Strong 8-bit Sboxes with Efficient Masking in Hardware

Erik Boss[1], Vincent Grosso[1], Tim Güneysu[2], Gregor Leander[1],
Amir Moradi[1], and Tobias Schneider[1]

[1] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
{erik.boss, vincent.grosso, gregor.leander, amir.moradi, tobias.schneider-a7a}@rub.de
[2] University of Bremen and DFKI, Germany
tim.gueneysu@uni-bremen.de

Abstract. Block ciphers are arguably the most important cryptographic primitive in practice. While their security against mathematical attacks is rather well understood, physical threats such as side-channel analysis (SCA) still pose a major challenge for their security. An effective countermeasure to thwart SCA is using a cipher representation that applies the threshold implementation (TI) concept. However, there are hardly any results available on how this concept can be adopted for block ciphers with large (i.e., 8-bit) Sboxes. In this work we provide a systematic analysis on and search for 8-bit Sbox constructions that can intrinsically feature the TI concept, while still providing high resistance against cryptanalysis. Our study includes investigations on Sboxes constructed from smaller ones using Feistel, SPN, or MISTY network structures. As a result, we present a set of new Sboxes that not only provide strong cryptographic criteria, but are also optimized for TI. We believe that our results will found an inspiring basis for further research on high-security block ciphers that intrinsically feature protection against physical attacks.

1 Introduction

Block ciphers are among the most important cryptographic primitives. Although they usually follow ad-hoc design principles, their security with respect to known attacks is generally well-understood. However, this is not the case for the security of their implementations. The security of an implementation is often challenged by physical threats such as side-channel analysis or fault-injection attacks. In many cases, those attacks render the mathematical security meaningless. Hence, it is essential that a cipher implementation incorporates appropriate countermeasures against physical attacks. Usually, those countermeasures are developed retroactively for a given, fully specified block cipher. A more promising approach is including the possibility of adding efficient countermeasures into the design from the very start.

For software implementations, this has been done. Indeed, a few ciphers have been proposed that aim to address the issue of protection against physical attacks by facilitating a masked Sbox by design. The first example is certainly

NOEKEON [18], other examples include Zorro [20], Picarro [33] and the LS-design family of block ciphers [21].

For hardware implementations, the situation is significantly different. Here, simple masking is less effective due to several side-effects, most notably glitches (see [27]). As an alternative to simple masking, a preferred hardware countermeasure against side-channel attacks is the so-called threshold implementation (TI) [32], as used for the cipher FIDES [6]. TI is a masking variant that splits any secret data into several shares, using a simple secret-sharing scheme. Those shares are then grouped in non-complete subsets to be separately processed by individual subfunctions. All subfunctions jointly correspond to the target function (i.e., the block cipher). Since none of the subfunctions depends on all shares of the secret data at any time, it is intuitive to see that it is impossible to reconstruct the secret by first-order side-channel observations. We provide a more detailed description of the functionality of threshold implementations in Section 2.

Unfortunately, it is not trivial to apply the TI concept to a given block cipher. The success of this process strongly depends on the complexity of the cipher's round function and its internal components. While the linear aspects of any cipher are typically easy to convert to TI, this is not generally true for the non-linear Sbox. For 4-bit Sboxes, it is possible to identify a corresponding TI representation by exhaustive search [10]. However, for larger Sboxes, in particular 8-bit Sboxes, the situation is very different. In this case, the search space is far too large to allow an exhaustive search. In fact, 8-bit Sboxes are far from being fully understood, from both a cryptographic and an implementation perspective.

With respect to cryptographic strength against differential and linear attacks, the AES Sbox (and its variants) can be seen as holding the current world record. We do not know of any Sbox with better properties, but those might well exist. Unfortunately, despite considerable effort, no TI representation is known for the AES Sbox that does not require any additional external randomness [7, 9, 31].

Our Contribution. In this paper we approach this problem of identifying cryptographically strong 8-bit Sboxes that provide a straightforward TI representation. More precisely, our goal is to give examples of Sboxes that come close to the cryptanalytic resistance of the AES Sbox. Also, the straight application of the TI concept to an Sbox should still lead to minimal resource and area costs. This enables an efficient and low-cost implementation in hardware as well as bit-sliced software.

In our work we systematically investigate 8-bit Sboxes that are constructed based on what can be seen as a mini-cipher. Concretely, we construct Sboxes based on either a Feistel-network (operating with two 4-bit branches and a 4-bit Sbox as the round function), a substitution permutation network or the MISTY network. This general approach has already been used and studied extensively. Examples of Sboxes constructed like this are used for example in the ciphers Crypton [25, 26], ICEBERG [40], Fantomas [21], Robin [21] and Khazad [3]. A more theoretical study was most recently presented by Canteaut *et al.* in [16].

Our idea extends the previous work by combining those constructions aiming at achieving strong cryptographic criteria with small Sboxes that are easy to share and intrinsically support the TI concept. As a result of our investigation, we present a set of different 8-bit Sboxes. These Sboxes are either (a) superior to the known constructions from a cryptographic perspective but can still be implemented with moderate resource requirements or (b) outperform all known constructions in terms of efficiency in the application of the TI concept to the Sbox, while still maintaining a comparable level of cryptographic strength with respect to other known Sboxes. All our findings are detailed in Table 1.

Outline. This work is structured as follows. Preliminaries on well-known strategies to construct Sboxes as well as the TI concept are given in Section 2. We discuss the applicability of TI on known 8-bit Sboxes in Section 3. The details and results of the search process are given in Sections 4 and 5, respectively. We conclude with Section 6.

2 Preliminaries

2.1 Cryptanalytic properties for Sboxes

In this subsection we recall the tools used for evaluating the strength of Sboxes with respect to linear, differential and algebraic properties. For this purpose, we consider an n-bit Sbox S as a vector of Boolean functions: $S = (f_0, \ldots, f_{n-1})$, $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$. We denote the cardinality of a set A by $\#A$ and the dot product between two elements $a, b \in \mathbb{F}_2^n$ by: $\langle a, b \rangle = \sum_{i=0}^{n-1} a_i b_i$.

Non-linearity. To be secure against linear cryptanalysis [28] a cipher must not be well-approximated by linear or affine functions. As the Sbox is generally the only non-linear component in an SP-network, it has to be carefully chosen to ensure a design is secure against linear attacks. For a given Sbox, the main criterium here is the Hamming distance of any component function, i.e. a linear combination of the f_i, to the set of all affine functions. The greater this distance, the stronger the Sbox with respect to linear cryptanalysis. The Walsh transform $W_S(a, b)$, defined as

$$W_S(a, b) := \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle a, x \rangle + \langle b, S(x) \rangle},$$

can be used to evaluate the correlation of a linear approximation $(a, b) \neq (0, 0)$. More precisely,

$$\mathrm{P}(\langle b, S(x) \rangle = \langle a, x \rangle) = \frac{1}{2} + \frac{W_S(a, b)}{2^{n+1}}.$$

The larger the absolute value of $W_S(a, b)$, the better the approximation by the linear function $\langle a, x \rangle$ (or the affine function $\langle a, x \rangle + 1$, in case $W_S(a, b) < 0$).

This motivates the following well known definition.

Definition 1 (Linearity). *Given a vectorial Boolean function S, its* linearity *is defined as*

$$\mathrm{Lin}(S) = \max_{a,b\neq 0} |W_S(a,b)|.$$

The smaller $\mathrm{Lin}(S)$, the stronger the Sbox is against linear cryptanalysis.

It is known that for any function S from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ it holds that $\mathrm{Lin}(S) \geq 2^{\frac{n+1}{2}}$ [17]. Functions that reach this bound are called Almost Bent (AB) functions. However, in the case $n > 4$ and n even, we do not know the minimal value of the linearity that can be reached. In particular, for $n = 8$ the best known non-linearity is achieved by the AES Sbox with $\mathrm{Lin}(S) = 32$.

Differential Uniformity. A cipher must also be resistant against differential cryptanalysis [5]. To evaluate the differential property of an Sbox, we consider the set of all non-zero differentials and their probabilities (up to a factor 2^{-n}). That is, given $a, b \in \mathbb{F}_2^n$ we consider

$$\delta_S(a,b) := \#\{x \in \mathbb{F}_2^n \| \; S(x+a) = S(x) + b\},$$

which corresponds to 2^n times the probability of an input difference a propagating to an output difference b through the function S. This motivates the following well known definition.

