Enabling 3-share Threshold Implementations for any 4-bit S-box

Sebastian Kutzner[1], Phuong Ha Nguyen[2], and Axel Poschmann[2].

[1] Laboratory of Physical Analysis & Cryptographic Engineering (PACE)
Temasek Laboratories @ NTU.
[2] Division of Mathematical Sciences,
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

Abstract. Threshold Implementation (TI) is an elegant and widely accepted countermeasure against 1-st order Differential Power Analysis (DPA) in Side Channel Attacks. The 3-share TI is the most efficient version of TI, but so far, it can only be applied to 50% of all 4-bit S-boxes. In this paper, we study the limitations of decomposition and introduce factorization to enable the 3-share TI for any optimal 4-bit S-box. We propose an algorithm which can decompose any optimal 4-bit S-box to quadratic vectorial boolean functions with a time complexity of 2^{19}. Furthermore, we use our new methodology in combination with decomposition to optimize ciphers utilizing many different S-boxes, and, to highlight the strength of our new methodology, we construct a 3-share Threshold Implementation of SERPENT which was believed to be not possible until now. Last, we show how to implemented all SERPENT S-boxes with only one mutual core.

1 Introduction

Side Channel Attacks (SCA) [4] were introduced in 1997 by Kocher *et al.* and exploit the fact that while a device is processing data, information about this data is leaked through different channels, e.g., power consumption, electromagnetic emanation and so forth. DPA [5] is a more advanced technique using statistical methods to analyze data collected from many measurements. It exploits the correlation between intermediate results, which depend on a small part of the secret, and the power consumption.

Several countermeasures (far too many to address all of them) have been proposed during the last years, for example, to increase the SNR ratio [9], to balance the leakage of different values [7] or to break the link between the processed data and the secret, i.e., masking [2]. Due to the presence of glitches masked implementation might still be vulnerable to DPA [8,10]. A recent countermeasure against DPA was introduced in 2006 by Nikova *et.al.* [11] and is called *Threshold Implementation* (TI). It is based on secret sharing (or multi-party computation) techniques and is provable secure against first order DPA even in the presence of glitches. Furthermore, it can be implemented very efficiently in hardware [13].

The number of shares required for a Threshold Implementation depends on the degree d of the non-linear function (S-box) and [11,12] have shown that it is at least $d+1$. It implies that the higher the degree of the non-linear function, the more shares are required and the larger is the implementation. Since a degree of two is the minimal degree of a non-linear function, the optimal number of shares is three. Therefore, to apply a 3-share Threshold Implementation to a larger degree function, this function must be represented as a composition of quadratic functions [13].

Nowadays, 4-bit S-boxes are used in cryptographic algorithms due to their efficient hardware implementation. In [6], the authors define a set of 4-bit S-boxes, which fulfill certain cryptographic properties to resist linear and differential cryptanalysis, as optimal S-boxes. The PRESENT S-box is an example for such an optimal 4-bit S-box for which, after decomposition to two quadratic boolean functions, the 3-share TI can be applied [13]. This motivates us to study which of these optimal S-boxes are suitable for the 3-share TI. In this paper, we show that all optimal 4-bit S-boxes which can be protected by the 3-share TI belong to the alternating group A_{16} of the symmetric group S_{16} [1]. This answer implies that we can not apply the 3-share TI to those S-boxes which do not belong to A_{16}. Therefore, we introduce the *Factorization structure*. This new idea has two big contributions: first, it allows us to decompose any 4-bit S-box to quadratic vectorial boolean functions and hence, enabling us to apply the 3-share TI to *any* given 4-bit S-box. Second, it helps us to optimize hardware implementation of ciphers with different S-boxes by enabling sharing a mutual core between them. To support our claims we first show how our idea can be used to efficiently implement the SERPENT S-boxes and second, how to apply the 3-share TI to SERPENT, what, up to now, was believed to be not possible.

Finding a decomposition or factorization of an arbitrary optimal S-box is not a trivial problem. Sometimes, the time complexity is more than 2^{52} and might be beyond our capacity. To solve this problem we first intensively study the structure of optimal S-boxes. We derive an algorithm which can not only decompose any optimal S-box with a time complexity of 2^{19} but also finds decompositions which are very efficient in terms of hardware costs.

The paper is organized as follows. In Section 2 the Threshold Implementation countermeasure and the results of the journal of cryptology paper are recalled. Section 3 studies the set of optimal 4-bit S-boxes for which the 3-share TI can be applied. In Section 4 the *Factorization Structure* is introduced. In Section 5 our new ideas are applied to the SERPENT cipher. Section 6 concludes the paper.

2 The Threshold Implementation Countermeasure

In this section we recall the preliminaries of the Threshold implementation countermeasure and revisit the results of [13] describing a 3-share TI of PRESENT.

[1] This result is independently found from that of http://eprint.iacr.org/2012/300.pdf which was accepted in CHES2012.

