Abstract. Cryptographic primitives have been designed to be secure against mathematical attacks in a black-box model. Such primitives can be implemented in a way that they are also secure against physical attacks, in a grey-box model. One of the most popular techniques for this purpose is masking. The increased security always comes with a high price tag in terms of implementation cost. In this work, we look at how the traditional design principles of symmetric primitives can be at odds with the optimization of the implementations and how they can evolve to be more suitable for embedded systems. In particular, we take a comparative look at the round 2 candidates of the NIST lightweight competition and their implementation properties in the world of masking.

Keywords: DPA · Masking · NIST · lightweight · competition · side-channel · symmetric · S-box

1 Introduction

In the Internet of Things, cryptographic calculations are increasingly deployed in highly constrained environments. As a result, the National Institute of Standards and Technology (NIST) has initiated a standardization effort for lightweight cryptography. Apart from security against mathematical attacks (i.e., cryptanalysis), cryptographic implementations nowadays are required to provide security against physical attacks such as side-channel analysis (SCA) as well. Especially lightweight devices are likely to be among the targets that an adversary may have physical access to. As a result, NIST considers the ability to provide side-channel resistance at low cost an important evaluation criterion.

One of the most popular countermeasures against SCA at the algorithmic level is masking. The essence of masking is to split any (sensitive) variable into multiple shares such that all shares are required to calculate that variable. The most common type of masking is Boolean masking, where the variables are split using an XOR operation. As a consequence, masking linear or affine operations is relatively straightforward and comes with a linear overhead factor of $d + 1$, where $d$ is the order of SCA resistance. Nonlinear operations on the other hand are very complicated and expensive to mask. Their cost grows exponentially with the security order $d$. Our treatment will focus mostly on the nonlinear components of symmetric primitives, e.g., S-boxes, since the cost of implementations is dominated by their overhead.

However, the cost of a masked implementation does not follow straightforwardly from the cost of an unmasked implementation and designing a primitive with optimizing one cost in mind, does not automatically optimize the other. As an example, consider the optimization goal of low latency in hardware implementations. Many lightweight primitives of the past years involve a less complex round function (and especially S-box) than that of

1https://csrc.nist.gov/Projects/Lightweight-Cryptography
looking at the NIST Lightweight Candidates from a Masking Point-of-View

the Advanced Encryption Standard (AES), which is repeated for a higher number of rounds. The rationale is that this both reduces area requirements and that a round-based encryption can run at a higher frequency. The complexity of masking also grows considerably with the algebraic degree of a function. Therefore, it is common to split complex functions into lower-degree (especially quadratic) components. Since those masked quadratic blocks have to be separated by register elements, the maximum frequency of masked implementations does not depend on the total complexity of the round function. Furthermore, we will show in this work that the increased number of rounds is detrimental to the speed of masked implementations.

We first consider the topic of S-boxes in Section 2 and explain some of their properties and methods of classification. We consider both mathematical and implementation properties. Next, Section 3 considers the design of symmetric cryptosystems (especially S-boxes) from the embedded systems engineer’s point-of-view. We list optimization goals for hardware and software implementations and discuss the state-of-the-art, including proposals in the recent NIST lightweight competition.

2 S-box Properties and Affine Equivalence

We list here cryptographic properties, which indicate the S-box’s strength against mathematical attacks (i.e. cryptanalysis) and which were traditionally considered the principal evaluation criteria in the choice of S-boxes for primitives. Next, we describe S-box classifications, a popular method to simplify the enormous search space of S-boxes and detail the most important results from the literature in this context. Finally, we identify some properties, which give information on the cost of implementing an S-box.

2.1 Cryptographic Properties

Notation An S-box is typically a balanced vectorial Boolean function \( F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m \), where each output \( y = F(x) \in \mathbb{F}_2^m \) is equiprobable for all inputs \( x \in \mathbb{F}_2^n \). Often, \( n = m \) and thus \( F \) is bijective. We denote the bits of \( x \in \mathbb{F}_2^n \) by \( x_i \) for \( i = 0 \ldots n - 1 \). An \( n \times m \) vectorial Boolean function can be split into \( m \) coordinate functions, each of which is a Boolean function \( f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2 \) for \( i = 0 \ldots m - 1 \). Let \( \circ \) denote the composition of functions, i.e. for \( F_1 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^l \) and \( F_2 : \mathbb{F}_2^l \rightarrow \mathbb{F}_2^m \): \( F_1 \circ F_2(x) = F_1(F_2(x)) \). We also consider the inner product of two bit-vectors as \( \langle x, y \rangle = \sum_i x_i y_i \).

