Compact Implementations of Multi-Sbox Designs

Begül Bilgin[1], Miroslav Knežević[2], Ventzislav Nikov[2] and Svetla Nikova[1]

[1] KU Leuven, ESAT-COSIC and iMinds, Belgium
{name.surname}@esat.kuleuven.be
[2] NXP Semiconductors, Belgium {name.surname}@nxp.com

Abstract. Implementations of cryptographic algorithms using several different Sboxes by design are typically considered burdensome. The first reason is that unlike single-Sbox designs, serialized implementations of such cryptographic algorithms require instantiations of all Sboxes which prohibits the desired reduction of area. The second reason is that applying countermeasures such as masking causes an undesired increase in area due to the amount of different nonlinear blocks in the algorithm. In this paper, we propose a novel method to implement multi-Sbox designs using as few nonlinear blocks as possible. We exemplify our finding on DES algorithm of which the Triple-DES variant is still widely used in practice. With this method, it is possible to implement the DES substitution layer, which is composed of eight 6×4 Sboxes, using only three 4-bit nonlinear and several affine 4-bit permutations. Our investigation shows that such an implementation requires less area than the state-of-the-art. Moreover, it opens up the possibilities for compact implementations with countermeasures.

1 Introduction

Technological developments in the field of low-end devices are proceeding at a rapid pace; preserving, however, never ending implementation challenges. Driven by the very fierce constraints, two of those are of utmost importance: silicon area and energy consumption. Based on economical and technical limitations these two constraints remain the key factors in today's evolution of low-cost devices. Due to the linear relationship between silicon area and chip manufacturing costs on one side, and the billions of devices produced every year on the other side, the total production cost is naturally a limiting factor.

To overcome the mentioned challenge, there have been many new lightweight block ciphers coming out of the academic research [2, 7, 14, 16, 20, 28, 29]. In practice, however, there are still billions of devices in the field carrying standardized ciphers such as Advanced Encryption Standard (AES) [13], Data Encryption Standard (DES) [24] and the Russian block cipher standard GOST [1] which are not necessarily considered lightweight. The reason for the former block cipher is mainly its high-degree 8-bit Sbox. There has been recent progress [8, 15, 17, 22] in optimizing implementations of AES and its Sbox. However, a very little improvement has been made on the lightweight implementations of DES and GOST. To our knowledge, [20] provides the most significant improvement when lightweight implementation of DES is considered. The mentioned implementation is serialized with 6-bit data-path. The authors emphasize that the cost of the Sboxes is significant and suggest to use one single alternative Sbox. In general, many multi-Sbox designs such as Lblock [30], mCrypton [21], the Advanced Encryption Standard candidates SERPENT [3] and TwoFish [27] share the same faith with DES by lacking lightweight implementations.

Due to the usage of these algorithms in devices, such as ID cards, smart cards, payment cards, the implementations of these algorithms are subject to implementation attacks. The most common implementation attack, namely differential power analysis (DPA), is based on observing the power consumption of the device [19]. Countermeasures, such as masking are suggested for secure implementations [4, 9, 18, 26]. Some masking schemes even provide security under the probing model in addition to DPA. Masking-like countermeasures have the advantage to perform efficiently when affine functions are considered. However, their implementation cost increases together with the degree of the nonlinear function to be protected. Therefore, it is desired to implement a cryptographic algorithm with minimum number of nonlinear functions. Even though there has been ongoing research on minimizing the field multiplications per S-box [11, 12], there has been no progress done on minimizing the number of nonlinear permutations of a cryptographic algorithm by looking at the full substitution layer. This also explains the lack of research on masking of multi-Sbox designs.

Contribution. In this work, we propose a method, based on decompositions of permutations, to reduce the number of nonlinear permutations of existing multi-Sbox designs for efficient hardware implementations. This method not only decreases the area of the unprotected implementation, but also creates a significant advantage for implementing masking-like countermeasures. We illustrate the proposed method on DES. We acknowledge that DES is considered insecure for many applications nowadays due to its 56-bit key length. However, this implementation can be used during the implementation of its successor Triple-DES (which essentially applies the DES algorithm three times with different keys to each input data block). Triple-DES, also denoted as 3DES or TDEA, is still massively used in the electronic payment industry. When the three consecutive instances of DES are applied using three different 56-bit keys, Triple-DES provides effective 112 bits of security. In addition, we provide implementation results and suggest design ideas to cryptographers.

Organization. We introduce the DES algorithm by detailing its subfunctions in the following section. Our method inherits ideas from the affine equivalence relation between the permutations and possible decompositions of them. An introduction to these subjects are also given in Section 2. The details of our method and its instantiation on DES Sboxes are provided in Section 2.3. We discuss our hardware implementation in Section 3 which is followed by the conclusion.