2.1 Threshold Implementation Countermeasure

In [11], the Threshold Implementations(TI) was introduced as a side channel attack countermeasure. It is based on secret sharing and multiparty computation and provable secure against the 1-st order DPA, even in the presence of glitches. The method can be described as follows. The variable x is divided into s shares x_i, $1 \leq i \leq s$, such that $x = \bigoplus_{i=1}^{s} x_i$. Let

$$F(x,\ y,\ z,\ \ldots) : GF(2)^m \rightarrow GF(2)^n$$

be a vectorial boolean function which needs to be shared. Denote $\bar{x_i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_s)$, i.e, the vector $\bar{x_i}$ does not contain the share x_i. In order to share F, a set of s vectorial boolean functions F_i is constructed and must fulfill the following properties:

1. **Non-completeness:** All the function F_i must be independent to the input variables $x,\ y,\ z,\ \ldots$, i.e the inputs of F_i does not have $x_i,\ y_i,\ z_i,\ \ldots$or $F_i = F_i(\bar{x_i})$.
2. **Correctness:** $F(x,\ y,\ z,\ \ldots) = \bigoplus_{i=1}^{s} F_i(\bar{x_i},\ \bar{y_i},\ \bar{z_i},\ \ldots)$.

According to the Theorem 2 and 3 of [11], if the inputs satisfy the following condition

$$Pr(\bar{x} = \bar{X}, \bar{y} = \bar{Y}, \ldots) = qPr(x = \bigoplus_{i}^{s} X_i, y = \bigoplus_{i}^{s} Y_i, \ldots), \tag{1}$$

where q is a constant, then the shared version of function F can resist against the first order DPA in presence of glitches.

In general, the function F is a S-box layer and the output of the previous round of the cipher is the input of the current round. Hence, we have a following property for the output of F. Assume that, the output of F is $u = (u_1, u_2, \ldots, u_i)$, then we have the following property for output of F:

Uniformity: $Pr(\bar{u} = \bar{U} | u = \bigoplus_{i}^{s} U_i)$ is a constant.

If the function $u = F(x)$ is invertible, then every vector $\bar{u}$ is reached for exactly one input vector $\bar{x}$. In this paper, the function F is a 4-bit S-box. Hence, its 3-share version is required to be 12-bit permutation.

The number of shares s depends on the degree of the original vectorial boolean function $F(x,\ y,\ z,\ \ldots)$. Assume that the degree of F is d, then s is computed as follows:

Theorem 1. *[12] The minimum number of shares required to implement a product of d variables satisfying Property 1 and 2 is given by*

$$s \geq 1 + d$$

Since the minimum degree of a nonlinear vectorial boolean function is 2, the number of shares s is at least 3. The more shares are needed, the bigger the hardware implementation. Therefore, the 3-share TI is the most efficient case.

2.2 3-share TI for cubic 4-bit S-boxes

In [13] the authors describe how to apply the 3-share TI to PRESENT. Since the PRESENT S-box $S(\cdot)$ is a cubic 4-bit permutation, the minimum number of shares is 4 [12]. To apply TI with only 3 shares, the authors decomposed the S-box into two quadratic permutations $S(\cdot) = F(G(\cdot))$ as shown in Figure 1.

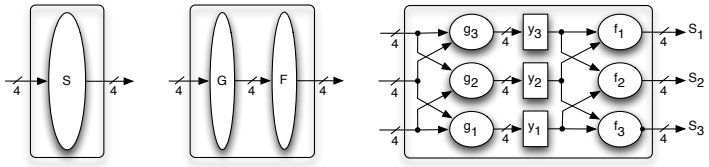

Fig. 1. Composition of PRESENT's s-box [13]

3 TI-Decomposition

In the previous section we recalled how the authors of [13] used the decomposition of a cubic function into two quadratic ones to reduce the number of needed shares in a TI from 4 to 3. In this section we want to investigate the decomposability of optimal 4-bit S-boxes in a more general way. First, we recall the definition of optimal S-boxes and decompositions. Then we define all optimal 4-bit S-boxes which can be decomposed into quadratic boolean functions, i.e., belong to A_{16}. In addition, we describe how to construct the decompositions for all these S-boxes with a time complexity of 2^{19}.

3.1 Optimal 4-bit S-boxes

Definition 1. *Two sboxes $S(x), S^{'}(x)$ are linearly equivalent iff there exist two 4×4-bit invertible matrices A, B and two 4-bit vector c, d such that*

$$S^{'}(x) = A(S(Bx \oplus c) \oplus d), \forall x \in \{0, \ldots, 15\}$$

An S-box is considered as optimal if it fulfills the following requirements [6]:

Definition 2. *Let $S : F_2^4 \rightarrow F_2^4$ be an S-box. If S fulfills the following conditions we call S an optimal S-box:*

1. *S is a bijection,*
2. *$Lin(S) = 8$,*
3. *$Diff(S) = 4$,*

where Lin(S) and Diff(S) are the linearity and differentiality of S-box S. The reader is refered to [6] for the definitions of Lin(S) and Diff(S).

In addition the authors define 16 classes of linearly equivalent S-boxes in S_{16}, i.e class 0, class 1, ..., class 15. An optimal S-boxes belongs to a certain class and each class can be represented by using one its S-box.