Algebraic Normal Form (ANF). The algebraic normal form is a unique representation of a Boolean function \( f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2 \) as a multivariate polynomial:

\[
f(x) = \sum_{j \in \mathbb{F}_2^n} \alpha_j \left( \prod_{i=0}^{n-1} x_i^{j_i} \right)
\]  

Algebraic Degree. The algebraic degree of a Boolean function \( f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2 \) is the highest degree that occurs in the ANF. It can be described as

\[
\text{Degr}(f) = \max_{j \in \mathbb{F}_2^n, \alpha_j \neq 0} \text{HW}(j)
\]  

with \( \text{HW}(j) \) the Hamming weight of \( j \). The algebraic degree of a vectorial Boolean function \( F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m \) is the largest degree of its coordinate functions:

\[
\text{Degr}(F) = \max_{0 \leq i < m} \text{Degr}(f_i)
\]
The algebraic degree (and more generally complexity of the algebraic description) plays a role in the resistance against algebraic attacks [CP02], which target a cryptosystem by considering it as a system of equations. For an n-bit bijective S-box, the largest possible algebraic degree is $n - 1$.

**Differential Uniformity.** Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ be a vectorial Boolean function. We define its difference distribution table (DDT) [BS90] as $\delta_F$ with for $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$:

$$\delta_F(\alpha, \beta) = \#\{x \in \mathbb{F}_2^n : F(x \oplus \alpha) = F(x) \oplus \beta\}$$  \hspace{1cm} (4)

The differential uniformity [Nyb93] is the largest value in the DDT for $\alpha \neq 0$:

$$\text{Diff}(F) = \max_{\alpha \neq 0, \beta} \delta_F(\alpha, \beta)$$  \hspace{1cm} (5)

This metric indicates the difficulty of differential cryptanalysis [BS90], a statistical attack methodology which exploits the probability that some input difference propagates to some output difference through the cipher. The larger the value $\text{Diff}$, the less uniform the probabilities in $\delta_F$ are and thus, the less resistant a function is against differential cryptanalysis. The lower bound for the differential uniformity of bijective S-boxes is 2. The S-boxes that obtain this limit (and thus have DDT with only values 0 and 2) are called almost perfect nonlinear (APN).

**Linearity.** Another statistical cryptanalysis which is considered an important threat to symmetric-key cryptosystems is linear cryptanalysis [Mat93]. Instead of considering input-and output-differences of functions, this attack considers linear combinations of the bits of inputs and outputs. Similarly, we can define a property, which measures the resistance of functions against this type of attack. The two-dimensional Walsh spectrum of a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ is defined as:

$$\hat{F}(\alpha, \beta) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, x \rangle} \cdot (-1)^{\langle \beta, F(x) \rangle}$$  \hspace{1cm} (6)

for $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$. It can also be computed as a linear approximation table (LAT) [CV94]:

$$\hat{F}(\alpha, \beta) = 2\#\{x \in \mathbb{F}_2^n : \langle \alpha, x \rangle = \langle \beta, F(x) \rangle\} - 2^n$$  \hspace{1cm} (7)

The linearity is the largest absolute value in the LAT for $\beta \neq 0$:

$$\text{Lin}(F) = \max_{\beta \neq 0, \alpha} |\hat{F}(\alpha, \beta)|$$  \hspace{1cm} (8)

In some sense, the linearity measures how easy it is to approximate a function by a linear function. Naturally, the smaller this value, the better the resistance against linear cryptanalysis. The lower bound for the linearity of bijective S-boxes is $2^{(n+1)/2}$ and the S-boxes for which the linearity equals this limit are called almost bent (AB). It was shown that every AB function is also APN [CV94].

**The AES S-box.** To illustrate these properties, we take the AES S-box as example, which can be represented as a vectorial Boolean function over $\mathbb{F}_2^8$. This function has the maximum algebraic degree of 7. Its differential uniformity and linearity are respectively 4 and 32. While not AB nor APN, this S-box remains to this day the best 8-bit S-box in the literature in terms of cryptographic properties. No S-boxes with lower differential uniformity or linearity have been found and it is not clear whether they even exist. The main reason that we are still unsure about this is the magnitude of the search space.
2.2 Classifications

When looking for S-boxes with good properties, we deal with a dimensionality problem. The number of possible bijections on $n$ bits is $2^n!$, which prohibits exhaustive search for $n > 3$. To manage the enormous search spaces of S-boxes, we divide them into classes, defined based on an equivalence property.

