ is present. For instance, consider the top row, where 1 is present corresponding to columns $\{x_0, x_2, x_1x_2, x_3\}$, it means that: $y_0 = x_0 \oplus x_1x_2 \oplus x_2 \oplus x_3$.

In this way, rest of the coordinate functions of the PRESENT SBox can be computed:

$$\begin{aligned} y_1 &= x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \oplus x_1x_3 \oplus x_1 \oplus x_2x_3 \oplus x_3, \\ y_2 &= x_0x_1x_3 \oplus x_0x_1 \oplus x_0x_2x_3 \oplus x_0x_3 \oplus x_1x_3 \oplus x_2 \oplus x_3 \oplus 1, \\ y_3 &= x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \oplus x_0 \oplus x_1x_2 \oplus x_1 \oplus x_3 \oplus 1. \end{aligned}$$

Note that the algebraic degree of the three coordinate functions is 3 and some of the monomials are common (indeed reducing the number of monomials of algebraic degree 3 is the working factor in our work at Section 5). □

Remark 2. The bit-slice matrix (and consequently the coordinate matrix too) of a bijective SBox is of full row-rank. □

Remark 3. Sometimes the coordinate functions of an SBox (written in algebraic normal form) are called "SBox to ANF" in the literature. □

Remark 4. For a bijective SBox, the last column of its coordinate matrix is null. □

Remark 5. The maximum of the algebraic degrees of the coordinate functions of an SBox is termed as the algebraic degree of the SBox. □

2.2 Affine Equivalence of SBoxes

We call SBoxes S_0 and S_1 *Affine Equivalent* (AE) if there exist two non-singular binary matrices A_0, A_1 (each with compatible dimension) such that $S_1 = A_0 \circ S_0 \circ A_1$; i.e., $S_1(x) = A_0 S_0(A_1(x \oplus c_0)) \oplus c_1$, for all inputs x and for some binary vectors c_0, c_1 of compatible dimension.

To find affine equivalent SBoxes of a given $n \times n$ SBox, binary non-singular matrices are required. The General Linear group over $\mathbb{F}_q$ of degree n, denoted by $\mathrm{GL}(n, \mathbb{F}_q)$, consists of $n \times n$ non-singular matrices with each entry is from $\mathbb{F}_q$. It is known thanks to Euler that, the order of $\mathrm{GL}(n, \mathbb{F}_q)$ is, $\prod_{k=0}^{n-1} (q^n - q^k)$. Therefore, the number of 3×3 and 4×4 binary (so, $q = 2$) non-singular matrices are 168 and 20160, respectively.

2.3 Side Channel Attack and Countermeasure

Side channel attacks, particularly those relying on information from power consumption or electromagnetic emanation, are of prominent concern while dealing with the physical security of the ciphers [5, 9, 25, 27, 31, 38]. It has been systematically shown that a cipher with sufficient classical security claims falls short against an adversary equipped with a side channel attack set-up. It therefore goes without saying, understanding the attacks and finding low-cost countermeasures are among the top research priorities by/for the community.

Side-channel attacks are based on the connection between a (learned) model and any intermediate variable in the implementation that might be leaking. Therefore, the countermeasures attempt to destroy the linkage of the model and the intermediate variables. *Masking* [31, Chapter 9] is considered a prominent countermeasure. A masking scheme randomly distributes each intermediate to introduce randomness in a way that the overall algorithmic flow in the cipher is unchanged, but the randomised operations makes the side channel leakage free from the intermediate variables. Depending on the strength of the attacker, various *degrees* of masking can be adopted.

Threshold Implementation

The threshold implementation (TI) is a form of masking, and is among the top recommendations against the side channel attacks [9, 10, 23, 26, 32, 34, 35, 36] specially for protecting the hardware implementation. Aside from a protection against the hardware implementation, TI is also claimed to work with the software implementation of a cipher [21, 40].

Typically, the TI of an affine function is considered straightforward, while that of a non-linear (in most block ciphers, the only non-linear component is the SBox) function is considered a strenuous task to accomplish. The TI of a given SBox can be realised either as *without decomposition* (the SBox is implemented as a combinational circuit) or as *with decomposition* (the SBox is implemented as a sequential circuit, typically it allows for smaller and more shallow netlists at the expense of pipelining) [23].

3 Threshold without Decomposition (Combinational SBox)

In essence, the without decomposition based threshold implementation takes the coordinate function (in ANF) form of an SBox, then converts into a suitable implementation satisfying a set of conditions. We consider the usual notation here, i.e., for an $n \times n$ SBox S we denote the input variables as $x_0, x_1, \dots, x_{n-1}$ and the output variables as $y_0, y_1, \dots, y_{n-1}$. Then it substitutes each x_i and each y_j by $d+1$ shares (where $d \geq$ the algebraic degree of the SBox). In the process, each the x_i and y_i variables are respectively replaced by the new variables $x_{i,j}$ and $y_{i,j}$; where $i \in \{0, 1, \dots, n-1\}$ and $j \in \{0, 1, \dots, d\}$. For simplicity, we write an SBox S in terms of its coordinate functions (in this case, $y_0, y_1, \dots, y_{n-1}$) as:

$$S : \begin{cases} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{cases}$$

Based on the existing literature [9, 16, 23, 26, 36], the 'without decomposition' based TI of S satisfies the following conditions:

(α) **Sharing of input variables.** $\bigoplus_{j=0}^{d} x_{i,j} = x_i$, for $i = 0, 1, \dots, n-1$.

(β) **Sharing of output variables/Correctness.** $\bigoplus_{j=0}^{d} y_{i,j} = y_i$ for $i = 0, 1, \dots, n-1$.

(γ) **Non-completeness.** At least one variable from $\{x_{i,0}, x_{i,1}, \ldots, x_{i,d}\}$ is missing in each of $y_{j,0}, y_{j,1}, \ldots, y_{j,d}$ for all $i, j \in \{0, 1, \ldots, n-1\}$.

(δ) **Uniformity.** All non-zero entries in the x_i ($\forall i$) versus $y_{i,j}$ ($\forall i, j$) frequency distribution table are equal.

After this, $d+1$ separate SBoxes $S_0, S_1, \ldots, S_d$ are implemented in parallel, where these SBoxes are given by the following arrangement of the coordinate functions:

$$S_0 : \begin{cases} y_0 & = y_{0,0} \\ y_1 & = y_{1,0} \\ & \vdots \\ y_{n-1} & = y_{n-1,0} \end{cases}$$

$$S_1 : \begin{cases} y_0 & = y_{0,1} \\ y_1 & = y_{1,1} \\ & \vdots \\ y_{n-1} & = y_{n-1,1} \end{cases}$$

$$\vdots$$

$$S_d : \begin{cases} y_0 & = y_{0,d} \\ y_1 & = y_{1,d} \\ & \vdots \\ y_{n-1} & = y_{n-1,d} \end{cases}$$

Thanks to the ingenious arrangement, each of the component SBoxes $(S_0, \ldots, S_d)$ computes some share of the original SBox (S). When combined, the coordinate functions of S (namely, $y_0, \ldots, y_{n-1}$) can be realised even if one $x_{i,j}$ variable in each $S_0, \ldots, S_d$ is randomised. Since one input variable $x_{i,j}$ is randomised, it makes the corresponding side channel leakage random, making it hard for the attacker to exploit secret information from the leakage (this outlines the philosophy of the side channel protection). At the beginning of the (protected) cipher execution, some input variables are fed random inputs and at the end all the shares are combined to get the intended cipher output. In other words the fundamental concept can be described as running multiple randomised modules (each is a bit different from the others) among which the inputs to the cipher are distributed. One module does not give exploitable information to the attacker, still the combined output from the modules results in the desired cipher output.