For the sake for convenience, we follow the notations in [6]. We write the 4×4-bit matrix A in the hexadecimal, for example:

$$A = \begin{pmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,1\,1 \end{pmatrix} = \begin{pmatrix} a \\ 4 \\ 8 \\ b \end{pmatrix} = \left(0xb84a\right), \tag{2}$$

By utilizing definition 1 we do not have to investigate all $S^{'}(\cdot)$ but we can focus our investigations on one representative $S(\cdot)$ of each class, i.e., if $S(\cdot)$ belongs to class i, then all the results of $S(\cdot)$ can be transferred to $S^{'}(\cdot)$, $i = 0, \ldots, 15$.

3.2 Decomposition of Optimal 4-bit S-boxes

Definition 3. *If a vectorial boolean function $S(\cdot)$ can be written as a composition of several lower degree vectorial boolean functions $F_1(\cdot), F_2(\cdot), \ldots, F_n(\cdot)$, i.e $S(\cdot) = F_n(\ldots F_2(F_1(\cdot)))$, then $F_1(\cdot), F_2(\cdot), \ldots, F_n(\cdot)$ is called the decomposition of $S(\cdot)$.*

We recall some properties of a permutation in S_{16}.

Lemma 1. *A_{16} is a subgroup of S_{16}, i.e if $p_1(\cdot)$ and $p_2(\cdot)$ are permutations in A_{16} then its composition permutation $p_3(\cdot) = p_1(p_2(\cdot))$ must be in A_{16} as well.*

Lemma 2. *All the linear and quadratic permutations in S_{16} are in A_{16}.*

Proof. In [13], the author stated that there are around 2^{26} quadratic permutations. Since the number of the linear and quadratic permutations is not big, we can check the parity of all these permutations. If a permutation has a parity of +1, then it belongs to A_{16} (i.e., it is an even permutation). If its parity equals -1, then it does not belong to A_{16} (i.e., it is an odd permutation). All the considered permutations have the parity +1. It implies that these permutations belong to A_{16}.

Theorem 2. *If a permutation $p(\cdot)$ is able to be represented as a composition of quadratic permutations, then $p(\cdot)$ is in A_{16}.*

Proof. The theorem is directly derived from the Lemma 1 and Lemma 2.

Note 1. The composition of a quadratic permutation and a linear permutation is a quadratic one. Hence, a quadratic permutation is able to be described as a composition of linear and quadratic permutations.

Based on Note 1, if one S-box of a considered class can be decomposed, then all S-boxes of this class can be decomposed as well. After checking the parities of permutation of all 16 representative S-boxes, the decomposable classes are: 0, 1, 2, 4, 5, 7, 8, 13 because their representative have parity +1. For example, the PRESENT S-box can be described as a composition of quadratic permutations because it belongs to class 1.

As described in 2.2, after the decomposition into several 4-bit quadratic permutations we need to convert each of these 4-bit permutations into 12-bit permutations. These 12-bit quadratic permutations have to fulfill all 3 requirements for a Threshold Implementation [12], i.e., non- completeness, correctness and the uniformity property.

Definition 4. *A 4-bit linear or quadratic permutation is called* sharable *if it can be converted to a 12-bit permutation which fulfills the following properties: correctness, non-completeness and uniformity. All linear permutations are sharable [12].*

Definition 5. *A 4-bit permutation is called* TI-decomposable *if it can be described as a composition of several* sharable *permutations.*

Our aim is to prove that all S-boxes belonging to the 8 classes 0, 1, 2, 4, 5, 7, 8, 13 are TI-decomposable S-boxes. In order to prove this, we shall show that there exist one TI-decomposable S-box in each class - all S-boxes in one class can be constructed by applying only linear permutations 1 and all linear permutations are sharable 4.

For an arbitrary S-box to be TI-decomposable it must belong to A_{16} (note that the reverse condition is not necessarily true). Also, an S-box $S(\cdot)$ might not always be decomposable into two quadratic permutations $F(G(\cdot))$, but one has to find a decomposition with at least three quadratic permutations $F(\cdot)$, $H(\cdot)$, $G(\cdot)$ such that $S(\cdot) = F(H(G(\cdot)))$. Even if we know that an S-box can be decomposed into three quadratic permutations, the time complexity for finding a solution for $F(\cdot)$, $H(\cdot)$ and $G(\cdot)$ is not practical, i.e., more than 2^{52} by using the method described in [13].

Therefore, we need a more efficient method to find a decomposition of arbitrary optimal S-boxes in A_{16}. We introduce the following lemma which not only solves this problem but also gives a deep insight into decomposition of S-boxes.

Let R be the following sharable permutation ,

$R = [\ 0, 4, 1, 5, 2, 15, 11, 6, 8, 12, 9, 13, 14, 3, 7, 10\ ]$.