Definition 2 (Differential Uniformity). *Given a vectorial Boolean function S, its* differential uniformity *is defined as*

$$\mathrm{Diff}(S) = \max_{a\neq 0,b} |\delta_S(a,b)|.$$

The smaller $\mathrm{Diff}(S)$, the stronger the Sbox regarding differential cryptanalysis.

It is known that for Sboxes S that have the same number of input and output bits it holds that $\mathrm{Diff}(S) \geq 2$. Functions that reach that bound are called Almost Perfect Nonlinear (APN). While APN functions are known for any number n of input bits, APN *permutations* are known only in the case of n odd and $n = 6$.

In particular, for $n = 8$ the best known case is $\mathrm{Diff}(S) = 4$, e.g., AES Sbox.

Algebraic degree. The algebraic degree is generally considered as a good indicator of security against structural attacks, such as integral, higher-order differential or, most recently, attacks based on the division property.

Recall that any Boolean function f can be uniquely represented using its Algebraic Normal Form (ANF):

$$f(x) = \sum_{u \in \mathbb{F}_2^n} a_u x^u,$$

where $x^u = \prod_{i=0}^{n-1} x_i^{u_i}$, with the convention $0^0 = 1$. Now, the algebraic degree can be defined as follows.

Definition 3 (Algebraic degree). *The algebraic degree of f is defined as:*

$$\deg(f) = \max_{u \in \mathbb{F}_2^n} \left\{ \sum_i u_i, a_u \neq 0 \right\}.$$

This definition can be extended to vectorial Boolean functions (Sboxes) as follows

$$\deg(S) = \max_{0 \leq i \leq n} \deg(f_i).$$

For a permutation on $\mathbb{F}_2^n$ the maximum degree is $n - 1$. Lots of permutations over $\mathbb{F}_2^n$ achieve this maximal degree. Again the AES Sbox is optimal in this respect, i.e., the AES Sbox has the maximal degree of 7 for 8-bit permutations.

Affine equivalence. An important tool in our search for good Sboxes is the notion of affine equivalence. We say that two functions f and g are affine equivalent if there exists two affine permutations A_1 and A_2 such that $f = A_1 \circ g \circ A_2$. The importance of this definition is given by the well-known fact that both the linearity and the differential uniformity are invariant under affine equivalence. That is, two functions that are affine equivalent have the same linear and differential criteria.

2.2 Construction of 8-Bit Sboxes

Apart from the AES Sbox, which is basically the inversion in the finite field $\mathbb{F}_{2^8}$, hardly any primary construction for useful, cryptographically strong, 8-bit Sboxes is known.

However, several secondary constructions have been applied successfully. Here, the idea is to build larger Sboxes from smaller Sboxes. For block ciphers this principle was first introduced in MISTY [29].

Later, this approach was modified and extended. In particular, it was used by several lightweight ciphers to construct Sboxes with different optimization criteria, e.g., smaller memory requirements, more efficient implementation, involution, and easier software-level masking.

There are basically three known constructions, all of which can be seen as mini-block ciphers: Feistel networks, the MISTY construction and SP-networks. Figure 1 shows how these constructions build larger Sboxes from smaller Sboxes. Note that the MISTY construction is a special case of the SPN. Indeed, the MISTY construction is equivalent to SPN when $F_1 = Id$ and the matrix $\mathcal{A} = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$.

For a small number of rounds, we can systematically analyze the cryptographic properties of those constructions (see [16] for the most recent results). However, for a larger number of rounds, a theoretical understanding becomes increasingly more difficult in most cases.

Table 1 shows the different characteristics of 8-bit Sboxes known in the literature that are built from smaller Sboxes. We excluded the PICARO Sbox [33]

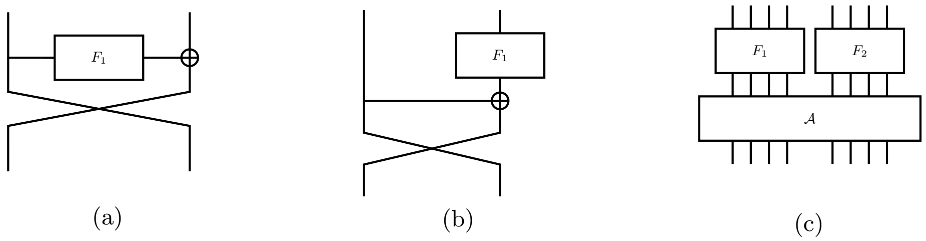


Fig. 1. (a): Feistel (b) MISTY (c) SPN

from the list, since it is not a bijection. Furthermore, Zorro is also excluded since the exact specifications of its structure are not publicly known. We refer often to this table as it summarizes all our findings and achievements.

2.3 Threshold Implementations

The first attempts to realize Boolean masking in hardware were unsuccessful, mainly due to glitches [27, 30]. Combinatorial circuits which receive both the mask and the masked data, i.e., secret sharing with 2 shares, most likely exhibit first-order leakage. Threshold Implementation (TI) has been introduced to deal with this issue and realize masking in glitchy circuits [32].

The TI concept has been extended to higher orders [8], but our target, in this work, is resistance against first-order attacks. Hence, we give the TI specifications only with respect to first-order resistance. Let us assume a k-bit intermediate value x of a cipher as one of its Sbox inputs (at any arbitrary round) and represent it as $\boldsymbol{x} = \langle x_1, \ldots, x_k \rangle$. For $n{-}1$ order Boolean masking, $\boldsymbol{x}$ is represented by $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n)$, where $\boldsymbol{x} = \bigoplus\limits_{i=1}^{n} \boldsymbol{x}^i$ and each $\boldsymbol{x}^i$ similarly denotes a k-bit vector $\langle x_1^i, \ldots, x_k^i \rangle$.

Applying linear functions over Boolean-masked data is trivial, since $\mathsf{L}(\boldsymbol{x}) = \bigoplus\limits_{i=1}^{n} \mathsf{L}(\boldsymbol{x}^i)$. However, realization of the masked non-linear functions (Sbox) is generally non-trivial and is thus the main challenge for TI. As per the TI concepts, at least $n = t + 1$ shares should be used to securely mask an Sbox with algebraic degree t. Moreover, TI defines three additional properties:

Correctness. The masked Sbox should provide the output in a shared form $(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^m)$ with $\bigoplus\limits_{i=1}^{m} \boldsymbol{y}^i = \boldsymbol{y} = \mathsf{S}(\boldsymbol{x})$ and $m \geq n$.

Non-completeness. Each output share $\boldsymbol{y}^{j \in \{1,\ldots,m\}}$ is provided by a component function $\mathsf{f}^j(.)$ over a subset of the input shares. Each component function $\mathsf{f}^{j \in \{1,\ldots,m\}}(.)$ must be independent of at least one input share.

Uniformity. The security of most masking schemes relies on the uniform distribution of the masks. Since in this work we consider only the cases with $n = m$ and bijective Sboxes, we can define the uniformity as follows. The masked Sbox with $n \times k$ input bits and $n \times k$ output bits should form a bijection. Otherwise, the output of the masked Sbox (which is not uniform) will appear at the input of the next masked non-linear functions (e.g., the Sbox at the next cipher round), and lead to first-order leakage.

Indeed, the challenge is the realization of the masked Sboxes with high algebraic degree. If $t > 2$, we can apply the same trick used in [32] and [34], i.e., by decomposing the Sbox into quadratic bijections. In other words, if we can write $S : G \circ F$, where both G and F are bijections with $t = 2$, we are able to implement the first-order TI of F and G with the minimum number of shares $n = 3$. Such a construction needs registers between the masked F and G to isolate the corresponding glitches.

After the decomposition, fulfilling all the TI requirements except *uniformity* is straightforward. As a solution, the authors of [10] proposed to find affine functions A_1 and A_2 in such a way that $F : A_2 \circ \mathcal{Q} \circ A_1$. If we are able to represent a uniform sharing of the quadratic function $\mathcal{Q}$, applying A_1 on all input shares, and A_2 on all output shares gives us a uniform sharing of F.