Example 3 (Uniformity). For the sake of clarity, here we present an example of a threshold sharing that does not have the uniformity property (Condition (δ)), which is adopted from [9, Chapter 3.3.1]. Consider the function, $y = x_0 x_1$ with the sharing (where $x_i = x_{i,0} \oplus x_{i,1} \oplus x_{i,2}$ for $i = 0, 1$; and $y = y_0 \oplus y_1 \oplus y_2$)[2]:

[2] Notice that all $9 = 3 \times 3$ cross-terms (i.e., quadratic monomials) appear in the expressions. Further, notice that y_j does not contain any $x_{\cdot,j}$, this satisfies non-completeness (Condition (γ)).

$$
\begin{aligned}
y_0 &= x_{0,1}x_{1,1} \oplus x_{0,1}x_{1,2} \oplus x_{0,2}x_{1,1},\\
y_1 &= x_{0,2}x_{1,2} \oplus x_{0,0}x_{1,2} \oplus x_{0,2}x_{1,0},\\
y_2 &= x_{0,0}x_{1,0} \oplus x_{0,0}x_{1,1} \oplus x_{0,1}x_{1,0}.
\end{aligned}
$$

Table 1: Non-conformity to uniformity property (example with $y = x_0x_1$)

(a) Frequency distribution table for (x_0, x_1) versus (y_0, y_1, y_2)

x_0	x_1	(y_0, y_1, y_2)							
		000	011	101	110	001	010	100	111
0	0	7	3	3	3	0	0	0	0
0	1	7	3	3	3	0	0	0	0
1	0	7	3	3	3	0	0	0	0
1	1	0	0	0	0	5	5	5	1

(b) Computation for $(x_0, x_1) = (1, 1)$

		$(x_{1,0}, x_{1,1}, x_{1,2})$			
		$(0,0,1)$	$(0,1,0)$	$(1,0,0)$	$(1,1,1)$
$(x_{0,0}, x_{0,1}, x_{0,2})$	$(0,0,1)$	$y_0 = 0$ $y_1 = 1$ $y_2 = 0$	$y_0 = 1$ $y_1 = 0$ $y_2 = 0$	$y_0 = 0$ $y_1 = 1$ $y_2 = 0$	$y_0 = 1$ $y_1 = 0$ $y_2 = 0$
	$(0,1,0)$	$y_0 = 1$ $y_1 = 0$ $y_2 = 0$	$y_0 = 1$ $y_1 = 0$ $y_2 = 0$	$y_0 = 0$ $y_1 = 0$ $y_2 = 1$	$y_0 = 0$ $y_1 = 0$ $y_2 = 1$
	$(1,0,0)$	$y_0 = 0$ $y_1 = 1$ $y_2 = 0$	$y_0 = 0$ $y_1 = 0$ $y_2 = 1$	$y_0 = 0$ $y_1 = 0$ $y_2 = 1$	$y_0 = 0$ $y_1 = 1$ $y_2 = 0$
	$(1,1,1)$	$y_0 = 1$ $y_1 = 0$ $y_2 = 0$	$y_0 = 0$ $y_1 = 0$ $y_2 = 1$	$y_0 = 0$ $y_1 = 1$ $y_2 = 0$	$y_0 = 1$ $y_1 = 1$ $y_2 = 1$

The example is given through Table 1. In particular, the corresponding frequency distribution table is shown in Table 1(a). An instance on how Table 1(a) is computed is shown in Table 1(b); where the computation corresponding to the last row (i.e., $(x_0, x_1) = (1, 1)$; and consequently, $y = 1$) is shown. One may note from Table 1(b) that (y_0, y_1, y_2) equals $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$) and $(1, 1, 1)$ with respective frequency of 5, 5, 5 and 1.

Notice from Table 1(a) that, the non-zero entries are not equal. Hence, this sharing does not conform to the uniformity property. □

Remark 6 (Note on uniformity). As stated in [9, Chapter 3.3], the absence of the uniformity property (Condition (δ)) in a threshold circuit does not leak side channel information by itself; but if it is used to drive another circuit, this second circuit may leak side channel information. However, we could not find any experimental evaluation in the literature, and this could be an interesting problem for future study. □

Remark 7 (Operational uniformity (relaxed) condition). In this article, we consider a more natural uniformity constraint. We aim at specifying a condition whereby the template attacks on the SBoxes are not possible. For all coordinate $0 \leq i < n$ and share $0 \leq j \leq d$, each distribution of $Y_{i,j}|X = x$ must be the same, irrespective the value of $x \in \mathbb{F}_2^n$.

As each $Y_{i,j}$ is a Boolean variable, this means that for all fixed value of $x \in \mathbb{F}_2^n$, we check that $\Pr(Y_{i,j}) \in [0, 1]$ is the same. This reflects the first-order probing resistance in the *Strong Non-Interference* (SNI, see [7]) setting. Now, note that $\Pr(Y_{i,j} = 1) = E(Y_{i,j})$. For the `PRESENT` SBox, we get for all x (see Section 5) the same distributions that follow:

$$\begin{cases} \Pr(Y_{0,0}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{0,1}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{0,2}|X=x)=\frac{3}{2^7}\ \ , \Pr(Y_{0,3}|X=x)=0\ \ \ ;\\ \Pr(Y_{1,0}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{1,1}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{1,2}|X=x)=\frac{27}{2^{10}}\ , \Pr(Y_{1,3}|X=x)=\frac{21}{2^{10}}\ ;\\ \Pr(Y_{2,0}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{2,1}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{2,2}|X=x)=\frac{27}{2^{10}}\ , \Pr(Y_{2,3}|X=x)=\frac{21}{2^{10}}\ ;\\ \Pr(Y_{3,0}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{3,1}|X=x)=\frac{1}{2^5}\ , \Pr(Y_{3,2}|X=x)=\frac{27}{2^{10}}\ , \Pr(Y_{3,3}|X=x)=\frac{21}{2^{10}}\ . \end{cases}$$

For instance, the fact that '$\Pr(Y_{0,3})=0$' is not a weakness. This results from the fact that the first coordinate of the SBox could be split in simply in $d=3$ shares while obeying TI precepts. From what we can tell, this is not a vulnerability. Otherwise, plunging any licit TI implementation enjoying a sharing into d into a sharing into $d+1$ would no longer be considered a valid, as the extra added share would be consistently equal to 0 (as in our case of the first coordinate function of the `PRESENT` SBox). □

3.1 Need for a Well-developed Algorithm

As noted (Section 1), the corresponding engineering methods to apply TI does not appear to be developed enough. No specific algorithm is available to the best of our understanding/finding, rather the concept is described mostly through some cherry-picked examples [9, 23]. This is somewhat surprising, specially given the state-of-the-art explores advanced side channel attack (e.g., [28, 29]). The problems that arise from explaining a concept through examples instead of a developing a proper algorithm can be manifested in a number of ways, as listed next.

Problem 1 (Ambiguous sharing). Since the concept is mostly communicated through examples, it is in general hard to come up with a proper algorithm. To illustrate the point, we look into the example given in [23, Section III.A]. The 3-variable quadratic function $f(x,y,z)=xy\oplus z$ is shown with the following 3-shares:

$$\begin{aligned} f_1 &= z_2 \oplus x_2y_2 \oplus x_2y_3 \oplus x_3y_2,\\ f_2 &= z_3 \oplus x_1y_3 \oplus x_3y_1 \oplus x_3y_3,\\ f_3 &= z_1 \oplus x_1y_1 \oplus x_1y_2 \oplus x_2y_1. \end{aligned}$$

This sharing is not unique. For instance, the following is also valid sharing of f_1 and f_2 (f_3 is unchanged):

$$\begin{aligned} f_1 &= \mathbf{z_3} \oplus x_2y_2 \oplus x_2y_3 \oplus x_3y_2,\\ f_2 &= \mathbf{z_2} \oplus x_1y_3 \oplus x_3y_1 \oplus x_3y_3. \end{aligned}$$

□

Problem 2 (Corner case: Low algebraic degree of a coordinate function). Consider the 3-bit SBox, `03214756`, which is given by the following coordinate functions:

$$\begin{aligned} y_0 &= x_0 \oplus x_1x_2,\\ y_1 &= x_0 \oplus x_1x_2 \oplus x_1,\\ y_2 &= x_2. \end{aligned}$$

Note that the coordinate function y_2 is of algebraic degree 1 whereas the rest are of algebraic degree 2 (i.e., not every coordinate function has the maximum algebraic degree). Thus, it is possible to consider 3-share masking for y_0 and y_1, but only 2-share is sufficient for y_2. Based on the existing literature (such as, the threshold implementation of PRESENT in [39]), it is not clear if this $(3, 3, 2)$-sharing is not considered valid (the number of shares for an SBox is taken as a constant for all its coordinate functions, regardless of the algebraic degree of the individual coordinate functions), even though it may be possible that this unequal sharing for individual coordinate functions does not leak any exploitable side channel information. Further research may be conducted on this.