We introduce a special structure of a decomposition:

$$S(\cdot) = A_n F(A_{n-1} F(....A_0(F(\cdot))...)$$

, where $A_n, \ldots, A_0$ are invertible matrices and $S(\cdot)$, $F(\cdot)$ are two vectorial boolean functions. We recall, based on 5, if $F(\cdot)$ is a sharable permutation then $S(\cdot)$ is a TI-decomposable S-box. We make the following observation:

1. if $A = 0x1249$, then $S(\cdot) = R(AR(\cdot)) \in$ class 0

2. if $A = 0x1248$, then $S(\cdot) = R(AR(\cdot)) \in$ class 1
3. if $A = 0x1259$, then $S(\cdot) = R(AR(\cdot)) \in$ class 2
4. if $A = 0x1295$, then $S(\cdot) = R(AR(\cdot)) \in$ class 8
5. if $A = 0x12e6$, then $S(\cdot) = R(AR(R(\cdot))) \in$ class 4
6. if $A = 0x1843$, then $S(\cdot) = R(AR(R(\cdot))) \in$ class 7
7. if $A = 0x134b$, then $S(\cdot) = R(AR(R(\cdot))) \in$ class 13
8. if $A = 0x14a7$, then $S(\cdot) = R(AR(R(R(\cdot)))) \in$ class 5

With this idea we are able to construct a representative TI-decomposable S-box for all latter mentioned 8 classes. Hence, based on 1 and Note 1 we are able construct all possible optimal S-boxes belonging to these 8 classes by adapting A, B, c and d. This way we can reduce the time complexity to find a decomposition for a given S-box to 2^{19}. Using this trick we derive the following lemma:

Lemma 3. *Let $F_i(\cdot)$, $1 \leq i \leq 4$, be sharable permutations. Then,*

1. *For any optimal S-boxes $S(\cdot)$ in the classes 0, 1, 2, 8, there exist sharable permutations $F_1(\cdot)$ and $F_2(\cdot)$ such that $S(\cdot) = F_1(F_2(\cdot))$.*
2. *For any optimal S-boxes $S(\cdot)$ in classes 4, 7, 13, there exist no sharable permutations $F_1(\cdot)$ and $F_2(\cdot)$ such that $S(\cdot) = F_1(F_2(\cdot))$ (proven by exhaustive search), but there exist $F_1(\cdot)$, $F_2(\cdot)$, $F_3(\cdot)$ such that $S(\cdot) = F_1(F_2(F_3(\cdot)))$.*
3. *For any optimal S-boxes $S(\cdot)$ in class 5, there exist no sharable permutations $F_1(\cdot)$ and $F_2(\cdot)$ such that $S(\cdot) = F_1(F_2(\cdot))$ (proven by exhaustive search), but there exist $F_1(\cdot)$, $F_2(\cdot)$, $F_3(\cdot)$, $F_4(\cdot)$ such that $S(\cdot) = F_1(F_2(F_3(F_4(\cdot))))$. Note that there might be a solution such that $S(\cdot) = F_1(F_2(F_3(\cdot)))$ belongs to class 5, but due to the time complexity we could not find such a solution by exhaustive search. Anyway, here the only goal was to construct an S-box belonging to class 5 out of an arbitrary number of sharable decompositions.*

Theorem 3. *All S-boxes which belong to classes 0, 1, 2, 4, 5, 7, 8, 13 are TI-decomposable.*

Proof. This theorem is directly derived from Lemma 3.

Out of the 16 classes there remain 8 classes which are not decomposable, i.e., all these S-boxes do not belong to A_{16}. Hence, there is no method known so far on how to apply the 3-share TI to these S-boxes. In the next section, we shall introduce a new methodology to solve this open problem.

4 Factorization

In the previous section we were left with the representatives of the 8 remaining classes: 3, 6, 9, 10, 11, 12, 14 and 15. We know that these representatives are odd permutations, i.e., they do not belong to A_{16} and hence can not be decomposed. In this section we will introduce a new methodology to solve this problem.

First, let us recall the two following lemmas.

Lemma 4. *The composition of an odd permutation and an even permutation is an odd permutation.*

Proof. The following facts are well-known: the parity of a permutation equals the product of the parities of its decomposed permutations. The parity of an odd permutation is -1, the parity of an even permutation is +1. Hence, the parity of a composition of an odd permutation and an even permutation is -1, i.e., an odd permutation.

Lemma 5. *The 4-bit cubic permutation* $\alpha(x) = (x + 1)\%16$, $0 \leq x \leq 15$, *i.e* $\alpha(\cdot)$ *is modulo-addition over finite field* F_{16}, *is an odd permutation.*

Proof. The permutation parity of $\alpha(\cdot)$ *is -1. It implies that* $\alpha(\cdot)$ *is an odd permutation.*

Denote $G_i(\cdot)$ [6] the representatives of class i and $H_i(\cdot)$ permutations such that $G_i(\cdot) = \alpha(H_i(\cdot))$, $i = 3, 6, 9, 10, 11, 12, 14, 15$. According to the Lemmas 4 and 5, $H_i(\cdot)$ must be an even permutation.

The way to our solution consists of two steps:

1. Prove that all $H_i(\cdot)$ are TI-decomposable.
2. Factorize $\alpha(\cdot)$.

If we can solve both steps, we can apply TI for all $G_i(\cdot)$ of the remaining 8 classes.

4.1 All $H_i(\cdot)$ are TI-decomposable

Lemma 6. *For all* $H_i(\cdot)$ *above, there exist no sharable permutations* $F_1(\cdot)$, $F_2(\cdot)$ *such that* $H_i(\cdot) = F_1(F_2(\cdot))$, *but there exist* $F_1(\cdot)$, $F_2(\cdot)$, $F_3(\cdot)$ *such that* $H_i(\cdot) = F_1(F_2(F_3(\cdot)))$.