2 Preliminaries

We denote an $m \times n$ Sbox with m input bits and n output bits (from GF(2^m) to GF(2^n)) as $S(x_1, \ldots, x_m) = (y_1 \ldots, y_n)$ and equivalently $S(x) = y$. Alternatively, we also use $S = .$ where . stands for table look-up of the Sbox. We use calligraphic letters to denote a set (e.g. $\mathcal{A}$) and $|.|$ (e.g. $|\mathcal{A}|$) to denote its cardinality.

2.1 Data Encryption Standard (DES)

A single iteration of DES (Fig. 1) takes as input a 64-bit plaintext together with a 56-bit key in order to output a 64-bit ciphertext. It is a balanced Feistel network where only 32 bits of the state are updated per round. The round function RF which takes a 32-bit input and a 48-bit round key follows the steps below:

Expansion. The 32-bit input is divided into eight 4-bit chunks. Each chunk is extended to 6 bits by inheriting the first (resp. last) bit from its adjacent chunk on the right (resp. left). Hence this operation duplicates half of the input bits to generate a 48-bit output.

Key mixing. The 48-bit round key is mixed with the expanded 48 bits by means of XORs.

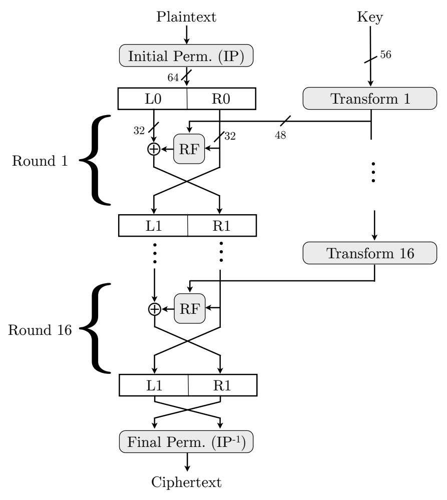

Fig. 1: DES block cipher

Substitution. The 48-bit output of key mixing is split into eight 6-bit chunks. Each 6-bit chunk is substituted to a 4-bit block using a different 6×4 Sbox. Hence, the substitution layer is composed of eight different Sboxes. These Sboxes are designed so that they can be represented as four 4-bit permutations (hereto mini-Sboxes). The outer bits (x_1, x_6) are used to select which mini-Sbox is used whereas the inner bits (x_2, x_3, x_4, x_5) are responsible for the look-up.

Permutation. The 32-bit output of the substitution layer is bit-wise permuted using a special permutation matrix.

Key schedule It is used to generate 48-bit round keys from the 56-bit master key. This generation is simply rotation and selection.

We refer to [10] for more detailed description.

2.2 Affine Equivalence Relation

For our efficient implementation, we mainly consider the 4-bit mini-Sboxes. Since these Sboxes are permutations, we focus on some properties of permutations in this section.

The set of all n-bit permutations form the symetric group $\mathcal{S}_{2^n}$. This set can be split using the affine equivalence relation between Sboxes as defined below:

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

Each group of n-bit permutations that are affine equivalent form a class. Hence, all the Sboxes in a class can be represented with a designated representative Sbox from the same class. Every Sbox in the same class has the same algebraic degree. Moreover, every Sbox in a class are either in the Alternating group $\mathcal{A}_{2^n}$ or not. Being in $\mathcal{A}_{2^n}$ implies that the Sbox can be represented with even number of transpositions. The set of non-Alternating permutations are denoted by $\mathcal{S}_{2^n} \setminus \mathcal{A}_{2^n}$.

Following the notation in [6], we represent linear, quadratic and cubic affine equivalence classes of 4-bit permutations as $\mathcal{A}_i$, $\mathcal{Q}_i$ and $\mathcal{C}_i$ where i stands for the class number when classes are ordered lexicographically. We refer to [6] for the list of all 302 affine-equivalence classes' representatives of 4-bit permutations.

It has been shown in [6] that there exist one affine, six quadratic and 295 cubic classes of 4-bit permutations. It is well known that all the affine permutations are in $\mathcal{A}_{16}$. Moreover, all the quadratic 4-bit permutations are shown to be in $\mathcal{A}_{16}$. From the remaining cubic permutations, only 144 of them are in $\mathcal{A}_{16}$. For the representatives of these specific classes, we refer the reader to [6].

2.3 Decomposition

Decomposition of a cubic Sbox S into two quadratic Sboxes S_1 and S_2, such that $S = S_1 \circ S_2$ is first proposed in [25] for the 4-bit PRESENT Sbox to ease the application of a countermeasure. Later in [5,6] this idea has been generalized to all 4-bit permutations with sometimes more than two Sboxes in decomposition ($S = S_1 \circ S_2 \circ \ldots \circ S_s$) as follows.