The same problem appears with some of the mainstream SBoxes. High-profile examples include PRESENT (given in Example 2), PICCOLO (E4B238091A7F6C5D) [41], PYJAMASK-128 (2D397BA6E0F4851C) [22], which are respectively given by the following coordinate functions (parenthesised numbers indicate the algebraic degree):

$$\begin{aligned}
y_0 &= x_0x_1x_2 \oplus x_0x_1 \oplus x_0x_2 \oplus x_0x_3 \oplus x_1x_2x_3 \oplus x_1x_3 \oplus x_1 \oplus x_2 \oplus x_3 \ (3),\\
y_1 &= x_0x_1 \oplus x_0 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_3 \oplus 1 \ (3),\\
y_2 &= x_1x_2 \oplus x_1 \oplus x_2 \oplus x_3 \oplus 1 \ (2),\\
y_3 &= x_0 \oplus x_2x_3 \oplus x_2 \oplus x_3 \oplus 1 \ (2);
\end{aligned}$$

and

$$\begin{aligned}
y_0 &= x_0x_1x_2 \oplus x_0x_1 \oplus x_0x_2 \oplus x_0x_3 \oplus x_0 \oplus x_1 \oplus x_2x_3 \oplus x_2 \ (3),\\
y_1 &= x_0x_2 \oplus x_0 \oplus x_2x_3 \oplus 1 \ (2),\\
y_2 &= x_0x_1x_2 \oplus x_0x_1 \oplus x_0 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_2 \oplus x_3 \ (3),\\
y_3 &= x_0 \oplus x_1x_2 \oplus x_3 \ (2).
\end{aligned}$$

□

Problem 3 (No automation). There does not appear to be any publicly available tool to server the purpose. For instance, the without decomposition based threshold implementation for the GIFT SBox (1A4C6F392DB7508E) [6] is computed manually. It is a laborious task, as it requires to compute the entire expression consisting of a few hundred to a few thousands (if not more) of monomials. In general, this task is tedious to carry out manually beyond 3-bit SBoxes and borderline impossible starting from 5-bit SBoxes. □

Problem 4 (Lack of further optimisation). In certain cases, there could be opportunity to further optimise the threshold expressions. For example, one may look for factorisation so that the netlist can be further optimised (without compromising the side channel protection). Other logic gates than {AND, XOR} can also be used to reduce the hardware cost. □

3.2 Our Approach

In this work, we take each coordinate function of the given SBox (which is essentially a vectorial Boolean function) one at a time, before moving on the next one. A bird's eye view of our approach can be found in Algorithm 1.

First we introduce the input and output sharing variables $(x_{i,j}, y_{i,j})$ to compute the Boolean function of the form: $\bigoplus_j y_{i,j} =$ function of $x_{i,j}$ where i is fixed[3]. After this, each monomial from

[3] Note that, the naïve AND count in the coordinate functions of the SBox affects the total number of monomials in the shares (more naïve AND count means more monomials in the coordinate functions of the SBox, this in turn leads to more monomials in the shares). The exact relationship in which the naïve AND count affects the total number of monomials is complicated, but it seems the relationship is similar to exponential.

the RHS is taken one-by-one and assigned to the LHS variables (i.e., $y_{i,j}$ for a fixed i) so that the Condition (β)) (i.e., use all the monomials from the RHS) and non-completeness) is satisfied. Next, in order to satisfy Condition (γ) (i.e., not all input sharing variables are used in any $y_{i,j}$), we determine a conflict list. For instance, if this list looks like this: $(\{x_{i,0}\}, \{x_{i,1}\}, \ldots)$; it means that, no $y_{i,0}$ can contain $x_{i,0}$, no $y_{i,1}$ can contain $x_{i,1}$, and so on. Then each monomial (which is picked up from the RHS of the expression before sharing) is checked for intersection with the element of conflict list corresponding to each $y_{i,j}$. If the intersection is null, then that monomial is assigned to $x_{i,j}$. The remaining property, Condition (δ), is hard to ensure in this way; therefore we simply check if it holds after the threshold implementation is completed, if it does not hold then we randomise and try again.

In total, we offer 3 options to randomise. In the first option, we shuffle the RHS of the combined expression. In the second option, we allow the ordering in which the $y_{i,j}$ variables are picked (for a fixed i). In the third, we allow to randomise the conflict list (e.g., allowing $y_{i,0}$ to have conflict with $x_{i,1}$ instead of $x_{i,0}$).

Algorithm 1: Sharing for threshold (without decomposition) for an SBox

```
Input: An $n \times n$ SBox given as coordinate functions $(y_0, \ldots, y_{n-1}$ with input variables $x_0, \ldots, x_{n-1})$
Output: Assignment of $d+1$ SBoxes given as coordinate functions
 1: for $i \leftarrow 0$ to $n-1$ do                                  ▷ Iterate over each coordinate function
 2:     for $j \leftarrow 0$ to $n-1$ do                              ▷ Iterate over all input variables
 3:         $X_j \leftarrow x_{j,0} \oplus x_{j,1} \oplus \cdots \oplus x_{j,d}$     ▷ Share input variables
 4:     $Y \leftarrow$ Substitute $x_j = X_j$ in $y_i$ for all $j$     ▷ Compute RHS of complete output sharing
 5:     for $j \leftarrow 0$ to $d$ do
 6:         $y_{i,j} \leftarrow 0$                                    ▷ Shares for $y_i$
 7:     for each monomial $m \in Y$ do                                ▷ Ordering can be randomised
 8:         for $j \leftarrow 0$ to $d$ do                            ▷ Ordering can be randomised
 9:             if $m$ is not conflicting with $y_j$ then             ▷ Conflicting variables can be randomised
10:                 $y_{i,j} \leftarrow y_{i,j} \oplus m$             ▷ Non-completeness is respected
11:     if Uniformity not satisfied then
12:         Go back to Step 5 with randomised options                 ▷ Try again with other assignment of $y_{i,j}$'s
13:     return Sharing of $y_i$ as $(y_{i,0}, y_{i,1}, \ldots, y_{i,d})$    ▷ $\bigoplus_j y_{i,j} = y_i$
```

Warning for Zero-sharing We display a warning message if some $y_{i,j}$ is a constant (happens if some coordinate function has less algebraic degree). As an illustration, reconsider the SBox described in Problem 2 (`03214756`). By exactly following Algorithm 1, the sharing is given as (notice that a warning is shown for $y_{2,2}$):

$$
\begin{aligned}
y_{2,0} &= x_{2,1} \oplus x_{2,2}, \\
y_{2,1} &= x_{2,0}, \\
y_{2,2} &= 0.
\end{aligned}
$$

It can be stated that, the presence of a zero sharing does not inherently contradict uniformity (Condition (δ)). Further, one may note that, the case for zero sharing can happen for a larger ($\geq$ 4-bit) SBox. For instance, consider a high degree SBox but some of its coordinate functions are affine (this kind of SBox is used in practice, such as that one used in `DEFAULT-LAYER` [3]).

3.3 Results

In the following results, we do not enforce uniformity (Condition (δ)). This is in concordance with the existing literature (e.g., [23]). Further, although not conclusive yet, our experiment hints that it may be hard to ensure this property (as our tool sometimes has to iterate few thousand times before finding a uniform sharing) in general. It is even possible that the this property does not hold for some Boolean functions[4].

Minimal Order (Algebraic Degree + 1) Sharing Our approach can deal with 3×3 and 4×4 SBoxes without any difficulty. Further, 5×5 SBoxes, such as that of ASCON's [19] works well (one may note from [37] that ASCON is recently selected as the primary choice in the LWC project run by NIST). Results with some SBoxes are summarised in Table 2, where we show the number of monomials along with STM 130nm ASIC cost (HCMOS9GP, Cadence v14.20 2016) in terms of gate equivalent (rounded off to nearest integer). Each of the SBoxes is used in a cipher, save for 048AFC691EBD7532 which is presented in [20, Section 3.4]. The number of shares are taken as the algebraic degree of the SBox plus 1.

Table 2: Without decomposition cost of some SBoxes (uniformity not enforced)

			Shares	# Monomials	Hardware☆
GIFT [6]		1A4C6F392DB7508E	4	265☆	526
PRESENT [11]		C56B90AD3EF84712	4	666	1132
PRINCE [12]		BF32AC916780E5D4	4	731	991
PICCOLO [41]		E4B238091A7F6C5D	4	399	645
SKINNY-64 [8]		C6901A2B385D4E7F	4	398	723
TWINE [42]		C0FA2B9583D71E64	4	626	723
PYJAMASK-128 [22]		2D397BA6E0F4851C	4	373	640
QARMA [1]		ADE6F735980CB124	4	562	826
NOEKEON Gamma [17]		7A2C48F0591E3DB6	4	387	697
DEFAULT [2, 3]	LS	037ED4A9CF18B265	3	102	60
	Non-LS	196F7C82AED043B5	4	213	412
Gao-Roy-Oswald [20]		048AFC691EBD7532	4	988	1658
ASCON★ [19]			3	160	85

☆: Gate equivalent in STM 130nm ASIC library (HCMOS9GP)

☆: Used in [23, Appendix B]

★: $\mathtt{4B\overline{1F141A}1592\overline{1B}58\overline{121D}36\overline{1C1E13}7E0D\overline{111810}C1\overline{1916}AF\overline{17}}$

Higher Order ($\geq$ Algebraic Degree + 2) Sharing Our tool can work with higher number of shares as well, without any actual change in the algorithm flow. We show some examples in Table 3, where the number of shares is taken as the algebraic degree of the SBox plus 2. It is worth noting that ASCON has recently been selected as the NIST LWC winner [37].