Proof. In Table 6 we provide the sharable permutations $F_1(\cdot)$, $F_2(\cdot)$, $F_3(\cdot)$ *with* $F_2(\cdot) = F_3(\cdot)$ *such that* $H_i(\cdot) = F_1(F_2(F_3(\cdot)))$ *for all* $H_i(\cdot)$. *The permutations* $F_1(\cdot)$, $F_2(\cdot)$ *(*$F_3(\cdot)$ *respectively) are written as a sequence of 16 hexadecimal digits. For example* $F_1 = de07f8213ba659c4$ *equals*

F_1 *= [0xd, 0xe, 0x0, 0x7, 0xf, 0x8, 0x2, 0x1, 0x3, 0xb, 0xa, 0x6, 0x5, 0x9, 0xc, 0x4]*

4.2 Factorize $\alpha(\cdot)$

We make the following observation: any given vectorial boolean function $S(\cdot)$ can be written as follows:

$$S(\cdot) = U(\cdot) \oplus V(\cdot),$$

where $\oplus$ is the bitwise *XOR* operation and $U(\cdot)$, $V(\cdot)$ are vectorial boolean function. We call this method **TI-Factorization**.

We may factorize any given optimal 4-bit S-box by using at least 3 quadratic vectorial boolean functions as follows and the idea is described in Figure 4.2:

Table 1. The F_1 and F_2 (F_3 respectively) for all H_i

H_i	F_1	F_2
3	de07f8213ba659c4	8c04159d72fa63eb
6	fe70d812396a5b4c	8c04159d63eb72fa
9	163d47f52a98c0eb	04268cea7351bfd9
10	138eba279f4605dc	0d481c5937eb26fa
11	14a9de0523f8cb76	028aec64935fb17d
12	1a95d04e68b2f73c	039b128a5ed74fc6
14	1af5b04e862d79c3	038a129bf57ce46d
15	10fd287e9c35a4b6	0a1b38295647fdec

1. Construct $U(\cdot)$ such that it contains all cubic terms of the ANF of $S(\cdot)$.
2. Find two vectorial boolean functions such that $U(\cdot) = F(G(\cdot))$
3. Compute $V(\cdot) = S(\cdot) \oplus U(\cdot)$

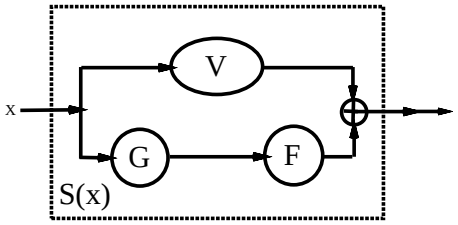

Fig. 2. Factorization Structure.

Note 2. To use the idea of factorization in the context of TI, the 3 properties of [11] - correctness, non-completeness and uniformity - have to be considered. For the properties to be fulfilled, $G(\cdot)$ must be a sharable permutation.

Definition 6. *A 4-bit linear or quadratic vectorial boolean function is called* sharable *if it can be converted to a 12-bit vectorial boolean function which fulfills the following properties: correctness and non-completeness. All 4-bit vectorial boolean function are sharable.*

Definition 7. *A 4-bit permutation is called* TI-factorizable *if it can be described as a composition of several sharable vectorial boolean functions and its converted 12-bit vectorial boolean function is a 12-bit permutation.*

Note 3. TI-decomposable S-boxes are a subset of TI-factorizable S-boxes.

Denote $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \alpha(x, y, z, w)$, where x, y, z, w, α_i, $1 \leq i \leq 4$, are in F_2. The ANF of $\alpha(\cdot)$ is

$$\begin{aligned}\alpha_1 &= x \oplus yzw \\ \alpha_2 &= y \oplus zw \\ \alpha_3 &= z \oplus w \\ \alpha_4 &= w \oplus 1\end{aligned}$$

Now, we show that the permutation $\alpha(\cdot)$ is TI-factorizable. In order to factorize $\alpha(\cdot) = F(G(\cdot)) \oplus V(\cdot)$, we use 3 sharable vectorial boolean functions $(a, b, c, d) = G(x, y, z, w)$ (a sharable permutation), $(A, B, C, D) = F(a, b, c, d)$ and $(X, Y, Z, W) = V(x, y, z, w)$ as follows:

ANF of $G(\cdot)$:

$$\begin{aligned}a &= x \oplus yz \\ b &= y \\ c &= z \\ d &= w\end{aligned}$$

ANF of $F(\cdot)$:

$$\begin{aligned}A &= ad \\ B &= 0 \\ C &= 0 \\ D &= 0\end{aligned}$$

and ANF of $V(\cdot)$:

$$\begin{aligned}X &= x \oplus xw \\ Y &= y \oplus zw \\ Z &= z \oplus w \\ W &= w \oplus 1\end{aligned}$$

In the appendix we describe the construction of the 12-bit permutation $\alpha_{12}(\cdot)$ of $\alpha(\cdot)$. Since $\alpha_{12}(\cdot)$ is a 12-bit permutation, $\alpha(\cdot)$ is TI-factorizable 7.

Note 4. The 3-share versions of $V(\cdot)$ and $F(\cdot)$ satisfy the non-completeness and correctness properties and their inputs are uniformly distributed, i.e., the equation 1 is satisfied. Therefore, the 3-share versions of $V(\cdot)$ and $F(\cdot)$ are secure to first order DPA. Since the 3-share version of $\alpha(\cdot)$ fulfills all required properties, i.e. non-completeness, correctness and uniformity and its input also satisfies the equation 1, the 3-share version of $\alpha(\cdot)$ is secure to first order DPA as well.