**Affine Equivalence.** It has been shown that transforming the inputs and outputs of an S-box with an affine function preserves many of its cryptographic properties, including the algebraic degree, differential uniformity and linearity [CCZ98]. Following this observation, we can define an equivalence relation based on such transformations. We call two functions $F_1 : F_2^n \rightarrow F_2^m$ and $F_2 : F_2^n \rightarrow F_2^m$ affine equivalent [CCZ98] if and only if there exists a pair of $n$-bit and $m$-bit invertible affine bijections $A$ and $B$ such that $F_1 = B \circ F_2 \circ A$.

**Exhaustive Classifications.** The affine equivalence (AE) property has enabled the exhaustive classifications of entire function spaces up to a certain size. For these sizes, we therefore also have exhaustive knowledge of the cryptographic properties that exist. The first classification of Boolean functions dates back to 1959 [Gol59]. By 1972, all Boolean functions with up to five input bits were classified up to AE by Berlekamp and Welch [BW72]. Maiorana [Mai91] was the first to identify the AE classes of 6-bit Boolean functions. Using an efficient algorithm for verifying AE by Biryukov et al. [BDBP03], De Cannière [De 07] created an exhaustive classification of all 4-bit bijective S-boxes. The dimensionality reduction is significant, since the classification allows one to consider only 302 AE classes instead of $16!$ permutations. However, the search for the classes themselves becomes too complex for larger sizes. To this day, no exhaustive classification for vectorial Boolean functions over $n$ bits with $n > 4$ exists.

**Partial Classifications.** Classifications have been extended to S-box sizes $n > 4$ by restricting certain properties. Brinkmann and Leander [BL08] constrained the search space to bijective S-boxes with optimal properties (APN) and were able to classify them exhaustively up to dimension 5. Alternatively, Bozilov et al. [BBS17] were able to exhaustively classify all quadratic 5-bit permutations with a dedicated search method for functions of algebraic degree two. Following an enhancement of the AE algorithm by Biryukov et al. [BDBP03], De Meyer and Bilgin [DB19] were able to optimize the algorithm of Bozilov et al. [BBS17], which led to the first classification of quadratic 6-bit functions, including balanced non-bijective Boolean functions.

2.3 Implementation Properties

**Circuit Properties.** The cost of an S-box circuit can be expressed with many different metrics. We typically count the number of gates (gate complexity) or look at the circuit depth. Depending on which gates we consider, we can obtain different cost estimations. Stoffelen [Sto16] for example distinguishes gate complexity for hardware implementations and bitslice gate complexity for software implementations. The former considers all types of gates which can be found in typical CMOS libraries (AND, OR, NOT, XOR, NAND, NOR, XNOR), while the latter only considers those for which a CPU instruction exists in most processors (AND, OR, NOT, XOR). The bitslice gate complexity can be used as an indicator of the speed of a software implementation, since each gate should map to one instruction. For hardware implementations, the gate complexity is related to the area of a circuit. For the latency of a circuit, we look at the circuit depth, which is the maximum number of gates on any path from an input to an output. Note that we typically only consider 2-input gates in these metrics for genericity and ease of comparison.
XOR vs. AND. Any function can be represented in terms of AND, XOR and NOT gates only, because these gates form a functionally complete set of operators. It is therefore common to consider only these gate complexities. Naturally, area or latency estimates based on gate counts are not exact, since each different type of instruction or gate has a different area or delay. Exact cost metrics can be obtained using gate-specific costs from a logic library, combined with distinct gate counts (AND gate complexity, XOR gate complexity, ...). In CMOS technology, a NAND gate consists of 4 transistors, while an XOR gate requires as much as 8. A linear function can thus be more expensive than a nonlinear function (in hardware). Traditionally, circuits and S-boxes have been optimized according to that philosophy. However, if we want efficient circuits for embedded systems exposed to side-channel attacks, we need to consider the cost of countermeasures such as masking. A masked XOR requires $d + 1$ regular XOR gates, whereas a masked AND requires about $(d + 1)^2$ AND gates and $2d(d + 1)$ XOR gates. It is therefore common to regard the cost of XOR negligible compared to that of AND.