4 Threshold with Decomposition (Sequential SBox)

The theory for a decomposition based TI requires finding two other SBoxes such that the composition of these SBoxes is the target SBox. Given an $n \times n$ SBox S, we want to find two $n \times n$ (bijective) SBoxes F_0 and F_1 such that

[4] To the best of our knowledge, no proof about the existence of the uniformity property is presented in the literature.

Table 3: Higher order without decomposition cost of some SBoxes (uniformity not enforced)

		Shares	# Monomials	Hardware☆
GIFT [6]	1A4C6F392DB7508E	5	451	432
SKINNY-64 [8]	C6901A2B385D4E7F	5	682	681
QARMA [1]	ADE6F735980CB124	5	967	983
ASCON★ [19]		4	257	142

☆: Gate equivalent in STM 130nm ASIC library (HCMOS9GP)

★: 4B$\overline{\texttt{1F141A15}}92\overline{\texttt{1B}}58\overline{\texttt{121D}}36\overline{\texttt{1C1E13}}$7E0D$\overline{\texttt{111810}}$C1$\overline{\texttt{1916}}AF\overline{\texttt{17}}$

(i) $F_1 \circ F_0 \equiv S$ (i.e., $F_1(F_0(x)) = S(x)\ \forall x$);
(ii) the algebraic degrees of both F_0 and F_1 are less than the algebraic degree of S.

*Example 4 (*PRESENT *SBox decomposition from [39]).* The authors in [39] find the following decomposition for the PRESENT SBox (C56B90AD3EF84712, cubic): F_1 = 1A4F3C69D2875E0B (quadratic) and F_0 = 5C6F7E184D3A2B09 (quadratic). The coordinate functions are:

$$F_1 : \begin{cases} y_0 = x_0 \oplus x_1 \oplus 1, \\ y_1 = x_0 \oplus x_2x_3 \oplus x_2, \\ y_2 = x_0x_2 \oplus x_0x_3 \oplus x_1 \oplus x_3, \\ y_3 = x_0 \oplus x_2x_3 \oplus x_3. \end{cases} \qquad F_0 : \begin{cases} y_0 = x_0 \oplus x_1x_2 \oplus x_1 \oplus x_3 \oplus 1, \\ y_1 = x_1 \oplus x_2, \\ y_2 = x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus 1, \\ y_3 = x_0. \end{cases}$$

Note that, when constructing the coordinate functions of the target SBox (refer Example 2 for this); y_i of F_0 becomes the corresponding x_i of F_1, $\forall i$. In other words, we first start with the coordinate functions of F_1, the replace each x_i with the RHS of the i^{th} coordinate function of F_0. For instance, to get y_0 of the PRESENT SBox ($= x_0 \oplus x_1x_2 \oplus x_2 \oplus x_3$); we start with y_0 of F_1 ($= \mathbf{x_0} \oplus \mathbf{x_1} \oplus 1$); then we replace $\mathbf{x_0}$ with y_0 of F_0 ($= x_0 \oplus x_1x_2 \oplus x_1 \oplus x_3 \oplus 1$), and $\mathbf{x_1}$ with y_1 of F_0 ($= x_1 \oplus x_2$). □

Remark 8. It is not possible to decompose a quadratic SBox S in this way. In order to decompose a quadratic SBox, one needs that F_0 and F_1 both have lower algebraic degree than that of S, implying that both F_0 and F_1 are affine. However, composition of two affine SBoxes does not produce a quadratic SBox. Hence, the quadratic SBoxes such as those used in ASCON [19] or BAKSHEESH [4] cannot be decomposed in this way. □

Despite being quite popular [23,33,39], it appears that there exists only one (publicly available) tool which is a courtesy of Petkova-Nikova[5]. This tool, available exclusively as executable files, can find decomposition based threshold for a given 3×3 or 4×4 SBox.

Here we describe a simple idea to find such decomposition in Algorithm 2. In short, we generate F_1 and F_0 in a way that $F_1 \circ F_0 \equiv S$ is always satisfied (by randomly constructing F_0 in a way so that the algebraic degree requirement is already satisfied, then adjusting F_1 according to F_0 and S); then we decide whether to keep or discard depending on the algebraic degree of F_1.

In Step 1, we specify that the algorithm be given candidate SBoxes from which it can generate F_0 (F_0 is affine equivalent to a given candidate SBox). For instance, suppose, we want to find the decomposition of a given cubic 4×4 SBox (which is typically the case for the common ciphers like PRESENT [11], SKINNY-64 [8] or GIFT [6]). In that case, we need to supply quadratic SBoxes to Algorithm 2. Following [9, Chapter 5], we choose the quadratic affine classes, $\{4, 12, 293, 294, 299, 300\}$.

Remark 9. For decomposition of 5-bit SBoxes, one may consider [13, Table 2]. □

[5] Hosted at Svetla Petkova-Nikova's official web-page: https://homes.esat.kuleuven.be/~snikova/ti_tools.html.

Algorithm 2: Threshold (with decomposition) for an SBox

```
Input: An $n \times n$ SBox $S$
Output: Two $n \times n$ SBoxes $F_1$ and $F_0$ such that $F_1 \circ F_0 \equiv S$
1: Pick $F_0$ randomly which is affine equivalent to a given lower algebraic degree SBox
2: for $i \leftarrow 0$ to $2^n - 1$ do                          ▷ Iterate over all SBox inputs
3:     $r \leftarrow F_0(i)$
4:     $F_1(r) \leftarrow S(i)$                                   ▷ $F_0 : i \mapsto r$, $F_1 : r \mapsto S(i)$
5: if algebraic degree of $F_1$ that of $S$ then
6:     Discard $F_1$, and go back to Step 1
7: return $(F_1, F_0)$
```

Overall, Algorithm 2 simplifies the problem (cf. the complication in [39, Section 3.2] or [23, Section III.A]). In summary, the previous approach constructs an exhaustive pool of SBoxes first in a deterministic way, which is extensively time and space consuming.

Remark 10. Based on the existing literature, it is not clear what exactly would be the problem if the algebraic degree of F_0 or F_1 is greater than (or equal to) that of S. The case where the algebraic degree requirement is not satisfied could be considered a future research. □

Some examples of decomposition with respect to three high-profile ciphers can be found in Table 4 (Table 4(a) for PRESENT [11], 4(b) for SKINNY-64 [8], and 4(c) for GIFT [6]). In order to estimate the implementation cost in ASIC (TSMC 65nm and UMC 180nm logic libraries), we use the LIGHTER tool [24] (the backward-compatible code from [18] is used for this purpose, as the URL for the LIGHTER code does not seem accessible). From the estimates, one can see that the our (F_1, F_0) take lower total cost compared to the PRESENT ($F_1 =$ 1A4F3C69D2875E0B $F_0 =$ 5C6F7E184D3A2B09) and GIFT ($F_1 =$ 7DB58E02CA4639F1, $F_0 =$ F9A8BECD71203645) SBox decomposition given in [39] and [23], respectively. This observation is further supported when we check the total ASIC cost under the STM 130nm library (HCMOS9GP, Cadence v14.20 2016), as shown in Table 5(a). More in-depth results for the UMC 65nm technology are shown in Table 5(b); we can see that the SBox pairs that we report in this paper in terms of total GE, total power and total delay. Thus, we infer that our research improves the state-of-the-art results for SBox decomposition, thereby opening the possibility for further reducing the threshold cost for PRESENT [39] and GIFT [23].