Let $\mathcal{M}$ define the set of all six quadratic permutation classes $\{\mathcal{Q}_{004}, \mathcal{Q}_{012}, \mathcal{Q}_{293}, \mathcal{Q}_{294}, \mathcal{Q}_{299}, \mathcal{Q}_{300}\} = \mathcal{M}$. It has been shown that all permutations in $\mathcal{A}_{16}$ can be decomposed using permutations from $\mathcal{M}$. Note that the possibility to decompose some cubic Sboxes to quadratics but not all can be described with the fact that all 4-bit quadratic permutations are in $\mathcal{A}_{16}$ as do the cubic classes with quadratic decomposition. The decomposition length of these cubic classes, which is defined as the minimum number of quadratic permutations used in such decompositions varies between two and four. Naturally, it is also possible to decompose these cubic Sboxes into one cubic and one quadratic permutation from $\mathcal{A}_{16}$.

The rest of the cubic permutations, which are in $\mathcal{S}_{16} \setminus \mathcal{A}_{16}$, can be represented using one cubic ($\in \mathcal{S}_{16} \setminus \mathcal{A}_{16}$) and one or more quadratic permutations ($\in \mathcal{A}_{16}$). In [5, 6], such decompositions are generated using the cubic permutations from a certain set of Sboxes, namely $\mathcal{N} = \{\mathcal{C}_{001}, \mathcal{C}_{003}, \mathcal{C}_{013}, \mathcal{C}_{301}\}$. It has been shown that the permutations in $\mathcal{N}$ have special properties that are advantageous when masking is considered. Even though we do not explicitly use these properties, we choose to use these classes when permutations form $\in \mathcal{S}_{16} \setminus \mathcal{A}_{16}$ are required.

sectionDecomposition to Simplify Multi-Sbox Designs Even thought there exists prior applications of particular Sboxes using decomposition, it is the first time that such a method is applied in order to reduce the implementation overhead of multi-Sbox designs and obtain a compact implementation on hardware. Here, we mainly describe our method on DES for easy understanding and provide implementation results for it. However, this method can be applied to other designs such as GOST [1], SERPENT [3] TwoFish [27], Lblock [30] and mCrypton [21].

2.4 Methodology

As mentioned in Section 2.1, DES comprises of eight different 6×4 Sboxes, which can be represented as 32 different 4×4 mini-Sboxes. This presents a challenge for a compact hardware implementation, since all of them need to be implemented (instantiated) separately.

Idea. In order to minimize the cost of several different Sboxes, we aim to decompose them such that minimum number of nonlinear permutations is used to jointly describe all mini-Sboxes. Therefore, we search for one or more decompositions of the mini-Sboxes that hold for as many Sboxes as possible.

Possible decompositions. It is known that the 32 mini-Sboxes of DES belong to 21 different affine equivalent classes [6]. Nine of these classes are in $\mathcal{A}_{16}$, specifically they are from the set $\{\mathcal{C}_{046}, \mathcal{C}_{073}, \mathcal{C}_{085}, \mathcal{C}_{086}, \mathcal{C}_{148}, \mathcal{C}_{184}, \mathcal{C}_{221}, \mathcal{C}_{254}, \mathcal{C}_{281}\} = \mathcal{K}$. As described in Section 2.3, being in $\mathcal{A}_{16}$ implies that an Sbox in $\mathcal{K}$ can be decomposed into a permutation from $\mathcal{M}$ and a permutation from $\mathcal{A}_{16}$. Given the affine equivalence relation, one can search for a solution of the form

$$
\begin{aligned}
S_{\mathcal{K}} &= C \circ F \circ B \circ G_{\mathcal{K}} \circ A && \text{(left decomposition)} \\
S_{\mathcal{K}} &= C \circ G_{\mathcal{K}} \circ B \circ F \circ A && \text{(right decomposition)} \qquad (1)
\end{aligned}
$$

for the mini-Sboxes in $\mathcal{K}$. Here, $S_{\mathcal{K}}$ is the representative of the targeted mini-Sbox, $G_{\mathcal{K}}$ is a quadratic permutation from one of the classes in $\mathcal{M}$, F is a permutation from $\mathcal{A}_{16}$, and A, B, C are affine permutations.

The remaining 12 mini-Sboxes are in $\mathcal{S}_{16} \setminus \mathcal{A}_{16}$, specifically from the set $\{\mathcal{C}_{059}, \mathcal{C}_{069}, \mathcal{C}_{079}, \mathcal{C}_{098}, \mathcal{C}_{117}, \mathcal{C}_{137}, \mathcal{C}_{139}, \mathcal{C}_{166}, \mathcal{C}_{204}, \mathcal{C}_{220}, \mathcal{C}_{257}, \mathcal{C}_{279}\} = \mathcal{L}$. Similar to the previous case, one can search for a solution of the form

$$\begin{aligned} S_{\mathcal{L}} &= C \circ F \circ B \circ G_{\mathcal{L}} \circ A && \text{(left decomposition)} \\ S_{\mathcal{L}} &= C \circ G_{\mathcal{L}} \circ B \circ F \circ A && \text{(right decomposition)} \end{aligned} \qquad (2)$$

where $S_{\mathcal{L}}$ is the representative of the targeted mini-Sbox, $G_{\mathcal{L}}$ is a permutation from $\mathcal{N}$, F is a permutation from $\mathcal{A}_{16}$, and A, B, C are affine permutations. Note that it is possible to find solutions where $G_{\mathcal{L}}$ is a permutation from $\mathcal{S}_{16} \setminus \mathcal{A}_{16}$ but not strictly from $\mathcal{N}$. Here, we ignore such decompositions for reasons described in Section 2.3.