Multiplicative Complexity. As a result, recent works often consider the metric multiplicative complexity ($MC$) [Sch88]. This is the minimal number of 2-input AND gates required to evaluate a function over the basis (AND, XOR, NOT). The $MC$ is an important metric for the area of masked hardware implementations and for the latency of masked software implementations. Note that it is a property of a function, not of a circuit and that it corresponds to the AND gate complexity of the most efficient implementation of that function, with respect to AND gate count.

Multiplicative Depth. For the latency of masked hardware implementations, we care about the circuit depth in terms of 2-input AND gates. In masked implementations, every layer of AND gates requires a register stage for synchronization [NRR06, RBN+15], which significantly affects the latency in terms of clock cycles. Given a circuit over the basis (AND, XOR, NOT), the multiplicative depth ($MD$) is the maximum number of 2-input AND gates on any path from an input to an output. This is a circuit-specific property. For any S-box $S$, the minimal multiplicative depth achievable follows directly from its algebraic degree: $MD \geq \lceil \log_2(\text{Degr}(S)) \rceil$.

Affine Equivalence. Interestingly and conveniently, the multiplicative complexity $MC$ is also invariant under AE, since affine transformations do not alter the number of AND gates.

3 Towards Cryptography Design for Masking

The consideration of implementation cost in the design of cryptographic components (and more specifically S-boxes) is not new. The S-boxes of the Data Encryption Standard (DES) were chosen in the first place according to a list of cryptographic criteria. Among those that fulfilled these criteria, the designers chose the ones that would be most efficiently implemented in hardware [MM82]. Daemen and Rijmen [DR98] pointed out that the coefficients in the MixColumns operation of AES were specifically chosen with implementation efficiency in mind. The S-box is based on an inversion operation, which means that hardware implementations for encryption and decryption can share the same inversion block.

However, as we move towards a world with more and more embedded devices, where side-channel attacks are a constant threat, we must shift our understanding of implementation cost to one that takes SCA countermeasures into account. In fact, since those countermeasures come with such large overheads, the consideration of implementation cost in the design process becomes even more important than before. The ongoing NIST
lightweight cryptography standardization contest even explicitly lists this as a requirement for candidates:

“While implementations will not be required to provide side-channel resistance, the ability to provide it easily and at low cost is highly desired.”

The same is stated for resistance against fault attacks. To achieve this, we need designers to become familiar with how their decisions influence the cost.

In this section, we will first identify important goals for the designer and properties to optimize based on the cost of masked implementations. Next, we will discuss recent trends in the state-of-the-art on cipher design and, in particular, assess how the NIST lightweight candidates comply with the SCA requirement.

3.1 Goals and Trade-offs

**Decomposability.** There exist many masked implementations of the AES S-box [BGN+14, DRB+16, GMK17, DRB18]. In each case, the S-box of algebraic degree 7 is decomposed into quadratic components, which can each be masked using, for example, an ISW multiplication [ISW03]. This approach is very popular in the masking of S-boxes. Bilgin et al. [BNN+12] were able to create threshold implementations (TI) [NRR06] for all 3- and 4-bit S-boxes up to AE by decomposing the cubic S-boxes into quadratic ones. A beneficial property for an S-box is therefore to be easily decomposable into quadratic or low-degree functions.

**Minimize Multiplicative Depth.** Glitches in hardware masked implementations require that all quadratic stages are separated by registers for synchronization. The number of decomposition functions therefore plays an important role. Ideally, it should be possible to implement the S-box with the minimal multiplicative depth. S-boxes that are not designed with this specification in mind, often need more than that to keep the AND gate complexity within reasonable bounds. For example, most masked AES S-boxes [BGN+14, DRB+16, GMK17] require at least four instead of three ($=\lceil \log_2(7) \rceil$) quadratic steps. The number of register stages mostly influences the latency of hardware implementations, because it directly determines the number of clock cycles. However, also the area footprint is affected by those registers, which have relatively high cost compared to combinational logic on Application-Specific Integrated Circuits (ASICs).