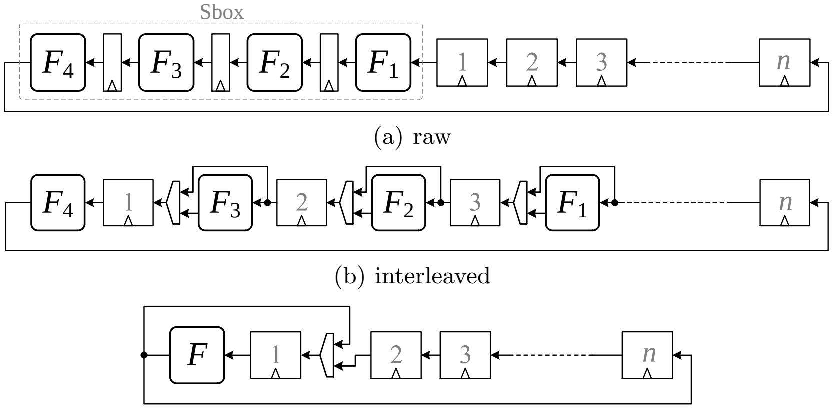
TI of 4-bit Permutations. In [11] the authors analyze 4-bit permutations and identify 302 equivalence classes. In the following, we use the same notation as in [11] to refer to these classes. Out of these 302, six classes are quadratic. These six quadratic functions, whose uniform TI can be achieved by *direct sharing* or with simple *correction terms* (see [11]) are listed in Table 2. We included their minimum area requirements as the basis of our investigations in the next sections. In contrast to the others, $\mathcal{Q}_{300}$ also needs to be decomposed for uniform sharing.

2.4 Design Architectures

Due to the high area overhead of threshold implementations (particularly the size of the shared Sbox), serialized architectures are favored, e.g. in [9, 31, 34, 38]. Our main target in this work is a serialized architecture in which one instance of the Sbox is implemented. Furthermore, we focus on byte-wise serial designs due to our underlying 8-bit Sbox target. In such a scenario, the state register forms a shift register, that at each clock cycle shifts the state bytes through the Sbox and makes use of the last Sbox output as feedback. Figure 2 depicts three different architectures which we can consider. Note that extra logic is not shown in this figure, e.g. the multiplexers to enable other operations like ShiftRows.

A shared Sbox with 3 shares should contain registers, e.g., PRESENT [34] and AES [9, 31]. As an example, if the shared Sbox contains 4 stages (see Figure 2(a)) and forms a pipeline, all of the Sbox computations can be done in $n + 3$ clock cycles, with n as the number of state bytes. We refer to this architecture as *raw* in later sections. Note that realizing a pipeline is desirable. Otherwise, the Sbox computations would take $3n + 1$ clock cycles.

As an alternative, we can use the state registers as intermediate registers of the shared Sbox. Figure 2(b) shows the corresponding architecture, where



(a) raw

(b) interleaved

(c) iterative

Fig. 2. Different serialized design architectures

more multiplexers should be integrated to enable the correct operation (as an example in Skinny [4]). In this case, all n shared Sboxes can be computed in n clock cycles. It is noteworthy that such an optimization is not always fully possible if intermediate registers of the shared Sbox are larger than the state registers (e.g., in case of AES [9, 31]).

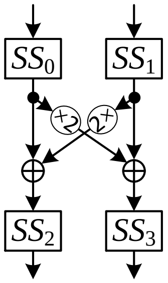
If the Sbox has been constructed by k times iterating a function F, it is possible to significantly reduce the area cost. Figure 2(c) shows an example. Therefore, similar to a raw architecture without pipeline, $(k-1)\,n+1$ clock cycles are required for n Sboxes. This is not efficient in terms of latency, but is favorable for low-throughput applications, where very low area is available and in particular when SCA protection is desired. We refer to this architecture as *iterative*.

3 Threshold Implementation of Known 8-Bit Sboxes

Amongst 8-bit Sboxes, the AES TI Sbox has been widely investigated while nothing about the TI of other Sboxes can be found in public literature. The first construction of the AES TI Sbox was reported in [31]. The authors made use of the tower-field approach of Canright [15] and represented the full circuit by quadratic operations. By applying second-order Boolean masking, i.e., three shares as minimum following the TI concept, all operations are independently realized by TI. On the other hand, the interconnection between (and concatenation of) uniform TI functions may violate the uniformity. Therefore, the authors integrated several fresh random masks – known as remasking or applying virtual shares [11] – to maintain the uniformity, in total 48 bits for each full Sbox. Since the AES TI Sbox has been considered for a serialized architecture, the authors formed a 4-stage pipeline design, which also increased the area by 138 registers.

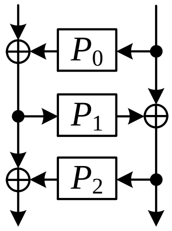
Later in [9] three more efficient variants of the AES TI Sbox were introduced. The authors applied several tricks, e.g., increasing the number of shares to 4 and

5 and reduce them back to 3 in order to relax the fresh randomness requirements. Details of all different designs are listed in Table 1. In short, the most efficient design (called *nimble*) forms a 3-stage pipeline, where 92 extra registers and 32 fresh random bits are required.



CLEFIA makes use of two 8-bit Sboxes S_0 and S_1. The first one is formed by utilizing four different 4-bit bijections and multiplication by 2 in GF(2^4) defined by polynomial $X^4 + X + 1$. The entire SS_0 : `E6CA872FB14059D3`[1], SS_1 : `640D2BA39CEF8751`, SS_2 : `B85EA64CF72310D9`, and SS_3 : `A26D345E0789BFC1` are cubic and – based on the classification given in [11] – belong to classes $\mathcal{C}_{210}$, $\mathcal{C}_{163}$, $\mathcal{C}_{160}$, and $\mathcal{C}_{160}$ respectively. Unfortunately, all these classes are of non-alternating group and cannot be shared with 3 shares, i.e., no solution exists either by decomposition or remasking[2]. We should use at least 4 shares (which is out of our focus), and its uniform sharing with 4 shares also needs to be done in at least 3 stages. Therefore, a 4-share version of TI S_0 can be realized in 6 stages.

The second one is constructed following the AES Sbox, i.e., inversion in GF(2^8), but with a different primitive polynomial and affine transformations. Based on the observations in [2, 36], inversion in one field can be transformed to another field by linear isomorphisms. Therefore, S_1 and the AES Sbox are affine equivalent and all difficulties to realize the AES TI Sbox hold true for S_1.



Crypton V0.5 utilizes two 8-bit Sboxes, S_0 and S_1, in a 3-round Feistel, as shown here. By swapping P_0 and P_2 the Sbox S_0 is converted to its inverse S_1. P_1 : `AF4752E693C8D1B0` belongs to the cubic class $\mathcal{C}_{295}$. Similar to the sub functions of CLEFIA, it belongs to the non-alternating group and cannot be shared with 3 shares. In short, at least 4 shares in 3 stages should be used. Further, P_0 : `F968994C626A135F` and P_2 : `04842F8D11F72BEF` are quadratic, non-bijective functions, but that does not necessarily mean that their uniform sharing with 4 shares does not exist. We have examined this issue by applying *direct sharing* [11], and we could not find their uniform sharing with either 3 or 4 shares. In this case, remasking is a potential solution. However, due to the underlying Feistel structure of S_0 and S_1, the non-uniformity of the shared P_0 and P_2 does not affect the uniformity of the resulting Sbox as long as the sharing of the *Sbox input* is uniform. More precisely, P_0 output is XORed with the left half of the Sbox input. If the input is uniformly shared, the input of P_1 is uniform regardless of the uniformity of the P_0 output. See [8] and [11], where it is shown that $a \cdot b$ (AND gate) cannot be uniformly shared with 3 shares, but $a \cdot b + c$ (AND+XOR) can be uniform if a, b, and c are uniformly shared. Therefore, a 4-share version of TI S_0 (resp. S_1) can be realized in 5 stages.

[1] In the following we denote functions by a hexadecimal-string in which the first letter denotes the first element of the look-up table implementing the function.