We now have solved both problems defined in the beginning of this chapter: we have proven that all H_i are TI-decomposable and $\alpha(\cdot)$ is TI-factorizable. From 3 it follows that all $G_i(\cdot)$ are also TI-factorizable and hence, the TI can be applied to all of the remaining 8 classes. Based on our results we derive the following theorem.

Theorem 4. *All 4-bit optimal S-boxes in the symmetric group S_{16} are TI-factorizable. It implies that all these S-boxes can be protected by using the 3-share TI.*

Note 5. We can generalize the factorization structure for 4-bit permutation as follows. It is always true that, there exists a set of quadratic vectorial boolean functions F_i, V and quadratic permutation G_i, $1 \leq i \leq n$, such that, for any 4-bit S-box $S(\cdot)$:

$$S(\cdot) = \bigoplus_{i=1}^{n} F_i(G_i(\cdot)) \oplus V(\cdot).$$

The Figure 4.2 describes the general idea. Hence, it may be possible to directly construct the 12-bit permutation $S_{12}(\cdot)$ of a given 4-bit cubic S-box $S(\cdot)$ without taking the detour using $\alpha(\cdot)$. We used $\alpha(\cdot)$ for the sake of clarity.

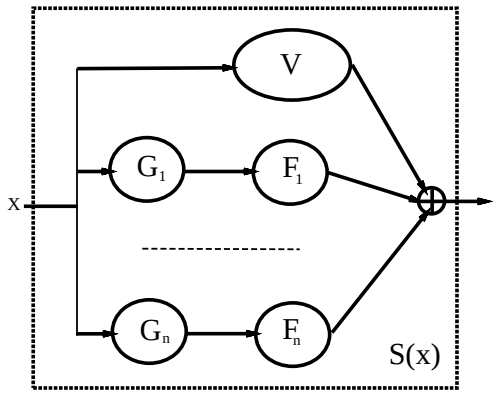

Fig. 3. General Factorization Structure.

5 Applications of TI-decomposable and TI-factorizable S-boxes

In [6] the authors propose an idea stating that two linear equivalent S-boxes can share the same core, i.e., the same class representative G_i, to save implementation

costs. Unfortunately, it is not possible to share a core between different equivalent classes. We pick up this idea and show that by using our new methodologies we can overcome this shortcoming.

In particular, we want to highlight two major benefits of our methodologies. First, using different compositions of the same sharable permutation enables us to construct representatives of all classes in A_{16}. We can use this fact to improve the idea of [6] and use the same core for all S-boxes in A_{16}. By using factorization we can even extend this idea further to classes not in A_{16}.

Second, since all these S-boxes, independent of their class, are now constructed by only using linear and quadratic permutations and/or vectorial boolean functions, and can be used for a 3-share Threshold Implementation.

In the following we want to give an example of how to apply our methodologies. We chose SERPENT because it has 8 different S-boxes of which 4 belong to A_{16} and the other 4 do not belong to A_{16}. This makes SERPENT the perfect example to show the application of our methodologies.

5.1 SERPENT

Let $S \sim S^{'}$ denote S is linear equivalent to $S^{'}$, and let G_i be a representative of class i. The equivalences of the 8 SERPENT S-boxes are as follows [6]:

$$\begin{aligned} S_0 &\sim G_2 \\ S_1 &\sim G_0 \\ S_2 &\sim S_6 \sim G_1 \\ S_3 &\sim S_7 \sim G_9 \\ S_4 &\sim S_5 \sim G_{14} \end{aligned} \tag{3}$$

As we can see there are 5 equivalent classes, meaning by using the idea of [6] we need to implement 5 cores.

Let us recall some results from section 3.2. Let R be the following sharable permutation:

$R = [\ 0, 4, 1, 5, 2, 15, 11, 6, 8, 12, 9, 13, 14, 3, 7, 10\ ]$.

1. if $A = 0x1249$, then $S(\cdot) = R(AR(\cdot)) \in$ class 0
2. if $A = 0x1248$, then $S(\cdot) = R(AR(\cdot)) \in$ class 1
3. if $A = 0x1259$, then $S(\cdot) = R(AR(\cdot)) \in$ class 2

Based on these results, instead of needing 3 cores G_0, G_1, G_2 to construct the 4 S-boxes S_0, S_1, S_2, S_6, we only need to implement $R(\cdot)$ and the matrices $0x1249$, $0x1248$ and $0x1259$ to construct 4 class representatives. Then, we construct the final 4 S-boxes S_0, S_1, S_2 and S_6 by applying $S_i = \hat{A}(S(\hat{B}x \oplus c) \oplus d)$ 1 with the parameters provided in Table 2.

Moreover, we make the following observation for the SERPENT S-boxes which do not belong to A_{16}.