**Algebraic Degree.** If the S-box is indeed chosen so that it can be implemented with minimal multiplicative depth $MD = \lceil \log_2(\text{Degr}(S)) \rceil$, then the algebraic degree becomes a direct indicator for the latency of the S-box. Naturally, there is a trade-off with cryptographic quality. Quadratic functions tend to have large differential uniformity and linearity. Finding the optimal trade-off is difficult. For S-boxes only, the AE classifications help to find the cryptographically strongest functions at the lowest cost. A larger investigation and comparison for S-boxes of many more sizes were made by Bilgin et al. [BDD+20]. However, the cryptographic strength of an S-box alone is not directly linked to that of the cipher, since it depends also on the linear layers and the number of rounds. Similarly, the latency of the entire cipher depends on the latency of the S-box, the number of rounds and the architecture used (see Figure 1). We should thus attempt to minimize the total multiplicative depth or algebraic degree of the cipher.

**Minimize Multiplicative Complexity.** In software, the depth has little importance since all operations are performed sequentially. The number of instructions in a masked implementation grows most with the number of AND operations. Hence, for software-oriented ciphers, it is most important to design primitives with low multiplicative complexity. Also
Figure 1: Illustration of the complicated trade-offs between cryptographic strength and latency.

in hardware, the multiplicative complexity is important for the area footprint of the S-box. However, recall that a low multiplicative complexity should not be achieved at the cost of a large multiplicative depth, even if area is more important than latency. Hence, in this case, the goal is to find S-boxes that have low level-$D$ multiplicative complexity, where $D$ is ideally the minimal multiplicative depth $MD = \lceil \log_2(\text{Degr}(S)) \rceil$.

The Inverse. Often, when an encryption uses the S-box $S$, its inverse $S^{-1}$ is required for decryption. The cost of the inverse is not always considered, because the cryptographic properties Diff and Lin are the same for $S$ and $S^{-1}$. The algebraic degree and multiplicative complexity, on the other hand, are not, which means considering only the implementation cost of $S$ may result in an expensive $S^{-1}$. In the survey of S-boxes of Bilgin et al. [BDD+20], the cryptographic and implementation properties of S-boxes and their inverses are investigated. Moreover, they consider also the possibility of sharing resources between encryption and decryption. The AES S-box, for example, uses an inversion, which is naturally an involution. Hence both encryption and decryption can use the same hardware components, which reduces the area footprint on a device that needs to be able to do both. Other than involutions, Bilgin et al. [BDD+20] identify several ways to minimize the combined area of $S$ and $S^{-1}$ and propose a selection of S-boxes that perform well in this regard, as well as cryptanalytically. Many NIST candidates avoid this issue by using a mode which only requires the forward S-box $S$.

Bit Sizes: Large vs. Small. AES is one of the few block ciphers that uses an 8-bit S-box. Most block ciphers use a 4-bit S-box. There are two ways to look at the choice of S-box size: from a cryptanalytic point-of-view and from a SCA point-of-view. The trade-off between cryptographic strength and implementation cost of small and large S-boxes is again complicated by the involvement of the linear layers. Hence, we leave it to the cryptanalysts to investigate it. Nevertheless, it is probable that the popularity of small bit sizes (e.g. 4) is more due to the lack of knowledge on the search space of larger S-boxes than due to a qualitative advantage. In addition, the success of AES has suppressed other ciphers that use an 8-bit S-box. From a SCA point-of-view however, larger S-boxes may enjoy some benefit in LUT-based implementations. DPA [KJJ99] on AES requires $2^8 = 256$ hypotheses to be made on each 8-bit subkey. This number is directly determined by the size of the S-box (or more specifically, the number of input bits that each output bit depends on). In a similar cipher with 4-bit S-boxes, only $2^4 = 16$ hypotheses would have to be made per subkey. More generally, in a state of $B$ bits with $n$-bit S-boxes, a DPA attack requires $2^{n-\log_2 n} B$ hypotheses to recover the entire round key. Hence, very large S-box sizes could interfere with the divide and conquer strategy of SCA. The problem is that their search spaces are too large to explore.