Table 4: ASIC costs (in GE) for SBox decomposition using LIGHTER

(a) PRESENT (C56B90AD3EF84712)

F_1	F_0	TSMC 65nm	UMC 180nm
08B7A31C46F9ED52✡	7E92B04D5CA1836F✡	33.50	27.67
547C0129BDA6E8F3	30B874A9FCED1256	36.50	30.67
47095C12E3DA8FB6	54FE32BA98DC0167	36.00	30.67
C5DAF706849BE312	017BA632DC4895EF	38.50	31.67
1A4F3C69D2875E0B	5C6F7E184D3A2B09	29.00	24.67
9B5C7F12DA483E06	32F10E98CD5BA467	37.50	32.67

✡: Taken from [39, Table 2]

Remark 11 (Third order decomposition). It is possible to find third order decomposition (i.e., F_2, F_1, F_0 such that $F_2 \circ F_1 \circ F_0 \equiv S$; and the algebraic degree of each of F_2, F_1, F_0 is less

(b) SKINNY-64 (C6901A2B385D4E7F)

F_1	F_0	TSMC 65nm	UMC 180nm
863C72FBA41E50D9	31FDA85720CE9B46	26.00	22.34
B72F0D95843E1CA6	DF64CE20A8759B13	26.50	22.67
2B7FA1E43C68D095	9AED54018BFC7623	28.00	24.34
863CFB72D950A41E	319BEC7520A8DF64	25.00	22.33
C46E18A37DF529B0	02DF46CE75B9138A	24.50	21.34
A3186EC45FD70B92	64EC20FD138A75B9	26.50	23.00
C368950D1EA47BF2	02468AFD1357B9CE	21.50	19.00
D0593C86A14E2BF7	573198CD4620ABFE	30.00	25.67
4AE1F27BD05968C3	ECB93157FDA80264	27.50	22.34
59D0863CE41A7FB2	7513ABFE640298CD	30.00	26.00
C6A1907F2BD54E38	01453289EFBACD67	30.50	27.01
CB30D42F6E95718A	0B81A3F4297E5CD6	35.00	29.67
570231CEFDB9A846	6FB25C3A4D09E718	29.50	23.67
20138A9BCEDF7546	8F61250734DAE9CB	25.00	22.34

(c) GIFT (1A4C6F392DB7508E)

F_1	F_0	TSMC 65nm	UMC 180nm
5638127C9EF0DAB4☆	4DF71A285CE60B39☆	26.00	22.34
AC14D72B39F68E05	2031BA896475FECD	26.00	21.34
FB2673D90415C8AE	AE9C30572614B8DF	29.00	25.33
7DB58E02CA4639F1	F9A8BECD71203645	23.50	21.34
CA7DB1428E39F506	5160FCAB7342DE89	28.00	24.34
E596308FD2A1CB74	BAFC374298DE1560	26.50	22.67
C1A472DBF9630E58	1230A8B95674ECFD	27.50	23.00
754AF982BDC6310E	D32AB4C579801E6F	28.00	24.33
D5A6192E803F4CB7	42CD3BA560EF1987	28.00	23.67
ACE8B6F27D395410	E0D156AB7948CF32	35.50	30.33

☆: Taken from [23, Table II]

than that of S) using our tool (see also [14, Section 4.1] for an example of higher order decomposition). For instance, the SKINNY-64 SBox (C6901A2B385D4E7F, cubic) can be decomposed as F_2 = 329EAD7645BC8F01 (quadratic), F_1 = 5B941CE0A37D826F (quadratic) and F_0 = 1AD6F3487520C9EB (quadratic). □

5 Further Optimisation Based on Affine Equivalence

In this part, we attempt to reduce the cost of an existing TI by optimising its SBox. In essence, we redesign the cipher with an affine equivalent SBox (so that the cipher specification is unchanged) which reduces the TI cost. The SBox is implemented as a combinational only circuit (i.e., no register) per coordinate functions. We show our method on the lightweight block cipher PRESENT [11], though it can be applied to any other cipher with similar construction. The idea used here can be somewhat compared to the weighted sum based SBox decomposition filtering method presented in [39, Section 3.2].

We study in this section the impact of affine equivalence on the SBox to reduce the cost of TI. Notice that the concept of affine equivalence we propose here is with respect to the representation of the cipher, not about changing the specification of the cipher. In other words, only the implementation changes, but the cipher description remains unchanged.

Table 5: Decomposition of `PRESENT` and `GIFT` SBoxes with ASIC costs
(a) STM 130nm

	F_1	Area		F_0	Area		Total area	
		μm^2	GE		μm^2	GE	μm^2	GE
`PRESENT`	`1A4F3C69D2875E0B`	104.894	17.34	`5C6F7E184D3A2B09`	62.533	10.34	167.427	27.68
	`08B7A31C46F9ED52`✡	90.774	15.00	`7E92B04D5CA1836F`✡	88.757	14.67	179.531	29.67
`GIFT`	`7DB58E02CA4639F1`	94.808	15.67	`F9A8BECD71203645`	60.516	10.00	155.324	25.67
	`5638127C9EF0DAB4`☆	86.740	14.34	`4DF71A285CE60B39`☆	82.705	13.67	169.445	28.01

✡: Taken from [39, Table 2]
☆: Taken from [23, Table II]

(b) UMC 65nm and TSMC 65nm

			UMC 65nm			TSMC 65nm		
			GE	Power*	Delay*	GE	Power*	Delay*
`PRESENT`	F_1	`1A4F3C69D2875E0B`	15.00	9.072	142	23.75	559.0	700
	F_0	`5C6F7E184D3A2B09`	11.75	6.542	224	18.00	471.0	660
	F_1	`08B7A31C46F9ED52`✡	13.75	7.612	244	26.00	641.0	990
	F_0	`7E92B04D5CA1836F`✡	15.25	8.989	215	26.00	665.0	880
`GIFT`	F_1	`7DB58E02CA4639F1`	13.25	8.422	123	14.00	29.3	310
	F_0	`F9A8BECD71203645`	10.50	5.998	121	19.00	432.0	570
	F_1	`5638127C9EF0DAB4`☆	12.50	6.033	238	20.00	514.0	840
	F_0	`4DF71A285CE60B39`☆	16.00	8.714	226	21.50	508.0	660

✡: Taken from [39, Table 2] $*$: Measured in $\times 10^{-7}$ W
☆: Taken from [23, Table II] $\star$: Measured in ps

5.1 Motivation and Basic Observation

The complexity of threshold implementations directly relates to the number of monomials in the coordinate function of the SBox. Let us consider the `PRESENT` [11] (which is also a standard, ISO/IEC 29192-2:2012[6]) SBox, `C56B90AD3EF84712`. From its coordinate functions given in Example 2, we notice the following properties:

- 8 monomials of degree 3,
- 7 monomials of degree 2,
- 10 monomials of degree 1, and
- 2 monomials of degree 0 (constant 1).

Upon a closer inspection, however, we observe some monomials are duplicates. The following factors contribute to the total cost:

1. **Computation:** Number of unique monomials of given degree (i.e., each monomial of a given degree is counted only once even if its multiplicity is higher).
2. **Reduction:** Number of XORs in the coordinate functions (represented in ANF).

About the `PRESENT` SBox, the contributing components are:

- 3 unique monomials of degree 3 (namely, $x_0x_1x_2$, $x_0x_1x_3$, and $x_0x_2x_3$),

[6] https://webstore.ansi.org/Standards/ISO/ISOIEC291922012-1383736.

- 5 unique monomials of degree 2, (the only absent one being x_0x_2),
- 4 monomials of degree 1, and 1 monomial of degree 0 (constant 1),
- 23 XORs.

Since the cost for TI for a higher degree monomial is much higher than a lower degree monomial, our aim here is to minimise the number of unique monomials of degree 3, then of degree 2, etc.

5.2 Improving Efficiency with Affine Equivalent SBox

From an implementation cost standpoint, it can be beneficial to represent the PRESENT block cipher in an equivalent notation where the SBox S is traded for one of its affine equivalent SBox, $S' = A^{-1} \circ S \circ A$, where A is an invertible affine mapping operating in $\mathbb{F}_2^4$. We reiterate that we do not aim at altering the cipher functionality, simply its representation. The usage of the affine mapping is compensated before and after the SBox application.