Goal. We aim to find F in $\mathcal{A}_{16}$ such that it satisfies Equations (1) and (2) and is the same for all these mini-Sboxes.

Algorithm to achieve the goal. We propose the following algorithm (we are considering thereafter the left decomposition only since for the right decomposition the method is similar):

1. Take one representative permutation $S_{\mathcal{K}}$, $S_{\mathcal{L}}$, $G_{\mathcal{K}}$ and $G_{\mathcal{L}}$ for each class in $\mathcal{K}$, $\mathcal{L}$, $\mathcal{M}$ and $\mathcal{N}$ respectively.
2. Compute $D_{\mathcal{K}} = S_{\mathcal{K}} \circ (A)^{-1} \circ (G_{\mathcal{K}})^{-1}$ and $D_{\mathcal{L}} = S_{\mathcal{L}} \circ (A)^{-1} \circ (G_{\mathcal{L}})^{-1}$ for all possible affine permutations A, i.e for all permutations in class $\mathcal{A}_{000}$ where $|\mathcal{A}_{000}| = 322,560$. Since $|\mathcal{K}| = 9$ and $|\mathcal{M}| = 6$, there exists $9 \times 322,560 \times 6$ solutions for $D_{\mathcal{K}}$. Similarly, $|\mathcal{L}| = 12$ and $|\mathcal{N}| = 4$ provides $12 \times 322,560 \times 4$ solutions for $D_{\mathcal{L}}$. In total, we get $9 \times 322,560 \times 6 + 12 \times 322,560 \times 4 = 32,901,120$ solutions for this left decomposition. We cluster them in $9 \times 6 + 12 \times 4 = 102$ groups each containing 322,560 solutions.
3. Cluster the groups in super groups of $9 + 12 = 21$ in total, i.e. for each mini-Sbox class.
4. Search for a class from $\mathcal{A}_{16}$ which is present in all these super groups (recall that $\mathcal{A}_{16}$ contains 151 classes).
5. If such a class exists, find $F \in \mathcal{A}_{16}$, which satisfies the relation $D_{\mathcal{K}} = C \circ F \circ B$ (and correspondingly $D_{\mathcal{L}} = C \circ F \circ B$), and hence Equations (1) and (2).

Note that we abuse the notation for A, B, C constantly which is certainly different for all mini-Sbox representatives $S_{\mathcal{K}}$ and $S_{\mathcal{L}}$.

2.5 Results

Applying the algorithm in Section 2.4, we found 12 solutions for the left decomposition and another 12 for the right decomposition – so in total 24 solutions. The found common class where F lies is one of $\{\mathcal{C}_{158}, \mathcal{C}_{159}\}$. In Table 1, we summarize our solutions. It can be observed that by fixing $\mathcal{C}_{158}$ or $\mathcal{C}_{159}$, we get 4 or 8 solutions respectively for left decomposition (alternatively 8 or 4 solutions respectively for right decomposition).

	Left Decomp.	Right Decomp.
$\mathcal{K}$	$\mathcal{C}_{158} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{012} \times \mathcal{C}_{158}$
	$\mathcal{C}_{158} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{293} \times \mathcal{C}_{158}$
	$\mathcal{C}_{159} \times \mathcal{Q}_{012}$	$\mathcal{Q}_{294} \times \mathcal{C}_{158}$
	$\mathcal{C}_{159} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{299} \times \mathcal{C}_{158}$
	$\mathcal{C}_{159} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{293} \times \mathcal{C}_{159}$
	$\mathcal{C}_{159} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{299} \times \mathcal{C}_{159}$
$\mathcal{L}$	$\mathcal{C}_{158} \times \mathcal{C}_{013}$	$\mathcal{C}_{013} \times \mathcal{C}_{158}$
	$\mathcal{C}_{158} \times \mathcal{C}_{301}$	$\mathcal{C}_{301} \times \mathcal{C}_{158}$
	$\mathcal{C}_{159} \times \mathcal{C}_{013}$	$\mathcal{C}_{013} \times \mathcal{C}_{159}$
	$\mathcal{C}_{159} \times \mathcal{C}_{301}$	$\mathcal{C}_{301} \times \mathcal{C}_{159}$

Table 1: Possible solutions for decomposing DES mini-Sboxes.