Bit Sizes: Odd vs. Even. Another contrast in S-box sizes is that between odd and even. Traditionally, often S-boxes of size a power of two were chosen, because of the datapath width in processors. For hardware implementations or bitsliced software implementations,
Looking at the NIST Lightweight Candidates from a Masking Point-of-View

this restriction does not make sense, but still, it is challenging to fit an odd-sized S-box into a block cipher with state size a power of two (e.g., 128 or 256). As a result, even-sized S-boxes (mostly 4) dominate in the literature. However, both from a cryptanalytic and implementation perspective, odd-sized S-boxes show an advantage over even-sized ones. The results of Bilgin et al. [BDD+20] show that S-boxes of odd size $n$ achieve the same cryptographic strength as S-boxes of even size $n + 1$, but at lower cost. They are especially interesting when it comes to low latency applications, since for every odd size $n$, there exists at least one AE class of quadratic APN S-boxes. These are S-boxes with optimal cryptographic properties, that can be implemented at minimal latency.

Clarifying Example. Choosing an S-box according to these goals and preferences is easier said than done. In the end, the cryptographic strength of a cipher remains the most important decision factor. However, what these guidelines aim to do is give more clarity about the impact of certain design decisions. In some cases, there are many S-boxes that result in the same security properties. It is exactly then that the masked implementation cost should be taken into account. Let us look at the popular 4-bit S-boxes as an example. Since the quadratic ones do not provide good cryptographic properties, only cubic ones are used in block ciphers. They need a multiplicative depth of (at least) two. In terms of latency, such a decision is wasteful in a way, since with $MD = 2$, it is possible to implement a fourth-degree S-box with better cryptographic properties. In fact, Bilgin et al. [BDD+20] showed that even with $MD = 1$, 5-bit S-boxes obtain better cryptographic properties than 4-bit S-boxes with $MD = 2$. And with these better cryptographic properties, it is possible that the number of rounds can be reduced, which even further optimizes the latency of masked implementations.

3.2 Discussion of the State-of-the-Art.

We will now look at the literature from the last years and show that some first steps have been taken towards the above goals. We will also critically assess some candidates from the NIST lightweight competition, that claim to have taken the cost of side-channel countermeasures into account in the design process. Note that our expertise does not extend to cryptanalysis and that many of the discussed ciphers are relatively new, i.e., not as scrutinized and established as AES. We therefore limit our treatment to an evaluation of the implementation properties only and say nothing about the cryptographic strength.

Multi-party Computation. The link between masking and the field of multi-party computation (MPC) has been pointed out in multiple works [ISW03, NRR06, PR11]. Both areas use secret sharing, which causes nonlinear operations to be more expensive than linear ones. As a result, we can see recent efforts into the design of cryptographic primitives with low multiplicative complexity. Albrecht et al. [ARS+15] introduced a family of ciphers, called LowMC, which is intended to minimize both its multiplicative complexity and depth. The design is based on a substitution-permutation network (SPN) with 3-bit S-boxes of $MD = 1$. For AES-like security parameters, they repeat the SPN for 12 rounds, which results in a total multiplicative depth of 12. Albrecht et al. [AGR+16] also introduce MIMC, a very simple construction consisting only of key additions and the quadratic map $x \rightarrow x^3$ in a finite field $\mathbb{F}_q$ with $q$ prime or a power of two. This latter cipher focuses more on multiplicative complexity than depth. They need to repeat the round function 82 times to achieve AES-like security parameters, so their total multiplicative depth is actually much than that of AES.

Keccak and PRIMATES. In recent years, several primitives have been introduced that explicitly mentioned side-channel attacks as motivation for their S-box choice. Most
prominent is the Keccak family of sponge functions [BDPV11], which has been selected as the SHA-3 Cryptographic Hash standard by NIST. They use quadratic 5-bit S-boxes with a very low multiplicative complexity of 5. The round function is repeated 18 times or more (depending on the state size), which means Keccak can have multiplicative depth as low as 18. Another permutation that uses a 5-bit S-box is PRIMATEs by Andreeva et al. [ABB+14]. They chose an S-box from a quadratic 5-bit AB class, which results in optimal cryptographic properties at only slightly higher MC than the Keccak S-box. Moreover, the permutation only requires 6 or 12 iterations of the round function, which results in a very small multiplicative depth. We summarize these results in Table 1.

Table 1: Comparison of primitives in the state-of-the-art. We denote with $n$ and $B$ respectively the S-box size and block size.