While detailed description of the cipher is skipped here for space constraint, an overview is given in Figure 1 for a quick reference. The use of the affine equivalent SBox is illustrated in Figure 2. In the part (a) of the figure, a simplified block diagram of PRESENT is depicted, with the main components:

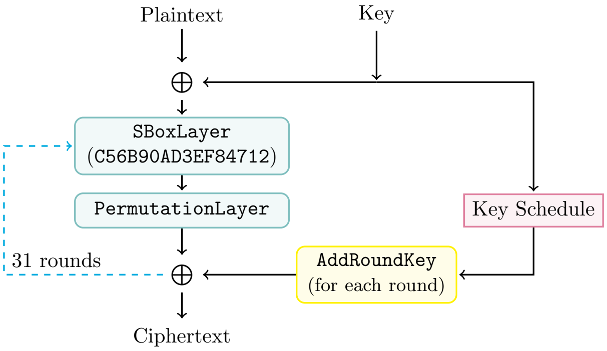
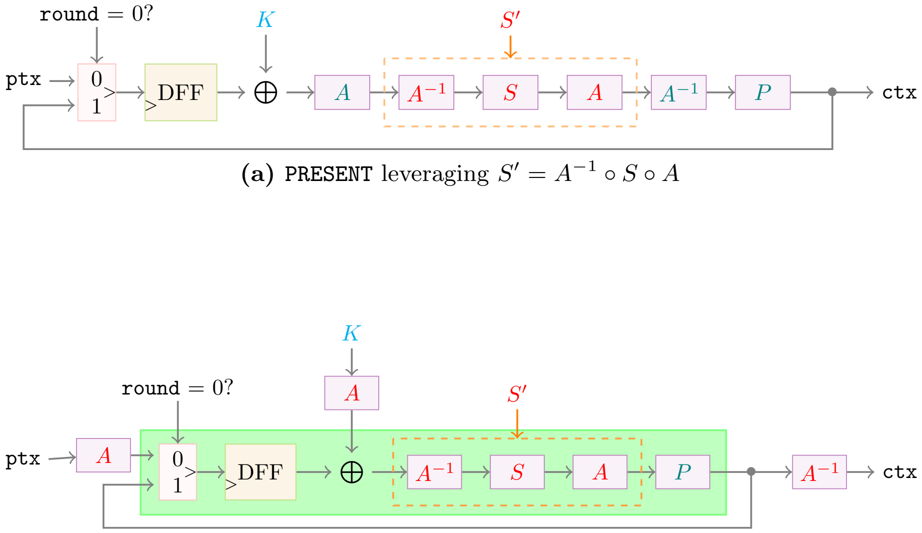


Figure 1: PRESENT encryption (schematic)

- Ports are plaintext (denoted as, "ptx"), round key (denoted as "K"), a selection signal indicating whether the encryption starts (denoted as, "round $= 0$?") or not, and an output ciphertext denoted as "ctx").
- Building blocks are the affine equivalent SBox S, the permutation layer P, a multiplexer allowing to input the plaintext or to iterate, and a DFF barrier storing the result computed till that particular round.

This can be implemented with a shorter critical path by pushing the conversion to/from affine representation outside of the main path. This is represented in the Figure 2(b): Assuming that the plaintext and the round keys are applied the affine transformation, then a regular datapath can be used, provided finally the ciphertext is applied to inverse affine mapping. This equivalent representation leverages the fact that all elements in the PRESENT cipher (permutation layer and multiplexer) are linear. Notice that the scheme in Figure 2(b) is correct only if A is linear. If it is affine, then the constant of the transformation shall be adapted for each operation, in particular the outer ones. The critical path, highlighted as the green box in Figure 2(b), is no different than the original critical path. The architecture shown in Figure 2 is thus suitable for masked TI with combinational SBox (i.e., without decomposition).



(a) `PRESENT` leveraging $S' = A^{-1} \circ S \circ A$

(b) Equivalent `PRESENT` leveraging $S' = A^{-1} \circ S \circ A$

Figure 2: `PRESENT` leveraging affine equivalent SBox

5.3 Results

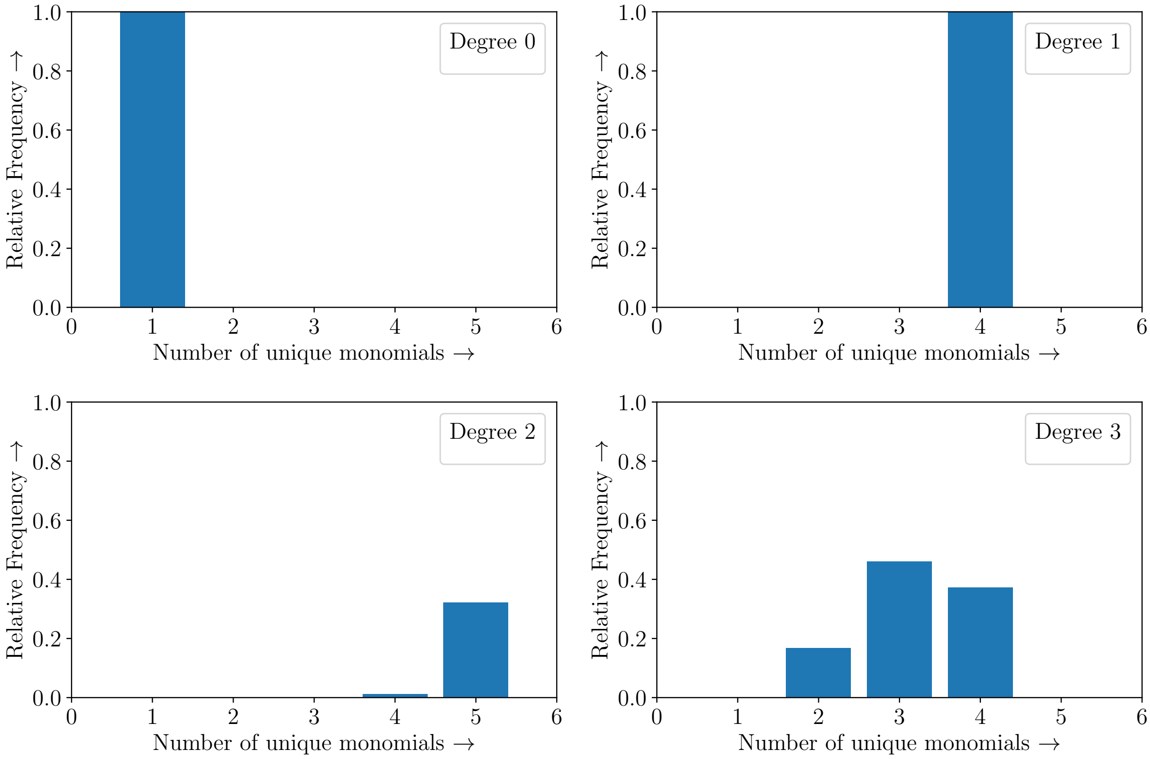
Efficiency Based on Algebraic Property For 1000 random choices of A, we count the number of monomials in each degree, and we get statistics as depicted in Figure 3. More specifically, Figure 3(a) shows the relative frequency distribution for number of unique monomials for each individual degrees. For instance, there exists a unique monomial for degree 0 (constant 1) 100% of the cases. Figure 3(b) shows the probability distribution of XOR count.

It can be seen that it is possible to reduce the number of monomials of degree 3 to only 2. We observe that there is no preferred choice for the constant in the affine transformation.

In our case, the transformed SBox is found as follows. The binary matrix multiplied to obtain the linear part, and the constant binary vector are given respectively by: $\begin{pmatrix} 1&0&0&0 \\ 1&1&1&0 \\ 1&1&0&0 \\ 1&0&1&1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$. The transformed SBox, `4EC20B1A5F3D9867`, has the coordinate functions:

$$\begin{aligned} y_0 &= x_0x_2 \oplus x_1x_2 \oplus x_3, \\ y_1 &= x_0x_2x_3 \oplus x_0 \oplus x_1x_3, \\ y_2 &= x_0x_1x_2 \oplus x_0x_1 \oplus x_1x_3 \oplus x_2 \oplus 1, \\ y_3 &= x_0x_2x_3 \oplus x_0 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1 \oplus x_2x_3. \end{aligned}$$

Therefore, we manage to get a transformed SBox with the following properties in its coordinate functions:



(a) Monomials

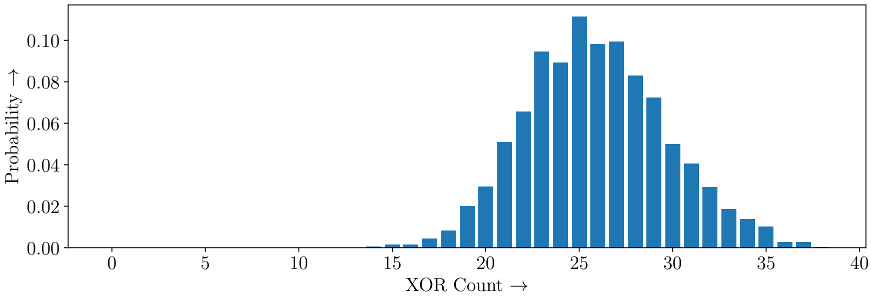
- ○ 2 unique monomials of degree 3 ($x_0x_2x_3$ and $x_0x_1x_2$),
- ○ 5 unique monomials of degree 2 ($x_0x_2, x_1x_2, x_1x_3, x_0x_1$ and x_2x_3),
- ○ 4 monomials of degree 1, and 1 monomial of degree 0 (constant 1),
- ○ only 13 XORs.