However, note that we only considered the *representatives* of the DES mini-Sboxes so far. In order to obtain the final solution for each of the 32 mini-Sboxes, we take into account that they are affine equivalent to the representative $S_{\mathcal{K}}$ (or $S_{\mathcal{L}}$), i.e. $S = C' \circ S_{\mathcal{K}} \circ A'$ (or $S = C' \circ S_{\mathcal{L}} \circ A'$), where S is one of the 32 mini-Sboxes, C' and A' are affine permutations (i.e. from class $\mathcal{A}_{000}$).

Therefore the representation for S becomes $S = C' \circ C \circ F \circ B \circ G_{\mathcal{K}} \circ A \circ A'$ (or $S = C' \circ C \circ G_{\mathcal{K}} \circ B \circ F \circ A \circ A'$) if $S \in \mathcal{A}_{16}$ and $S = C' \circ C \circ F \circ B \circ G_{\mathcal{L}} \circ A \circ A'$ (or $S = C' \circ C \circ G_{\mathcal{L}} \circ B \circ F \circ A \circ A'$) if $S \in \mathcal{S}_{16} \setminus \mathcal{A}_{16}$. Obviously, $C' \circ C$ and $A \circ A'$ can be replaced by one single affine permutation. For simplicity we represent this compositions with C and A respectively and reach the final decomposition for all 32 mini-Sboxes given in Equation (1) and (2) correspondingly.

Hereon, we refer to the j^{th} mini-Sbox of the i^{th} DES Sbox as S^{ij}, where $1 \leq i \leq 8$, $0 \leq j \leq 3$. As described in Equation (1) and (2), these mini-Sboxes belong to either of the two defined cases (denoted hereafter with S_{left}^{ij} and S_{right}^{ij}). Furthermore, each of these mini-Sboxes can be represented as explained in the previous section as a composition of two higher-degree (quadratic or cubic) and three affine vectorial boolean functions (permutations). In other words:

$$S_{left}^{ij} = \begin{cases} C^{ij} \circ F \circ B^{ij} \circ G_{\mathcal{K}} \circ A^{ij}, & \text{if } (i,j) \in \{(1,*), (5,1), (5,2), (5,3), \\ & \text{(6,1), (6,2), (7,3), (8,3)}\} \\ C^{ij} \circ F \circ B^{ij} \circ G_{\mathcal{L}} \circ A^{ij}, & \text{if } (i,j) \in \{(2,*), (3,*), (4,*), (5,0), \\ & \text{(6,0), (6,3), (7,0), (7,1), (7,2), (8,0),} \\ & \text{(8,1), (8,2)}\} \end{cases}$$

$$S_{right}^{ij} = \begin{cases} C^{ij} \circ G_{\mathcal{K}} \circ B^{ij} \circ F \circ A^{ij}, & \text{if } (i,j) \in \{(1,*), (5,1), (5,2), (5,3), \\ & \text{(6,1), (6,2), (7,3), (8,3)}\} \\ C^{ij} \circ G_{\mathcal{L}} \circ B^{ij} \circ F \circ A^{ij}, & \text{if } (i,j) \in \{(2,*), (3,*), (4,*), (5,0), \\ & \text{(6,0), (6,3), (7,0), (7,1), (7,2), (8,0),} \\ & \text{(8,1), (8,2)}\} \end{cases}$$

where $G_{\mathcal{K}}$ is quadratic, and $G_{\mathcal{L}}$ and F are cubic vectorial Boolean functions and A^{ij}, B^{ij}, C^{ij} are affine. As we can see, F, $G_{\mathcal{K}}$, and $G_{\mathcal{L}}$ can be shared for all the mini-Sboxes and therefore make the whole implementation considerably smaller.

Due to page limitations, we provide only one possible tuple $(F, G_{\mathcal{K}}, G_{\mathcal{L}})$ out of 24 tuples that satisfy the solution for the DES Sbox (12 solutions for S_{left}^{ij} and 12 for S_{right}^{ij}) in Table 2. For each tuple, the affine vectorial Boolean functions A^{ij}, B^{ij}, C^{ij} are then uniquely determined. The representations in Table 2 are in the form of lookup tables. The others solutions are equivalent and trivial to derive.

3 Hardware Implementation

We consider a serialized structure for our lightweight implementation which uses one-Sbox block that can calculate all DES Sboxes. The smallest DES implementation known so far, presented in [20] is an example of such a serialized structure. For compatibility, we inherit the mentioned implementation together with the same 6-bit input 4-bit output behavior of the S-box layer in our implementation.

Our implementation does not possess any countermeasures against physical attacks. However, we emphasize that this one-Sbox structure,

Table 2: DES Sbox decomposition (one of the 24 possible representations).