Table 2. The parameters $\hat{A}$, $\hat{B}$, $\hat{c}$, $\hat{d}$ of the S-boxes S_0, S_1, S_2, S_6 of SERPENT

	$\hat{A}$	$\hat{B}$	$\hat{c}$	$\hat{d}$	class
SERPENT S_0 [1]	0x4659	0x3f98	0xa	0x2	2
SERPENT S_1 [1]	0xd597	0xc43a	0xf	0x8	0
SERPENT S_2 [1]	0xbd87	0x2418	0xe	0x1	1
SERPENT S_6 [1]	0x5978	0xce96	0x7	0xa	1

1. if $A = 0x1529$, then $S(\cdot) = R(AR(R(\cdot))) \sim H_9$
2. if $A = 0x1c38$, then $S(\cdot) = R(AR(R(\cdot))) \sim H_{14}$

Where $G_i = (H_i + 1)\%16$, i = 9, 14. Hence, we can construct H_9 and H_{14} by only using $R(\cdot)$, the matrices $0x1529$, $0x1c38$, and by applying $H_i = \tilde{A}(S(\tilde{B}x \oplus \tilde{c}) \oplus \tilde{d})$ the $R(\cdot)$ with the parameters provided in Table 3. Recall that H_i is not necessarily a representative of class i but is needed to construct the representative G_i of class i.

Table 3. The parameters $\tilde{A}$, $\tilde{B}$, $\tilde{c}$, $\tilde{d}$ of H_9, H_{14} of SERPENT

	$\tilde{A}$	$\tilde{B}$	$\tilde{c}$	$\tilde{d}$
SERPENT H_9	0x4896	0x62e3	0xe	0xd
SERPENT H_{14}	0xba4d	0xb8da	0xf	0x1

To construct the 4 remaining SERPENT S-boxes S_3, S_4, S_5, S_7, we now need to apply $S_i = A(\alpha(H_i)(Bx \oplus c) \oplus d)$ with the parameters provided in Table 4.

Table 4. The parameters A, B, c, d and class of some S-boxes

	A	B	c	d	class
SERPENT S_3 [1]	0xfbc5	0xbaf6	0x9	0xe	9
SERPENT S_4 [1]	0xa98d	0x8147	0xb	0x9	14
SERPENT S_5 [1]	0xad89	0x124e	0x0	0x8	14
SERPENT S_7 [1]	0x8947	0x427f	0x6	0x4	9

Taking all results together, instead of implementing 8 S-boxes or 5 cores, we are now able to implement all S-boxes using only one core $R(\cdot)$ (which is sharable permutation), the function $\alpha(\cdot)$ and linear transformations with the parameters provided in this paper. More importantly, it is now possible to apply the 3-share TI to all SERPENT S-boxes, which up to now, was believed to be not possible for the ones not in A_{16}.

5.2 Generic algorithm to factorize S-boxes $\notin A_{16}$

To factorize a given optimal S-box $S(\cdot)$ which is not in A_{16} we perform the following steps:

1. Determine the class of the S-box $S(\cdot)$, i.e., find A, B, c, d such that $S(x) = A(G_i(Bx \oplus c) \oplus d)$.
2. Look up the corresponding F and G in Table 6, i.e., $G_i(\cdot) = \alpha(F(G(\cdot)))$.
3. The given S-box $S(x)$ is factorized as follows:

$$S(x) = A(\alpha(F(G(Bx \oplus c))) \oplus d)$$

In Table 5 we provide some parameters for several 4-bit S-boxes which do not belong to A_{16}.

6 Conclusion

The 3-share TI is an efficient and well-accepted countermeasure against DPA. Until now, it was only possible to apply this countermeasure to optimal 4-bit S-boxes which belong to A_{16} (roughly 50 % of all optimal S-boxes).

First, we give a deep insight into the decomposition of optimal S-boxes and provide a methodology to share a mutual core between all S-boxes in A_{16}. This technique allows for very efficient (in terms of size) hardware implementations. Based on this methodology, we present a new algorithm which can find a decomposition of any given S-box in A_{16} with a time complexity of 2^{19}.

Second, we introduce a new methodology called factorization. This idea enables us to decompose S-boxes not belonging to A_{16}. We show that it is now possible to share a mutual core between all optimal S-boxes allowing the most efficient implementations. More importantly though, since all S-boxes are factorized to linear and quadratic vectorial boolean functions only, we can apply the 3-share TI to all optimal S-boxes, which was believed to be not possible until now.

References

1. Eli Biham, Ross Anderson, and Lars Knudsen. Serpent: A new block cipher proposal. In *In Fast Software Encryption 98*, pages 222–238. Springer-Verlag, 1998.
2. Jean-Sébastien Coron and Louis Goubin. c. In *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, CHES, pages 231–237, London, UK, UK, 2000. Springer-Verlag.
3. Peter Schweitzer Daniel Engels, Markku-Juhani O. Saarinen and Eric M. Smith. The hummingbird-2 lightweight authenticated encryption algorithm.
4. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.
5. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.
6. G. Leander and A. Poschmann. On the classification of 4 bit s-boxes. In *Proceedings of the 1st international workshop on Arithmetic of Finite Fields*, WAIFI '07, pages 159–176, Berlin, Heidelberg, 2007. Springer-Verlag.
7. Stefan Mangard. Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints. In *Systems CHES 2005, 7th International Workshop*, pages 172–186. Springer, 2005.
8. Stefan Mangard, Berndt M. Gammel, and Infineon Technolgies Ag. Side-channel leakage of masked cmos gates. In *in Topics in Cryptology - CT-RSA 2005, The Cryptographers Track at the RSA Conference 2005*, pages 351–365. Springer, 2005.
9. 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.
10. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked aes hardware implementations. In *CHES 2005*, pages 157–171, 2005.
11. Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.
12. Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *Journal of Cryptology*, pages 292–321, October 2010.
13. Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2,300 ge. *J. Cryptol.*, 24(2):322–345, April 2011.