Statistics on Netlist Parameters Now we study the real impact of affine transformation to the property of the netlist; namely, we are interested in the algebraic features that drive the netlist properties. We study the netlist area (number of monomials vs. GE) and its logical depth (number of monomials vs. critical path). For the sake of simplicity, every gate is attributed a unitary area and propagation time.

The goal is to figure out which property of the algebraic expression determines most the area and/or depth in the netlist. The following features are considered:

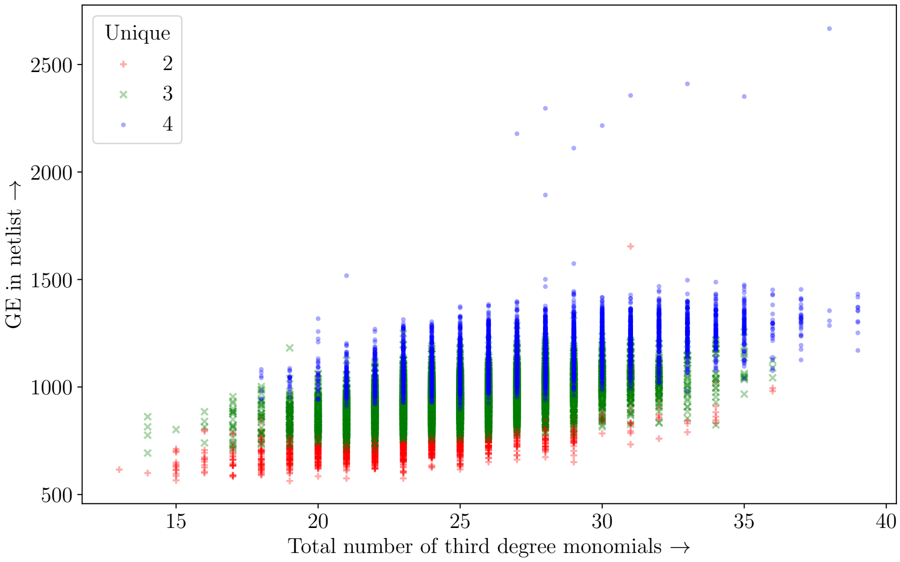
- The number of unique monomials of various degrees (3, 2, 1, and 0) – mostly, the number of unique monomials of degree 3 is to be minimised.
- XOR count is to be reduced to make area usage low.

The results are summarised in Figure 4 (Figure 4(a) for area, 4(b) for logical depth; and finally Figure 4(c) for area versus logical depth); based on 10000 random selections of affine transformation $A : \mathbb{F}_2^4 \to \mathbb{F}_2^4$; where we group based on the number of unique monomials for better readability. It can be seen that, as expected, reducing the unique number of monomials of degree 3 is the main parameter to reduce the gate count and the depth of the netlists. The smallest netlist has 589 gates (of depth 18), and the most shallow one has depth 11 (and 728 gates). For comparison, the reference netlist (i.e., when A is the identity matrix) has 863 gates



(b) XOR

Figure 3: Statistics for affine equivalent SBox search for `PRESENT`

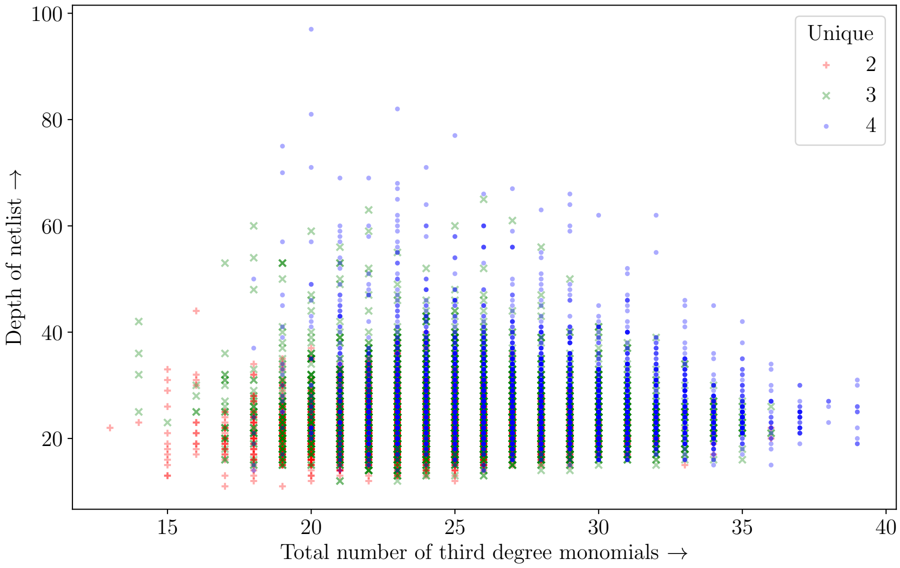


(a) Area (GE) vs. third degree monomials

and a depth of 23. Hence an area reduction of more than 31% or a depth reduction of more than 52% is observed by applying our methodology. Further, as it can be seen from Figure 4(c), larger netlists also have (slightly) longer critical path, on average.

6 Conclusion

This work takes a deeper looks into the problem of finding threshold implementation of SBoxes. The first main contribution of this work is to present an open-source tool for automating the task for threshold implementation for a large pool of SBoxes. Our tool returns 'without decomposition' (Section 3) and 'with decomposition' (Section 4) based implementations. Despite being quite popular, such a tool seems overdue. Among other results, we show improvement over [39] and [23]. The second main contribution (Section 5) comes from an alternate representation of the `PRESENT` SBox [11] so that the TI cost can be reduced. The idea is to replace the original SBox by one of its affine equivalent SBox (so that the cipher description remains unchanged), but this new SBox has lower threshold cost. Overall, we show over 31% improved area and over 52% improved depth compared to the naïve implementation.

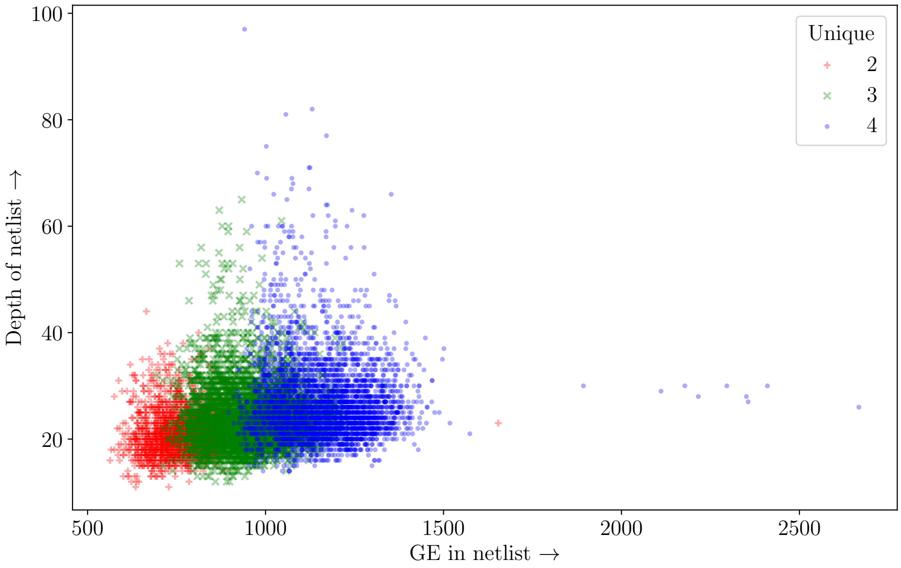


(b) Logical depth vs. third degree monomials

One interesting follow-up work could be to find SBoxes with lower AND count (but with other desirable cryptographic properties) so that the cipher is more suitable for adopting TI. Besides, as noted in Remark 6, it would be interesting to evaluate the amount of side channel leakage from the circuit which takes input from another circuit not obeying the uniformity property. As the main objective in Section 5 is to find another SBox, works like [30, 43] can be incorporated in the search procedure in the future.