F	0123458A6BCF7D9E	$G_{\mathcal{K}}$	0123457689CDEFBA	$G_{\mathcal{L}}$	0123456789CDEFBA
A^{10}	43DA8F169E0752CB	B^{10}	0145ABEFCD896723	C^{10}	6B0D94F21C7AE385
A^{11}	EDB874213065A9FC	B^{11}	0145ABEFCD896723	C^{11}	618FDA349E7025CB
A^{12}	43168FDA9ECB5207	B^{12}	0145ABEFCD896723	C^{12}	5E92A16DB07C4F83
A^{13}	7F6E2A3B8091D5C4	B^{13}	0514AFBE9C8D3627	C^{13}	B4691EC378A5D20F
A^{20}	2156DEA90374FC8B	B^{20}	02318AB94675CEFD	C^{20}	971F5BD34AC2860E
A^{21}	78D23C96F05AB41E	B^{21}	57021346ECB9A8FD	C^{21}	79F1E0683DB5A42C
A^{22}	7C1AE583294FB0D6	B^{22}	BFC8EA9D26517304	C^{22}	EAD926158CBF4073
A^{23}	D7A0C6B15F284E39	B^{23}	5261E9DA7043CBF8	C^{23}	9E2561DA70CB8F34
A^{30}	E079B52CA43DF168	B^{30}	5F0A4E1BC693D782	C^{30}	F27AD058E36BC149
A^{31}	52BC709EDA34F816	B^{31}	8F439E5261AD70BC	C^{31}	06357142BD8ECAF9
A^{32}	F30C956A48B72ED1	B^{32}	BFC8EA9D26517304	C^{32}	9A12ED658B03FC74
A^{33}	E68091F74C2A3B5D	B^{33}	1A29380BF4C7D6E5	C^{33}	A0E46C285F1B93D7
A^{40}	8FAD0725E9CB6143	B^{40}	5261E9DA7043CBF8	C^{40}	2FC1D03E947A6B85
A^{41}	89AB2301EFCD4567	B^{41}	5261E9DA7043CBF8	C^{41}	8F3470CB61DA9E25
A^{42}	5270DAF83416BC9E	B^{42}	5261E9DA7043CBF8	C^{42}	8F3470CB61DA9E25
A^{43}	5270DAF83416BC9E	B^{43}	5261E9DA7043CBF8	C^{43}	295ED6A1F4830B7C
A^{50}	EA7326BF049DC851	B^{50}	5F0A4E1BC693D782	C^{50}	7AF258D03EB61C94
A^{51}	A8B975643120ECFD	B^{51}	021346578A9BCEDF	C^{51}	AF0514BE72D8C963
A^{52}	7EA3D40918C5B26F	B^{52}	0123456789ABCDEF	C^{52}	7F3BD5914C08E6A2
A^{53}	AE40FB158C62D937	B^{53}	0167EF89CDAB2345	C^{53}	72BEFA369C5014D8
A^{60}	28F56CB10AD74E93	B^{60}	EBC963412705AF8D	C^{60}	A41F972CE05BD368
A^{61}	EFAB1054DC982367	B^{61}	01236745EFCD89AB	C^{61}	C70B834F5E921AD6
A^{62}	A578F02DC31E964B	B^{62}	04152637BFAE9D8C	C^{62}	8CBF4073EAD92615
A^{63}	B9CE46318AFD7502	B^{63}	02318AB94675CEFD	C^{63}	8FADBC9E34160725
A^{70}	F719D53B806EA24C	B^{70}	8F439E5261AD70BC	C^{70}	DEB8A9CF30564721
A^{71}	E284C0A6593F7B1D	B^{71}	8C37AE1504BF269D	C^{71}	BA103298FE5476DC
A^{72}	027564138AFDEC9B	B^{72}	5261E9DA7043CBF8	C^{72}	9E70618F43ADBC52
A^{73}	AC8E1735BD9F0624	B^{73}	02468ACE9BDF1357	C^{73}	E6193BC4A25D7F80
A^{80}	C74F921A38B06DE5	B^{80}	B8FC03471256A9ED	C^{80}	896754BACD2310FE
A^{81}	2A6E91D5B3F7084C	B^{81}	831A29B07CE5D64F	C^{81}	B193280AC6E45F7D
A^{82}	905CF63AE72B814D	B^{82}	BF73AE628C409D51	C^{82}	086E91F7C4A25D3B
A^{83}	56A912EDCF308B74	B^{83}	0167CDAB2345EF89	C^{83}	0437C8FBAE9D6251

which only has three nonlinear permutation blocks F, $G_{\mathcal{K}}$, $G_{\mathcal{L}}$, is highly advantageous when countermeasures such as masking are considered. This is due to the fact that implementing a nonlinear function in masked domain is very challenging whereas a linear function can be implemented in a straight-forward way. Moreover, it has been shown in [4] that the required number of shares for linear functions is smaller than that of nonlinear functions.