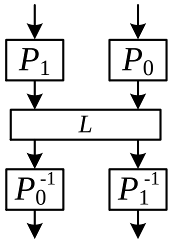
[2] Alternatively, one can apply the technique presented in [24].

Table 1. Criteria for the 8-bit Sboxes

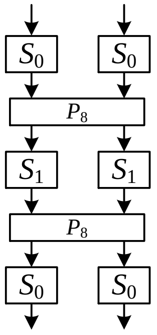
	Diff.	Lin.	Deg.	Iter.	AND	**Unprotected**			**Threshold Implementation**[a]					Type
						Area[GE]		Delay	Area[GE]		Delay	Stage	Mask	
				#[b]	#[c]	itera.[d]	raw[e]	ns	itera.[d]	raw[f]	ns	#[g]	#[h]	
AES [19]	**4**	**32**	**7**		32[13]		236	5.69		4244[31]		5	48	Inversion
										3708[9]		3	44	
										3653[9]		3	44	
										2835[9]		3	32	
CLEFIA (S_0) [39]	10	56	**7**							4 shares		6	0	SPN
CLEFIA (S_1) [39]	**4**	**32**	**7**							like AES		3	32	Inversion
Crypton V0.5 [25]	**8**	64	5				68	1.76		4 shares		5	0	Feistel
Crypton V1 [26]	10	64	6				111	2.40		4 shares		6	0	SPN
ICEBERG [40]	**8**	64	**7**				151	2.39		2115	1.67	9	0	SPN
Fantomas [21]	16	64	5		**11**		130	2.43		766	1.72	4	0	MISTY
Khazad [3]	**8**	64	**7**				154	2.48		2062	1.87	9	0	SPN
Robin [21]	16	64	6	3	**12**	28	79	2.37	319	1180	1.73	6	0	Feistel
Scream v3 [22]	**8**	64	6		**12**		87	2.38		2204	2.00	6	0	Feistel
Whirlpool [37]	**8**	56	**7**				146	2.37		2203	2.08	9	0	SPN
$\mathbf{SB}_1$	16	64	6	8	16	**8**	**57**	**1.38**	**51**	1189	**1.09**	8	0	SPN (BitP)
$\mathbf{SB}_2$	16	64	4	**2**	**12**	46	99	1.99	253	**631**	1.70	**2**	0	SPN (Mat)
$\mathbf{SB}_3$	**8**	60	**7**	4	24	48	198	3.98	273	1498	2.10	4	0	SPN (Mat)
$\mathbf{SB}_4$	**8**	**56**	**7**	5	30	29	140	4.09	202	1507	2.10	5	0	Feistel
$\mathbf{SB}_5$	10	60	**7**	9	27	12	95	3.19	78	1583	1.10	9	0	SPN (BitP)
$\mathbf{SB}_6$	10	60	**7**	4	20	49	174	4.78	226	1247	1.95	4	0	SPN (Mat)

[a] with 3 shares
[b] number of iterations of a unique function
[c] number of AND gates, important for masked bit-sliced software implementations
[d] excluding the required extra logic, e.g, multiplexers and registers
[e] fully combinatorial
[f] including pipeline registers
[g] number of stages in the pipeline
[h] number of fresh mask bits required for each full Sbox

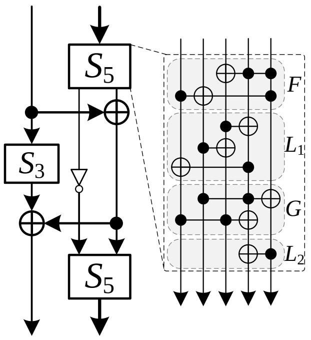
Crypton V1 Sboxes are made of two 4-bit bijections P_0 : `FEA1B58D9327064C`, P_1 : `BAD78E05F634192C` and their inverse in addition to a linear layer in between. P_0 and its inverse P_0^{-1} belong to the cubic class $\mathcal{C}_{278}$, which can be uniformly shared with 3 and 4 shares but in 3 stages. Both P_1 and its inverse P_1^{-1} are affine equivalent to the non-alternating cubic class $\mathcal{C}_{295}$, that – as given above – must be shared at least with 4 shares. Therefore, in order to share each Crypton V1 Sbox, 4 shares in a construction with 6 stages should be used.



ICEBERG is formed by two 4-bit bijections S_0 : `D7329AC1F45E60B8` and S_1 : `4AFC0D9BE6173582` in a 3-round SPN structure, where permutation P_8 is a bit permutation. Both S_0 and S_1 are affine equivalent to the cubic class $\mathcal{C}_{270}$, which needs at least 3 stages to be uniformly shared with 3 shares. Therefore, a uniform sharing of the ICEBERG Sbox with 3 shares can be realized in 9 stages without any fresh randomness. Amongst the smallest decompositions, we suggest $A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ for S_0 with

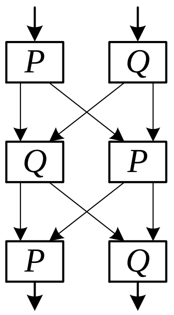


A_1 : `B038F47CD65E921A`, A_2 : `C6824E0AD7935F1B`, A_3 : `3DB50E8679F14AC2`, A_4 : `AC24E860BD35F971`, and for S_1 with A_1 : `63EB50D827AF149C`, A_2 : `D159F37BC048E26A`, A_3 : `2AE608C43BF719D5`, A_4 : `C5814D09E7A36F2B`, and $\mathcal{Q}_{294}$: `0123456789BAEFDC`.



Fantomas utilizes one 3-bit bijection S_3 : `03615427` and one 5-bit bijection S_5 : `00, 03, 12, 07, 14, 17, 04, 11, 0C, 0F, 1F, 0B, 19, 1A, 08, 1C, 10, 1D, 02, 1B, 06, 0A, 16, 0E, 1E, 13, 0D, 15, 09, 05, 18, 01` in a 3-round MISTY construction. S_3 is affine equivalent to the quadratic class $\mathcal{Q}_3^3$, which can be uniformly shared with 3 shares in at least 2 stages. As a decomposition, we considered $S_3 : \; A_3 \circ \mathcal{Q}_1 \circ A_2 \circ \mathcal{Q}_2 \circ A_1$ with A_1 : `07342516`, A_2 : `02461357`, A_3 : `01235476`, $\mathcal{Q}_1$: `01234576`, and $\mathcal{Q}_2$: `01234675`.

The construction of S_5, as shown here, consists of 4 Toffoli gates and 4 XORs. The quadratic F and G, as well as linear parts L_1 and L_2 are correspondingly marked. Hence, we can decompose $S_5 : \; L_2 \circ G \circ L_1 \circ F$. The uniform sharing of both F and G can be found by direct sharing. Therefore, the Fantomas Sbox can be uniformly shared with 3 shares in 4 stages, without any fresh mask. Figure 3(a) depicts the block diagram representation, and the area requirements are listed in Table 1. Each Sbox cannot be implemented iteratively, and each Sbox computation has a latency of 4 clock cycles. However, a pipeline design can send out Sbox results in consecutive clock cycles, but with a 4-clock-cycle latency.



Khazad utilizes the Anubis Sbox, which is also based on a 3-round SPN. Two 4-bit bijections P : `3FE054BCDA967821` and Q : `9E56A23CF04D7B18` in addition to a bit permutation layer form the 8-bit Sbox. Similar to ICEBERG, both P and Q belong to the cubic class $\mathcal{C}_{270}$. Therefore, the uniform sharing of the Khazad (resp. Anubis) Sbox can be realized in 9 stages without fresh masks. For the decomposition, we suggest $A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ for P with A_1 : `04C862AE15D973BF`, A_2 : `A2E680C4B3F791D5`, A_3 : `842EA60CB71D953F`, A_4 : `80D5C491A2F7E6B3`, and for Q with A_1 : `082A3B194C6E7F5D`, A_2 : `3FB71D952EA60C84`, A_3 : `19D53BF708C42AE6`, A_4 : `0B38291A4F7C6D5E`.