Appendix

In this section we will prove that $\alpha_{12}(\cdot)$ is 12-bit permutation. Based on $F(\cdot)$, $G(\cdot)$, $V(\cdot)$, the 12-bit permutation $\alpha_{12}(\cdot)$ of $\alpha(\cdot)$ is constructed as follows:

The 4-bit input x, y, z, w is split into 3 shares, i.e., $x = x_1 \oplus x_2 \oplus x_3$, $y = y_1 \oplus y_2 \oplus y_3$, $z = z_1 \oplus z_2 \oplus z_3$, $w = w_1 \oplus w_2 \oplus w_3$. Hence, the twelve bit input now is x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3, w_1, w_2, w_3.

The ANF of 12-bit $G_{12}(\cdot)$ of $G(\cdot)$ is:

$$
\begin{aligned}
a_1 &= x_2 \oplus y_2 z_2 \oplus y_2 z_3 \oplus y_3 z_2 \\
a_2 &= x_3 \oplus y_3 z_3 \oplus y_1 z_3 \oplus y_3 z_1 \\
a_3 &= x_1 \oplus y_1 z_1 \oplus y_1 z_2 \oplus y_2 z_1 \\
b_1 &= y_2 \\
b_2 &= y_3 \\
b_3 &= y_1 \\
c_1 &= z_2 \\
c_2 &= z_3 \\
c_3 &= z_1 \\
d_1 &= w_2 \\
d_2 &= w_3 \\
d_3 &= w_1
\end{aligned}
$$

The ANF of 12-bit $F_{12}(\cdot)$ of $F(\cdot)$ is:

$$
\begin{aligned}
A_1 &= a_2 d_2 \oplus a_2 d_3 \oplus a_3 d_2 \\
A_2 &= a_3 d_3 \oplus a_1 d_3 \oplus a_3 d_1 \\
A_3 &= a_1 d_1 \oplus a_1 d_2 \oplus a_2 d_1 \\
B_1 &= 0 \\
B_2 &= 0 \\
B_3 &= 0 \\
C_1 &= 0 \\
C_2 &= 0 \\
C_3 &= 0 \\
D_1 &= 0 \\
D_2 &= 0 \\
D_3 &= 0
\end{aligned}
$$

The ANF of 12-bit $V_{12}(\cdot)$ of $V(\cdot)$ is:

$$\begin{aligned}
X_1 &= x_2 \oplus x_3w_3 \oplus x_2w_3 \oplus x_3w_2 \\
X_2 &= x_3 \oplus x_1w_1 \oplus x_1w_3 \oplus x_3w_1 \\
X_3 &= x_1 \oplus x_2w_2 \oplus x_1w_2 \oplus x_2w_1 \\
Y_1 &= y_2 \oplus z_3w_3 \oplus z_2w_3 \oplus z_3w_2 \\
Y_2 &= y_3 \oplus z_1w_1 \oplus z_1w_3 \oplus z_3w_1 \\
Y_3 &= y_1 \oplus z_2w_2 \oplus z_1w_2 \oplus z_2w_1 \\
Z_1 &= z_2 \oplus w_2 \\
Z_2 &= z_3 \oplus w_3 \\
Z_3 &= z_1 \oplus w_1 \\
W_1 &= w_2 \oplus 1 \\
W_2 &= w_3 \\
W_3 &= w_1
\end{aligned}$$

Then $\alpha_{12}(\cdot) = F_{12}(G_{12}(\cdot)) \oplus V_{12}(\cdot)$ is a 12-bit permutation.

Table 5. The parameters A, B, c, d and class of some S-boxes

	A	B	c	d	class
HB2 S0 [3]	0x8749	0x42ef	0x7	0x9	9
HB2 S1 [3]	0x1e43	0xf8c2	0xb	0x9	10
HB2 S2 [3]	0x8d9a	0x412b	0xc	0x7	14
HB2 S3 [3]	0x3f41	0x76f2	0xe	0x7	15
HB2 $S0^{-1}$ [3]	0xfcb5	0x75fc	0xc	0x1	9
HB2 $S1^{-1}$ [3]	0x59de	0x328e	0xa	0x2	10
HB2 $S2^{-1}$ [3]	0xf314	0xe6f4	0xd	0xc	15
HB2 $S3^{-1}$ [3]	0xa9d8	0x8217	0x7	0x8	14
SERPENT S_3^{-1} [1]	0x7498	0x24ef	0xa	0xb	9
SERPENT S_4^{-1} [1]	0xf431	0xbaf2	0x6	0xd	15
SERPENT S_5^{-1} [1]	0x1f34	0xbaf8	0xe	0x6	15
SERPENT S_7^{-1} [1]	0x5cbf	0xd5f6	0x4	0xd	9