3.1 Instantiation of the multi-Sbox design

We focus on the specific solution suggested in Table 2 out of the 24 equivalent choices. Figure 2 represents a block diagram of our DES Sbox architecture. The input of the Sbox consists of 6 bits of which the two outer bits will be used together with the 3-bit clock counter of one round in order to decide which mini-Sbox calculation operation is followed. We denote this selection bits as *sel*. The remaining 4 bits of the input are sent to the first affine permutation layer. Depending on the calculated mini-Sbox, and hence the value of *sel*, the input of the nonlinear function $G_{\mathcal{K}}$ or $G_{\mathcal{L}}$ is selected from the outputs of A^{ij}. The output of $G_{\mathcal{K}}$ (resp. $G_{\mathcal{L}}$) is used as input to the second affine permutation layer. Similar to the previous step, the input of the nonlinear permutation F is chosen from the outputs of B^{ij} with the help of *sel*. The output of F is sent to the final layer of affine permutations C^{ij}. The correct output of the DES Sbox is chosen from the outputs of C^{ij} using the *sel*. There are no registers in this implementation.

Please note that several affine permutation blocks, specifically $A^{43}, B^{11}, B^{12}, B^{32}, B^{40}, B^{41}, B^{42}, B^{43}, B^{50}, B^{63}, B^{70}, B^{72}$ and C^{32} (i.e. 13 in total) are represented with dotted lines and they are not connected. This is due to the fact that these permutation blocks are equivalent to other permutation blocks (Table 3) which can be reused to obtain even smaller footprint.

3.2 Synthesis Results

We implemented the proposed method and all eight 6×4 Sboxes of DES for a fair comparison (i.e. we assumed 6-bit input and 4-bit output including the final multiplexer which is also depicted in Fig 2). We described both options as their corresponding Boolean function representation and let our synthesis tool (Synopsys Design Vision D-2010.03-SP4) optimize these functions using the compile_ultra command. Synthesis results show

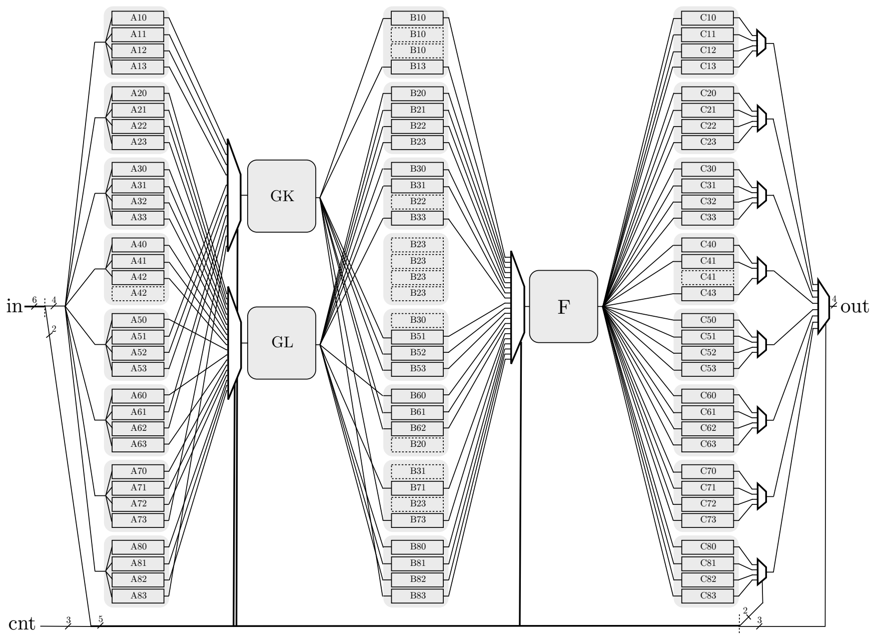

Fig. 2: Substitution layer of DES composed of 8 Sboxes (equivalently 32 mini-Sboxes).

$A^{42} = A^{43}$
$B^{10} = B^{11} = B^{12}$
$B^{40} = B^{41} = B^{42} = B^{43} = B^{23} = B^{72}$
$B^{22} = B^{32}$
$B^{50} = B^{30}$
$B^{63} = B^{20}$
$B^{70} = B^{31}$
$C^{41} = C^{42}$

Table 3: List of affine permutations that are equal.

40% decrease of hardware (1017 GE vs. 603 GE) area when 45nm NanGate standard cell library is used [23].

Moreover, we also implemented DES in a serialized manner as suggested in [20] by replacing the Sbox with ours. The synthesis with the mentioned settings yields 2089 GE. Since our library and the library used by [20] are different as well as the synthesis tools, we can not directly compare the exact GE values. In [20], it has been stated that 32% of the area is consumed by the Sboxes. Following this metric, we conclude that our Sbox which employs 28% of the area is smaller.

We repeated the same analysis with Cadance RTL Compiler RC14.22 and the same library since the performance of the optimization can vary depending on the synthesis tool. We got similar results.