Robin is constructed based on the 3-round Feistel, similar to Crypton V0.5, but a single 4-bit bijection S_4 plays the role of all functions P_1, P_2, and P_3. Although the swap of the nibbles in the last Feistel round is omitted, the Robin Sbox is the only known 8-bit Sbox which can be implemented in an iterative fashion. S_4 : `086D5F7C4E2391BA` has been taken from [41], known as the Class 13 Sbox. S_4 is affine equivalent to the cubic class $\mathcal{C}_{223}$ and, as stated above, can be uniformly shared with 3 shares in 2 stages. As one of the smallest solutions we considered $A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with A_1 : AE268C04BF379D15, A_2 : C480A2E6D591B3F7, A_3 : 20A8B93164ECFD75. Therefore, with no extra fresh randomness we can realize uniform sharing of the Robin Sbox with 3 shares in 6 stages.

In order to implement this construction, we have four different options. A block diagram of the design is shown in Figure 3(b) (the registers filled by the gray color are essential for pipeline designs).

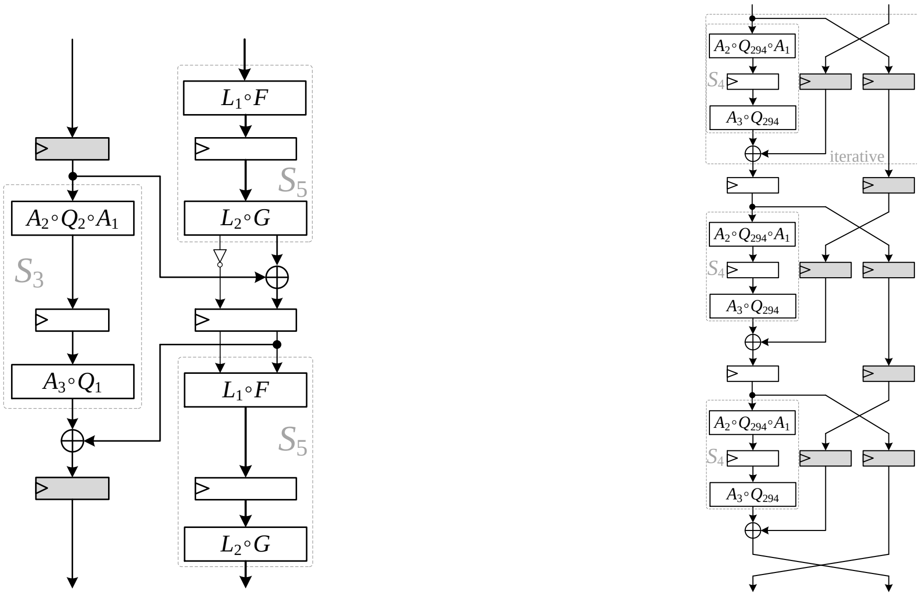
- Iterative, w/o pipeline, each Sbox in 6 clock cycles.
- Iterative, pipeline, each two Sboxes in 6 clock cycles.
- Raw, w/o pipeline, each Sbox in 6 clock cycles.
- Raw, pipeline, each 6 Sboxes in 6 clock cycles, each one with a latency of 6 clock cycles.

Note that extra control logic (such as multiplexers) is required for all iterative designs which is excluded from Figure 3(b) and Table 1 for the sake of clarity.

Scream V3 is similar to that of Crypton V0.5, i.e., 3-round Feistel. P_0, and P_2 are replaced by two *almost perfect nonlinear* (APN) functions $APN1$: 020B300A1E06A452 and $APN2$: 20B003A0E1604A25, and P_1 by S_1 : 02C75FD64E8931BA. Similar to Crypton V0.5, the two APN functions are not bijective. However, they are cubic rather than quadratic. The source of these two APNs is the construction given in [16]. We can decompose both of them into two quadratic functions as $APN1 : \ F \circ G$ and $APN2 : \ F \circ (\oplus 1) \circ G$, with F : 020B30A01E06A425 and G : 0123457689ABCDFE. By $(\oplus 1)$ we represent an identity followed by XOR with constant 1, i.e., flipping the least significant bit. Uniform sharing of G with 3 shares can be easily achieved by direct sharing. F, however, cannot be easily shared. F consists of three 2-input AND gates which directly give three output bits. To the best of our knowledge, F cannot be uniformly shared without applying remasking. However, as stated for Crypton V0.5, the non-uniformity of F (in general $APN1$ and $APN2$) does not play any role if S_1 is uniformly shared.

S_1 is affine equivalent to the cubic class $\mathcal{C}_{223}$ which can be uniformly shared in 2 stages with 3 shares. Therefore, the Scream V3 Sbox can be shared by 3 shares in 6 stages, without any fresh random masks. There are many options to decompose S_1; as one of the smallest solutions we suggest $S_1 : \ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with A_1 : 26AE159D37BF048C, A_2 : 4C086E2A5D197F3B, A_3 : 082A3B194C6E7F5D.

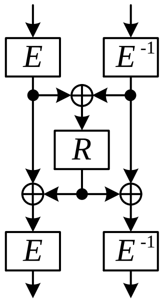
Whirlpool employs three different 4-bit bijections E, E^{-1} and R in a customized SPN. E : 1B9CD6F3E874A250 and its inverse are affine equivalent to the cubic class $\mathcal{C}_{278}$, which can be uniformly shared with 3 shares in at least 3 stages. R : 7CBDE49F638A2510 also belongs to the cubic class $\mathcal{C}_{270}$. As given for ICEBERG and Khazad, $\mathcal{C}_{270}$ needs 3 stages for a uniform sharing with 3 shares. Hence, the entire Whirlpool Sbox can be uniformly shared with 3 shares in 9 stages, without any extra randomness. The decomposition of R is similar to that of Khazad, i.e., $R \ : \ A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with



(a) Fantomas (b) Robin

Fig. 3. Threshold Implementation of Robin and Fantomas Sboxes, each signal represents 3 shares, the gray registers for pipeline variants

A_1 : `02138A9BCEDF4657`, A_2 : `0C48A6E21D59B7F3`,
A_3 : `C509E72BD418F63A`, A_4 : `0A1B4E5F28396C7D`.
However, the decomposition of E and E^{-1} are more costly. One of the cheapest solutions is $A_4 \circ \mathcal{Q}_{294} \circ A_3 \circ \mathcal{Q}_{293} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ for E with A_1 : `048CAE2673FBD951`, A_2 : `80C4B3F7A2E691D5`, A_3 : `0B834FC71A925ED6`, A_4 : `014589CD2367ABEF`, and for E^{-1} with A_1 : `A2F76E3B80D54C19`, A_2 : `A280E6C4B391F7D5`, A_3 : `95F31D7B84E20C6A`, A_4 : `2736AFBE05148D9C`, and $\mathcal{Q}_{293}$: `0123457689CDEFBA`.



Due to their required minimum 4 shares, except for CLEFIA, Crypton V0.5, and Crypton V1, we have implemented TI for all the aforementioned Sboxes, and have given their area requirements as well as the number of stages (clock cycles) in Table 1. For the synthesis, we used Synopsys Design Compiler with the UMC-L18G212T3 [42] ASIC standard cell library, i.e., UMC 0.18μm technology node. It is noteworthy that amongst all the Sboxes we covered, the Robin Sbox is the only one which can be iteratively implemented. We should also emphasize that Midori [1] and Skinny [4] (in their 128-bit versions) make use of 8-bit Sboxes. Midori 8-bit Sboxes are made by concatenating two 4-bit Sboxes and the Skinny one by four times iterating an 8-bit quadratic bijection. In both cases their differential and linear properties are 64 and 128 respectively, which are notably less compared to the strong 8-bit Sboxes listed in Table 1. Therefore, we did not consider them in our investigations.