References

1. Avanzi, R.: The qarma block cipher family – almost mds matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. Cryptology ePrint Archive, Report 2016/444 (2016) `https://eprint.iacr.org/2016/444`. 11, 12
2. Baksi, A.: Classical and Physical Security of Symmetric Key Cryptographic Algorithms. PhD thesis, School of Computer Science & Engineering, Nanyang Technological University, Singapore (2021) `https://dr.ntu.edu.sg/handle/10356/152003`. 11
3. Baksi, A.: DEFAULT: Cipher-Level Resistance Against Differential Fault Attack. Springer, Singapore (2022) `https://link.springer.com/chapter/10.1007/978-981-16-6522-6_8`. 10, 11
4. Baksi, A., Breier, J., Chattopadhyay, A., Gerlich, T., Guilley, S., Gupta, N., Isobe, T., Jati, A., Jedlicka, P., Kim, H., Liu, F., Martinásek, Z., Sakamoto, K., Seo, H., Shiba, R., Shrivastwa, R.R.: Baksheesh: Similar yet different from gift. Cryptology ePrint Archive, Paper 2023/750 (2023) `https://eprint.iacr.org/2023/750`. 12
5. Baksi, A., Kumar, S., Sarkar, S.: A new approach for side channel analysis on stream ciphers and related constructions. IEEE Transactions on Computers (2021) 4
6. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: Gift: A small present. Cryptology ePrint Archive, Report 2017/622 (2017) `https://eprint.iacr.org/2017/622`. 9, 11, 12, 13
7. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P., Grégoire, B., Strub, P., Zucchini, R.: Strong Non-Interference and Type-Directed Higher-Order Masking. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. (2016) 116–129 7
8. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. IACR Cryptology ePrint Archive **2016** (2016) 660 11, 12, 13
9. Bilgin, B.: Threshold Implementations As Countermeasure Against Higher-Order Differential Power Analysis. PhD thesis, Katholieke Universiteit Leuven and University of Twente (2015) `https://www.esat.kuleuven.be/cosic/publications/thesis-256.pdf`. 1, 2, 4, 5, 6, 7, 8, 12



(c) Area (GE) vs. logical depth for third degree monomials

Figure 4: SBoxes AE to PRESENT SBox

10. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Tokareva, N., Vitkup, V.: Threshold Implementations of Small S-boxes. Cryptography and Communications **7**(1) (2015) 3–33 5
11. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: CHES. Volume 4727., Springer (2007) 450–466 2, 3, 11, 12, 13, 14, 15, 19
12. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knežević, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: Prince - a low-latency block cipher for pervasive computing applications (full version). Cryptology ePrint Archive, Report 2012/529 (2012) https://ia.cr/2012/529. 11
13. Božilov, D., Bilgin, B., Sahin, H.A.: A note on 5-bit quadratic permutations' classification. IACR Transactions on Symmetric Cryptology **2017**(1) (Mar. 2017) 398–404 12
14. Božilov, D., Knežević, M., Nikov, V.: Optimized threshold implementations: Securing cryptographic accelerators for low-energy and low-latency applications. Cryptology ePrint Archive, Paper 2018/922 (2018) https://eprint.iacr.org/2018/922. 14
15. Caforio, A., Collins, D., Glamocanin, O., Banik, S.: Improving first-order threshold implementations of skinny. Cryptology ePrint Archive, Report 2021/1425 (2021) https://ia.cr/2021/1425. 1
16. Daemen, J.: Changing of the guards: a simple and efficient method for achieving uniformity in threshold sharing. Cryptology ePrint Archive, Report 2016/1061 (2016) https://ia.cr/2016/1061. 5
17. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: Nessie Proposal: NOEKEON (2000) http://gro.noekeon.org/Noekeon-spec.pdf. 11
18. Dasu, V.A., Baksi, A., Sarkar, S., Chattopadhyay, A.: LIGHTER-R: optimized reversible circuit implementation for sboxes. In: 32nd IEEE International System-on-Chip Conference, SOCC 2019, Singapore, September 3-6, 2019. (2019) 260–265 13
19. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to NIST (2019) https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ascon-spec-round2.pdf. 11, 12
20. Gao, S., Roy, A., Oswald, E.: Constructing ti-friendly substitution boxes using shift-invariant permutations. In Matsui, M., ed.: Topics in Cryptology – CT-RSA 2019, Cham, Springer International Publishing (2019) 433–452 11
21. Gaspoz, J., Dhooghe, S.: Threshold implementations in software: Micro-architectural leakages in algorithms. Cryptology ePrint Archive, Paper 2022/1546 (2022) https://eprint.iacr.org/2022/1546. 5
22. Goudarzi, D., Jean, J., Kölbl, S., Peyrin, T., Rivain, M., Sasaki, Y., Sim, S.M.: Pyjamask v1.0 (2019) 9, 11
23. Jati, A., Gupta, N., Chattopadhyay, A., Sanadhya, S.K., Chang, D.: Threshold implementations of gift: A trade-off analysis. IEEE Trans. Inf. Forensics Secur. **15** (2020) 2110–2120 1, 2, 5, 8, 11, 12, 13, 14, 15, 19

24. Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. IACR Trans. Symmetric Cryptol. **2017**(4) (2017) 130–168 13
25. Kumar, S., Dasu, V.A., Baksi, A., Sarkar, S., Jap, D., Breier, J., Bhasin, S.: Side Channel Attack On Stream Ciphers: A Three-Step Approach To State/Key Recovery. IACR Transactions on Cryptographic Hardware and Embedded Systems (2022) 4
26. Kutzner, S., Nguyen, P.H., Poschmann, A.: Enabling 3-share threshold implementations for any 4-bit s-box. Cryptology ePrint Archive, Report 2012/510 (2012) https://eprint.iacr.org/2012/510. 1, 5
27. Lomné, V.: Power and Electro-Magnetic Side-Channel Attacks: Threats and Countermeasures. PhD thesis, Docteur de l'Université Montpellier II (2010) https://sites.google.com/site/victorlomne/research. 4
28. Lomné, V., Prouff, E., Rivain, M., Roche, T., Thillard, A.: How to Estimate the Success Rate of Higher-Order Side-Channel Attacks. In Batina, L., Robshaw, M., eds.: Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings. Volume 8731 of Lecture Notes in Computer Science., Springer (2014) 35–54 8
29. Lomné, V., Prouff, E., Roche, T.: Behind the scene of side channel attacks. Cryptology ePrint Archive, Report 2013/794 (2013) https://eprint.iacr.org/2013/794. 8
30. Lu, Z., Mesnager, S., Cui, T., Fan, Y., Wang, M.: An stp-based model toward designing s-boxes with good cryptographic properties. Des. Codes Cryptogr. **90**(5) (2022) 1179–1202 20
31. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - Revealing the secrets of smart cards. Springer (2007) 4, 5
32. Moradi, A., Schneider, T.: Side-channel analysis protection and low-latency in action - case study of prince and midori. Cryptology ePrint Archive, Paper 2016/481 (2016) https://eprint.iacr.org/2016/481. 5
33. Müller, N., Moos, T., Moradi, A.: Low-latency hardware masking of PRINCE. In Bhasin, S., Santis, F.D., eds.: Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings. Volume 12910 of Lecture Notes in Computer Science., Springer (2021) 148–167 12
34. Nikova, S., Nikov, V., Rijmen, V.: Decomposition of permutations in a finite field. Cryptogr. Commun. **11**(3) (2019) 379–384 5
35. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: International conference on information and communications security, Springer (2006) 529–545 5
36. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptol. **24**(2) (2011) 292–321 5
37. NIST: Lightweight Cryptography Standardization Process: NIST Selects Ascon (February 7 2023) https://csrc.nist.gov/News/2023/lightweight-cryptography-nist-selects-ascon. 11
38. Peeters, E.: Advanced DPA Theory and Practice: Towards the Security Limits of Secure Embedded Circuits. 1 edn. Springer-Verlag New York (2013) 4
39. Poschmann, A., Moradi, A., Khoo, K., Lim, C., Wang, H., Ling, S.: Side-Channel Resistant Crypto for Less than 2, 300 GE. J. Cryptology **24**(2) (2011) 322–345 2, 9, 12, 13, 14, 15, 19
40. Sasdrich, P., Bock, R., Moradi, A.: Threshold implementation in software - case study of PRESENT. In Fan, J., Gierlichs, B., eds.: Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings. Volume 10815 of Lecture Notes in Computer Science., Springer (2018) 227–244 5
41. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings. (2011) 342–357 9, 11
42. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A lightweight, versatile block cipher. ECRYPT (2011) https://www.nec.com/en/global/rd/tg/code/symenc/pdf/twine_LC11.pdf. 11
43. Wadhwa, M., Baksi, A., Hu, K., Chattopadhyay, A., Isobe, T., Saha, D.: Finding desirable substitution box with SASQUATCH. IACR Cryptol. ePrint Arch. (2023) 742 20