Note that the variety of the affine permutations which also increases the amount of the multiplexers consume a big ratio of the area. We emphasize that it is possible to minimize such cost with careful selection of Sboxes during the design process.

4 Conclusion

We discuss a novel method to describe many permutations of the same size with high degree using fewer smaller degree permutations using the affine equivalence relation between their decompositions. We exemplified our method on DES algorithm for which we used only one quadratic, two cubic 4-bit permutations instead of 32 cubic 4-bit permutations or equivalently eight degree five functions. We leave other instantiations of this work to the reader. Moreover, the mentioned method not only has the advantage of resulting with smaller implementations, but also leads to possible low-cost implementations with countermeasures since it minimizes the number of nonlinear elements in the description.

5 Acknowledgments

This work has been supported in part by the Research Council of KU Leuven (OT/13/071) and by GOA (tense). B. Bilgin was partially supported by the FWO project G0B4213N and she is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

References

1. Gost, gosudarstvennyi standard 28147-89. Cryptographic Protection for Data Processing Systems, Government Committee of the USSR for Standards, 1989.

2. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. In *eprint.iacr.org/2013/404*, 2013.
3. E. Biham, R. J. Anderson, and L. R. Knudsen. Serpent: A new block cipher proposal. In S. Vaudenay, editor, *FSE*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.
4. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. Higher-order threshold implementations. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer Berlin Heidelberg, 2014.
5. B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, and G. Stutz. Threshold implementations of all 3×3 and 4×4 s-boxes. In *Cryptographic Hardware and Embedded Systems-CHES 2012*, Lecture Notes in Computer Science, pages 76–91. Springer Berlin Heidelberg, 2012.
6. B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, N. Tokareva, and V. Vitkup. Threshold implementations of small S-boxes. *Cryptography and Communications*, 7(1):3–33, 2015.
7. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007*, Lecture Notes in Computer Science, pages 450–466. Springer Berlin Heidelberg, 2007.
8. D. Canright. A Very Compact S-Box for AES. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, pages 441–455, 2005.
9. C. Carlet, L. Goubin, E. Prouff, M. Quisquater, and M. Rivain. Higher-order masking schemes for S-Boxes. In A. Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer Berlin Heidelberg, 2012.
10. D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, 38(3):243–250, May 1994.
11. J.-S. Coron, A. Roy, and S. Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 170–187. Springer Berlin Heidelberg, 2014.
12. J.-S. Coron, A. Roy, and S. Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. *Journal of Cryptographic Engineering*, 5(2):73–83, 2015.
13. J. Daemen and V. Rijmen. Aes proposal: Rijndael, 1998.
14. C. De Canniere, O. Dunkelman, and M. Knežević. KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, Lecture Notes in Computer Science, pages 272–288. Springer Berlin Heidelberg, 2009.
15. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings Information Security*, 152(1):13–20, 2005.
16. J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. The LED block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2011*, Lecture Notes in Computer Science, pages 326–341. Springer Berlin Heidelberg, 2011.
17. P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D.Hämäläinen. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In

Euromicro Conference on Digital System Design, pages 577–583. IEEE Computer Society, 2006.

18. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer Berlin Heidelberg, 2003.
19. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Berlin Heidelberg, 1999.
20. G. Leander, C. Paar, A. Poschmann, and K. Schramm. New lightweight DES variants. In *Fast Software Encryption 2007*, Lecture Notes in Computer Science, pages 196–210. Springer Berlin Heidelberg, 2007.
21. C. H. Lim and T. Korkishko. mCrypton–A lightweight block cipher for security of low-cost RFID tags and Sensors. In *Information Security Applications 2006*, Lecture Notes in Computer Science, pages 243–258. Springer Berlin Heidelberg, 2006.
22. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer Berlin Heidelberg, 2011.
23. NanGate. The NanGate 45nm Open Cell Library. http://www.nangate.com.
24. National Bureau of Standards. *Data Encryption Standard*. U. S. Department of Commerce, Washington, DC, USA, Jan. 1977.
25. A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling. Side-channel resistant crypto for less than 2,300 GE. *Journal of Cryptology*, 24(2):322–345, 2011.
26. E. Prouff and M. Rivain. Masking against side-channel attacks: A formal security proof. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.
27. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. *The Twofish Encryption Algorithm: A 128-bit Block Cipher*. John Wiley & Sons, Inc., New York, NY, USA, 1999.
28. T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-bit block-cipher CLEFIA. In *Fast Software Encryption 2007*, Lecture Notes in Computer Science, pages 181–195. Springer Berlin Heidelberg, 2007.
29. T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. Twine: A lightweight, versatile block cipher. In *In ECRYPT Workshop on Lightweight Cryptography 2011*, pages 146–169, 2011.
30. W. Wu and L. Zhang. Lblock: A lightweight block cipher. In J. Lopez and G. Tsudik, editors, *Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 327–344. Springer Berlin Heidelberg, 2011.