Table 2. Performance figures of 4 × 4 quadratic bijections with respect to their TI cost

	Table	Area [GE]	# of stages
$\mathcal{Q}^4_4$	`0123456789ABDCFE`	27	1
$\mathcal{Q}^4_{12}$	`0123456789CDEFAB`	63	1
$\mathcal{Q}^4_{293}$	`0123457689CDEFBA`	84	1
$\mathcal{Q}^4_{294}$	`0123456789BAEFDC`	51	1
$\mathcal{Q}^4_{299}$	`012345678ACEB9FD`	114	1
$\mathcal{Q}^4_{300}$	`0123458967CDEFAB`	151	2 ($\mathcal{Q}_{12} \circ \mathcal{Q}_4$)

4 Finding TI-Compliant 8-Bit Sboxes

Our goal is to find strong 8-bit Sboxes which can be efficiently implemented as threshold implementations. To this end, we incorporate the idea of building an 8-bit Sbox from smaller Sboxes in our search. In particular we aim to construct a round function that can be easily shared and iterated to generate a cryptographically strong Sbox. Easily shareable in our context refers to functions for which an efficient uniform shared representation is known. Thus, if we find a function with these properties, the resulting sequence of round functions will be a good cryptographic Sbox which can be efficiently masked. As done previously, we concentrate on the three basic constructions mentioned above: Feistel, SPN, and MISTY. As the number of possible choices for SPN is too large for an exhaustive search, we focus on two special cases for the linear layer of the SP-network. First, instead of allowing general linear layers we focus on bit-permutations only. Those have the additional advantage of being basically for free, both in hardware and in a (bitsliced) software implementation. Second, we focus on linear layers which correspond to matrix multiplications over $\mathbb{F}_{16}$. Those cover the MISTY construction as a special case.

In all cases, the building blocks for our round function are 4-bit Sboxes. As described in Section 2, those Sboxes are well-analyzed and understood regarding both their threshold implementation [11] and their cryptographic properties. To minimize the number of required shares, we mainly consider functions with a maximum degree of two. Additional shares, otherwise, may increase the area or randomness requirements for the whole circuit. In [11], six main quadratic permutation classes are identified which are listed in Table 2. All existing quadratic 4-bit permutations are affine equivalent to one of those six. However, it should be noted that permutations of class $\mathcal{Q}^4_{300}$ cannot be easily shared with three shares without decomposition or additional randomness. Therefore, we mainly focus on the other classes from our search. Note that we include the identity function $\mathcal{A}^4_0$ in the case of the SPN construction. Since the identity function does not require any area, round functions based on a combination of identity and one quadratic 4-bit permutation can result in very lightweight designs.

One important difference to all previous constructions listed in Table 1 is that we do consider higher number of iterations for our constructions. This is motivated by two observations. First, it might allow to improve upon the cryptographic criteria and second it might be beneficial to actually use a simpler round function, in particular those that can be implemented in one stage, more often than a more complicated round function with a smaller number of itterations. As can be seen in Table 1 this approach of increasing the number of itterations is quite successful in many cases.

Next we describe in detail the search for good Sboxes for each of the three constructions we considered.

4.1 Feistel-Construction

As a first construction, we examine round functions using a Feistel-network similar to Figure 1(a). By the basic approach described below, we were able to exhaustively investigate all possible constructions based on any 4-bit to 4-bit function for any number of itterations between 1 and 5. This can be seen as an extension (in the case of $n = 4$ and for identical round functions) to the results given in [16] where up to 3 rounds have been studied.

However, such an exhaustive search is not possible in a naive way. As there are 2^{64} 4-bit functions and checking the cryptographic criteria of an n-bit Sbox requires roughly 2^{2n} basic operations, a naive approach would need more than 2^{80} operations.

Fortunately, this task can be accelerated by exploiting the distinct structure of Feistel-networks while still covering the entire search space.

We recall the definition of a Feistel round for the function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$:

$$\mathrm{Feistel}_F^1 : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \times \mathbb{F}_2^n, \qquad (L, R) \mapsto (R \oplus F(L), L).$$

We denote by $\mathrm{Feistel}_F^n$ the nth functional power of $\mathrm{Feistel}_F^1$, i.e.,

$$\mathrm{Feistel}_F^n = \mathrm{Feistel}_F^1 \circ \mathrm{Feistel}_F^1 \circ \cdots \circ \mathrm{Feistel}_F^1 \,.$$

To reduce the search space, we show below that if $G = A \circ F \circ A^{-1}$ for an invertible affine function A, then $\mathrm{Feistel}_F^n$ is affine equivalent to $\mathrm{Feistel}_G^n$.

Thus, we can reduce our search space from all 2^{64} functions, to roughly XY functions. Indeed, Brinkmann classified all 4 to 4 bit functions up to extended affine equivalence [14]. There are 4713 equivalence classes up to extended affine equivalence. Now, with the results given in the full version of the paper [12], it is enough to consider all functions of the form $A_1 \circ F + C$, where A_1 is an affine permutation and C is any linear mapping on 4 bits. As $\mathrm{Feistel}^n_{A_1 \circ F \circ A_2 + C'}$ is affine equivalent to the function $\mathrm{Feistel}^n_{A_2 \circ A_1 \circ F \circ A_2 \circ A_2^{-1} + C' \circ A_2^{-1}} = \mathrm{Feistel}^n_{A_2 \circ A_1 \circ F + C}$, this will exhaust all possibilities up to affine equivalence. Doing so, we reduce the search space to:

$$\#Sboxes = 4713 \cdot 2^4 \cdot |\mathrm{GL}(2,4)| \cdot 2^{16} \simeq 2^{46.50}. \tag{1}$$

As this is still a large search space, we emplyed GPUs to tackle this task.

4.2 SPN-Construction with Bit-Permutations as the Linear Layer

In addition to Feistel-networks, we examined round functions which are similar to Figure 1(c). However, $\mathcal{A}$ is replaced by an XOR with a constant followed by an 8-bit permutation. Depending on F_1 and F_2, this construction can lead to very lightweight round functions since constant addition and simple bit permutations are very efficient in hardware circuits. For F_1 and F_2 we consider the five quadratic permutations (listed in Table 2) as well as the identity function (denoted by $\mathcal{A}_0^4$). Obviously, we exclude the combination $F_1 = F_2 = \mathcal{A}_0^4$. There are 8! different 8-bit permutations and 256 possibilities for the constant addition. If we looked for all combinations of all affine equivalents of the chosen functions, we would have to test

$$\#Sboxes = 256 \cdot 8! \cdot 35 \cdot 322560^4 \cdot 10 \simeq 2^{105} \tag{2}$$

Sboxes. This is clearly not feasible. Therefore, we decide to restrict the number of possibilities for each of the two functions. In particular, we only consider the representative for each class as presented in [11] without affine equivalents. This reduces the search space to

$$\#Sboxes = 256 \cdot 8! \cdot 35 \cdot 10 \simeq 2^{32}, \tag{3}$$

which can be completely processed.

Similar to the Feistel-network, it is possible to further reduce the complexity of the search. To this end, we first define the round function for this type of Sbox as

$$\begin{aligned} \mathrm{BitPerm}^1_{F_1,F_2,C,P} : \mathbb{F}_2^n \times \mathbb{F}_2^n &\to \mathbb{F}_2^{2n} \\ (L,R) &\mapsto P\big((F_1(L)||F_2(R)) \oplus C\big), \end{aligned}$$

where || denotes the concatenation of the two parts. Furthermore, it can be trivially seen that for every combination of an 8-bit permutation P_1 and an 8-bit constant C_1 there exist a complementary combination of an 8-bit permutation P_2 and an 8-bit constant C_2 with

$$P_1\big((L||R) \oplus C_1\big) = P_2\big((R||L) \oplus C_2\big), \quad \forall\, R, L \in \mathbb{F}_2^n.$$

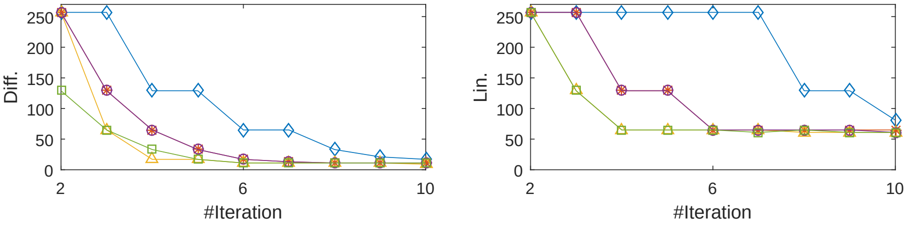
Thus, the search can be speeded up since $\mathrm{BitPerm}^1_{F_1,F_2,C_1,P_1}$ is the same as $\mathrm{BitPerm}^1_{F_2,F_1,C_2,P_2}$. Therefore, we only need to check

$$\#Sboxes = 256 \cdot 8! \cdot 20 \cdot 10 \simeq 2^{31} \tag{4}$$

Sboxes for this type of round function.

4.3 SPN-Construction with $\mathbb{F}_{16}$-linear Layers only

For the last type of construction, we consider another special case of the construction depicted in Figure 1(c). Here we restrict ourselves to the case where $\mathcal{A}$ corresponds to a multiplication with a 2×2 matrix with elements from $\mathbb{F}_{16}$. Additionally, a constant is again added to the outputs of F_1 and F_2. As noted before, a special case of this construction is the MISTY technique.



(a) Differential Uniformity

(b) Linearity

Fig. 4. The smallest achievable differential uniformity and linearity for each number of iterations for round functions with $\mathbb{F}_{16}$-linear layers and $F_1 = \mathcal{A}_0^4$ and $(\diamond)F_2 = \mathcal{Q}_4^4$, $(*)F_2 = \mathcal{Q}_{12}^4$, $(\triangle)F_2 = \mathcal{Q}_{293}^4$, $(\circ)F_2 = \mathcal{Q}_{294}^4$, $(\square)F_2 = \mathcal{Q}_{299}^4$.

For F_1 and F_2 we consider the five quadratic functions and the identity function. Just like for the bit permutation round function, it is not feasible to check all affine equivalents. Therefore, we limit our search to these functions. The field multiplication is performed with the commonly used polynomial $X^4 + X + 1$ [23]. Given that the matrix needs to be invertible and provide some form of mixture between the two halves, this leaves us with 61200 possibilities for the matrix multiplication. It is further possible to apply the same optimization as for permutation-based round functions. Therefore, we need to check

$$\#Sboxes = 256 \cdot 61200 \cdot 20 \cdot 10 \simeq 2^{31.5} \qquad (5)$$

Sboxes for this type of round function.

5 Results

We completed the search for the three aforementioned types of round functions with up to ten iterations.

The search for Feistel-networks for all 4713 classes takes around two weeks on a machine with four NVIDIA K80s for a specific set of parameters. In particular, the performance depends on the bounds defined by cryptographic properties (differential uniformity) as well as the iteration count of the network. Note that, with respect to cryptographic criteria, our search shows that for iterations ≤ 5 *no* 8*-bit balanced Feistel with identical round functions* can have a linearity below 56 *and* a differential uniformity below 8.

Furthermore, the search for SPNs with bit permutations (resp. with $\mathbb{F}_{16}$-linear layer) required around 48 hours (resp. 54 hours) on one Intel Xeon CPU with 12 cores. It was possible to detect some very basic relations between the security, number of iterations and area of the Sbox. Figure 4 shows the smallest differential uniformity and linearity values which can be achieved for a specific number of iterations using a round function based on the $\mathbb{F}_{16}$-linear layer with constant addition. As expected, the more iterations are applied, the higher resistance against linear and differential cryptanalysis could be achieved. The size of each of the considered quadratic permutations is given in Table 2. Bigger functions like $\mathcal{Q}^4_{293}$ and $\mathcal{Q}^4_{299}$ achieve good cryptographic properties with fewer iterations than smaller functions like $\mathcal{Q}^4_4$. For the other combinations of (F_1, F_2) and types of round functions the graphs behave similarly. Depending on the remaining layers of the cipher and the targeted use case, a designer needs to find a good balance between the parameters. In the following, we present a few selected Sboxes optimized for different types of applications.

In our evaluation, we only consider Sboxes with differential uniformity at most 16 and linearity of at most 64. These are the worst properties between the observed constructed 8-bit Sboxes in Table 1. From the cryptographic standpoint, our Sboxes should not be inferior to these functions. We identified the following strong Sboxes that cover the most important scenarios.

- $\mathbf{SB}_1$: This Sbox possesses a very small round function. In a serial design the round function is usually implemented only once to save area.
- $\mathbf{SB}_2$: This Sbox is selected to enable an efficient implementation in a round-based design. For this not only the size of the round function is important but also the number of iterations. Additional iterations require additional instantiations of the round function with a dedicated register stage. Furthermore, this Sbox requires the least number of iterations and can be implemented with a very low number of AND gates. Thus, it is also suited to masked software implementations.
- $\mathbf{SB}_3$: This Sbox has very good cryptographic properties and requires one less iteration than $\mathbf{SB}_4$.
- $\mathbf{SB}_4$: This Sbox has very good cryptographic properties.
- $\mathbf{SB}_5$: This Sbox is similar to $\mathbf{SB}_1$ which has a small round function. However, it trades area for better cryptographic properties.
- $\mathbf{SB}_6$: This Sbox is similar to $\mathbf{SB}_2$ that is optimized for raw implementations. However, it trades area for better cryptographic properties.

5.1 Selected Sboxes

In this section, we supply the necessary information to implement the selected Sboxes. For this, we first recall the basic structure of the round functions. Table 1 shows that our selected round functions consists of bit permutations and $\mathbb{F}_{16}$-linear layers. The structure of both types is similar to Figure 1(c). We denote the most (resp. least) significant four bits as L (resp. R). The round function $\mathit{Round} : \mathbb{F}_2^4 \times \mathbb{F}_2^4 \mapsto \mathbb{F}_2^8$ is then defined as

$$\mathrm{Round}(L, R) = P\big((F_1(L)||F_2(R)) \oplus C\big),$$

where C is an 8-bit constant and $P(.)$ denotes either an 8-bit permutation or an $\mathbb{F}_{16}$-linear layer. In Table 3, we describe a specific bit permutation with an eight-element vector where each element denotes the new bit position, e.g., no permutation is `01234567` whereas complete reversal is `76543210`. The $\mathbb{F}_{16}$-linear layer is realized as a multiplication with a 2×2 matrix with elements in $\mathbb{F}_{16}$.

Table 3. Specifics of the selected Sboxes.

	F_1	F_2	Const. (Hex)	Parameter	Type	Iterations
$\mathbf{SB}_1$	$\mathcal{A}_0^4$	$\mathcal{Q}_{294}^4$	`04`	`62750413`	Perm.	8
$\mathbf{SB}_2$	$\mathcal{Q}_{293}^4$	$\mathcal{Q}_{293}^4$	`EE`	$[2, 4, 4, 2]$	Matrix	2
$\mathbf{SB}_3$	$\mathcal{Q}_{293}^4$	$\mathcal{Q}_{299}^4$	`6C`	$[2, 2, 3, 11]$	Matrix	4
$\mathbf{SB}_5$	$\mathcal{Q}_4^4$	$\mathcal{Q}_{294}^4$	`85`	`20647135`	Perm.	9
$\mathbf{SB}_6$	$\mathcal{Q}_{293}^4$	$\mathcal{Q}_{294}^4$	`F8`	$[0, 5, 13, 15]$	Matrix	4

	F	G	A	Type	Iterations
$\mathbf{SB}_4$	`0001024704638EAD`	`028A9B1346CEDF57`	`6273627351405140`	Feistel	5

Let us denote the most (resp. least) significant four input bits to the matrix multiplication as L_M (resp. R_M). The multiplication is then defined as

$$\mathrm{MatMul}(L_M, R_M) = (E_1 \cdot L_M \oplus E_2 \cdot R_M || E_3 \cdot L_M \oplus E_4 \cdot R_M)\,,$$

where $E_1, E_2, E_3, E_4 \in \mathbb{F}_{16}$ are the elements of the chosen matrix. To describe the linear layers of our Sboxes we give the specific $[E_1, E_2, E_3, E_4]$ for each matrix in Table 3.

These parameters combined with the number of iterations enable the realizations of each Sbox. To increase efficiency of the TI the constant is added to only one of the shares. In some cases, the area of the design can be reduced by adding a particular constant to the two remaining shares. This is based on the fact that an additional NOT gate can turn e.g., an AND gate to a smaller NAND gate [35]. The following linear layer still needs to be applied to all shares. Table 3 contains this condensed description of the selected Sboxes. Further details for each of them can be found in the full version of the paper [12].

For $\mathbf{SB}_4$, since it uses a Feistel-network, we construct the Sbox using the round function $H(x) = G(F(x)) \oplus A(x)$, where F is taken from the 4713 equivalence classes; G and A represent the linear and affine parts respectively. H, F, G and A are all 4-bit to 4-bit functions. The full definition of the round is then simply $(L, R) \mapsto (R \oplus H(L), L)$.

5.2 Comparison

Table 1 gives an overview of our results and we summarize the most important observations in the following. The first observation is that our proposed designs do not require fresh mask bits to achieve uniformity. This is an improvement over all TI types of the AES Sbox and some other Sboxes from Table 1. They need up to 64 bits of randomness for one full Sbox. Given that modern ciphers usually include multiple rounds with many Sboxes, this can add up to a significant amount of randomness which needs to be generated.

Furthermore, all of our proposed Sboxes can be implemented iteratively. This comes with the advantage that even the more complex designs, e.g., $\mathbf{SB}_4$ and $\mathbf{SB}_5$, can be realized with very few gates depending on the design architecture. From all the other Sboxes in Table 1 this is only possible for Robin and its round function requires more area than any of our proposed Sboxes.

In particular, $\mathbf{SB}_1$ and $\mathbf{SB}_2$ require the least area in their respective target architectures (i.e., iterative and raw) out of all considered 8-bit Sboxes. The difference for the iterative architecture is especially large where $\mathbf{SB}_1$ needs roughly six times less area than the Robin Sbox.

$\mathbf{SB}_2$ requires the least number of stages. Additionally, it requires only 12 AND gates for the whole Sbox which is very close to the best number, i.e., 11 for Fantomas. This is an advantage for masked bit-sliced implementations making $\mathbf{SB}_2$ suitable for software and hardware designs. A more detailed discussion of this aspect is given in the full version of the paper [12].

As expected, we did not find any Sbox with better cryptographic properties than the AES Sbox. However, $\mathbf{SB}_3$ and $\mathbf{SB}_4$ can still provide better resistance against cryptanalysis attacks than most of the other considered Sboxes. This comes at the cost of an increased area for the raw implementations. Nevertheless, the required area is still smaller than any AES TI and their round function is still smaller than Robin for iterative designs.

As depicted in Figure 4, a trade-off between resources and cryptographic properties is possible. If $\mathbf{SB}_1$ and $\mathbf{SB}_2$ do not provide the desired level of security and $\mathbf{SB}_3$ and $\mathbf{SB}_4$ are too large, $\mathbf{SB}_5$ and $\mathbf{SB}_6$ might be the best solution. Their cryptographic properties are still better or equal than the competitors while the area is significantly smaller than $\mathbf{SB}_3$ and $\mathbf{SB}_4$. For the sake of completeness, we included the area requirement of the unprotected implementation as well as the latency of different designs in Table 1.

Decryption usually requires the inverse of the Sbox. Therefore, it is important that the Sbox inverse has comparably good properties to the original Sbox. For $\mathbf{SB}_4$ this is obvious since the Feistel-structure makes it straightforward to construct the inverse. Therefore, inverse $\mathbf{SB}_4$ has exactly the same properties as $\mathbf{SB}_4$. For the other cases, this is not trivial. Nevertheless, the inverse of each of our-considered quadratic functions is self-affine equivalent. For more information the interested reader is referred to the full version of the paper [12].

6 Conclusion and Future Work

In this work we identified a set of six 8-bit S-boxes with highly useful properties using a systematic search on a range of composite Sbox constructions. Our findings include 8-bit Sboxes that provide comparable or even higher resistance against linear and differential cryptanalysis with respect to other 8-bit Sbox but intrinsically support the TI concept without any external randomness. At the same time our selected Sboxes come with a range of useful implementation properties, such as a highly efficient serialization option, or a very low area requirement. Future work comprises extended criteria for the Sbox composition, including diffusion layers beyond permutations.

Acknowledgements

This work is partly supported by the DFG Research Training Group GRK 1817 Ubicrypt and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 643161 (ECRYPT-NET).

References

1. Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.
2. Elad Barkan and Eli Biham. In How Many Ways Can You Write Rijndael? In *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 160–175. Springer, 2002.
3. PSLM Barreto and Vincent Rijmen. The Khazad legacy-level block cipher. *Primitive submitted to NESSIE*, 97, 2000.
4. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The Skinny Family of Block Ciphers and its Low-Latency Variant Mantis. In *CRYPTO 2016*, Lecture Notes in Computer Science. Springer, 2016. to appear.
5. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1991.
6. Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In *CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2013.
7. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A More Efficient AES Threshold Implementation. In *AFRICACRYPT 2014*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2014.
8. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In *ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.
9. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-Offs for Threshold Implementations Illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.
10. Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold Implementations of All 3 × 3 and 4 × 4 S-Boxes. In *CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2012.
11. Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, Natalia Tokareva, and Valeriya Vitkup. Threshold implementations of small S-boxes. *Cryptography and Communications*, 7(1):3–33, 2015.
12. Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider. Strong 8-bit Sboxes with Efficient Masking in Hardware. Cryptology ePrint Archive, 2016. `http://eprint.iacr.org/`.
13. Joan Boyar and René Peralta. A New Combinational Logic Minimization Technique with Applications to Cryptology. In *Experimental Algorithms, SEA 2010*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2010.
14. Marcus Brinkmann. EA classification of all 4 bit functions. personal communication, 2008.
15. David Canright. A Very Compact S-Box for AES. In *CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
16. Anne Canteaut, Séebastien Duval, and Gaëtan Leurent. Construction of Lightweight S-Boxes using Feistel and MISTY structures. In *SAC 2015*, volume

9566 of *Lecture Notes in Computer Science*, pages 373–393. Springer, 2016. to appear.

17. Florent Chabaud and Serge Vaudenay. Links Between Differential and Linear Cryptanalysis. In *EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer, 1995.
18. Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie proposal: NOEKEON. In *First Open NESSIE Workshop*, pages 213–230, 2000.
19. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
20. Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block Ciphers That Are Easier to Mask: How Far Can We Go? In *CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2013.
21. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. In *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, 2015.
22. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, Anthony Journault, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM Side-Channel Resistant Authenticated Encryption with Masking – ver 3. submission to CAESAR competition of authenticated ciphers, `https://competitions.cr.yp.to/round2/screamv3.pdf`.
23. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In *CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.
24. Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann. Enabling 3-Share Threshold Implementations for all 4-Bit S-Boxes. In *ICISC 2013*, volume 8565 of *Lecture Notes in Computer Science*, pages 91–108. Springer, 2014.
25. Chae Hoon Lim. CRYPTON: A New 128-bit Block Cipher - Specification and Analysis. *NIST AES Proposal*, 1998.
26. Chae Hoon Lim. A Revised Version of Crypton - Crypton V1.0. In *FSE 1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 1999.
27. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
28. Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In *EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1994.
29. Mitsuru Matsui. New Block Encryption Algorithm MISTY. In *FSE 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 1997.
30. Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.
31. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.
32. Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.

33. Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance. In *ACNS 2012*, volume 7341 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2012.
34. 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. Cryptology*, 24(2):322–345, 2011.
35. Axel York Poschmann. *Lightweight cryptography: cryptographic engineering for a pervasive world*. PhD thesis, Ruhr University Bochum, 2009.
36. Håvard Raddum. More Dual Rijndaels. In *AES Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 142–147. Springer, 2004.
37. Vincent Rijmen and PSLM Barreto. The WHIRLPOOL hash function. *World-Wide Web document*, page 72, 2001.
38. Aria Shahverdi, Mostafa Taha, and Thomas Eisenbarth. Silent Simon: A threshold implementation under 100 slices. In *HOST 2015*, pages 1–6. IEEE, 2015.
39. Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.
40. François-Xavier Standaert, Gilles Piret, Gaël Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat. ICEBERG : An Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware. In *FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer, 2004.
41. Markus Ullrich, Christophe De Cannière, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding Optimal Bitsliced Implementations of 4×4-bit S-boxes. In *Symmetric Key Encryption Workshop*, page 20, 2011.
42. Virtual Silicon Inc. 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18μm, July 2004.