<table>
<thead>
<tr>
<th>Primitive</th>
<th>$n/B$/#Rnds</th>
<th>S-box MC</th>
<th>S-box MD</th>
<th>MC/bit</th>
<th>Tot MD</th>
<th>Tot MC/bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES [DR98]</td>
<td>8/128/10</td>
<td>32</td>
<td>4</td>
<td>4</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>LowMC [ARS+15]</td>
<td>3/196/14</td>
<td>3</td>
<td>1</td>
<td>0.96*</td>
<td>14</td>
<td>13.5</td>
</tr>
<tr>
<td>Keccak [BDPV11]</td>
<td>5/200/18</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>PRIMATEs [ABB+14]</td>
<td>5/120/[6/12]</td>
<td>7</td>
<td>1</td>
<td>1.4</td>
<td>6/12</td>
<td>8.4/16.8</td>
</tr>
</tbody>
</table>

*LowMC does not apply S-boxes to the entire state

Length Increasing Structures. Another trend in the literature is to build large S-boxes from smaller ones, using length-increasing structures which are inspired by block cipher design. In particular, 8-bit S-boxes of this type have been used in several cryptographic primitives, including CRYPTON [Lim98], Khazad [BR00], Whirlpool [BR11], ICEBERG [SPR+04] and CLEFIA [SSA+07]. Comparative studies of such S-boxes, including new proposals, were made by Canteaut et al. [CDL15] and by Boss et al. [BGG+16]. By construction, these S-boxes are decomposable, since they are assembled from quadratic building blocks. Note however that their increased bit size does not increase the complexity of a DPA attack, because the hypotheses can be made about the smaller (e.g. 4-bit) subcomponents. Moreover, most of the S-boxes obtained in this way have quite a large MD and none so far achieve cryptographic properties as good as the AES S-box.

3.2.1 NIST Lightweight Competition.

Given the above-acquired knowledge on primitive design and given that the NIST lightweight competition explicitly states that the cost of SCA countermeasures should be taken into account, we now take a look at some of the Round 2 candidates.\(^2\)

Side-Channel Claims. Many (not all) candidates make a note about having considered side-channel attacks. However, this claim is often not very well-argued. In some cases, it is justified by the fact that the design uses “easy-to-mask” operations such as bitwise functions. While this is more convenient for the masking designer, it gives no guarantees about the total cost. Some proposals use existing primitives and use their lightweight property as justification. However, these primitives were not necessarily designed with SCA in mind. Other candidates use AES and argue that a lot of research exists on masking the AES. The many works on masking AES [BGN+14, DRB+16, GMK17, DRB18] indeed confirm that there is an abundance of available literature on the subject, but the existence of a lot of research does not imply that its results are most efficient. This holds especially for mask conversions between Boolean and arithmetic masking [BCZ18, CGTV15], which

\(^2\)Descriptions can all be found at https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates
are required for ARX ciphers. In the NIST proposal SPARKLE, the argumentation for side-channels is again that a large amount of research exists on this topic. However, whether these conversions are considered efficient is highly disputable.

A few of the candidates stand out in their treatment of SCA. Firstly, Goudarzi et al. describe and implement a masked version of their scheme Pyjamask in software. Secondly, the proposal ISAP by Dobraunig et al. is based on the ISAP mode of operation [DEM+17], which is a leakage-resilient mode of operation, designed to provide security against DPA by a re-keying mechanism. In contrast with masking, this countermeasure acts at the protocol level instead of at the algorithmic level. Other proposals that claim to use some form of leakage resilience are Xoodyak, Spook, Ascon, DryGASCON and Subterranean.

**Implementation Properties.** Since the claims on SCA are often badly motivated and since we need to be able to correctly compare different candidates, we collect some of their properties in Table 2. We limit our selection to primitives used in proposals that make some claim about the consideration of SCA and list several properties related to the multiplicative complexity and depth of the ciphers. Naturally, these properties should not be considered by themselves, as the cost of implementations depends on several of them jointly, and cryptographic strength is not taken into account here. Note also that some properties depend on the objective of the implementation. For example, the GIFT S-box can be implemented with 4 AND gates and depth 4 or with 5 AND gates and depth 2. The AES and SKINNY S-boxes have algebraic degree 8 and 6 respectively, which means one can implement both with depth 3. However, their multiplicative complexity at depth 4 is a lot smaller.

We recall the most important influences on the cost for different cases:

**Hardware with focus on low latency:** The latency of a (serial or round-based) masked implementation will depend strongly on the total multiplicative depth (Tot MD). We calculate this as the multiplicative depth of the S-box (S-box MD), multiplied with the number of rounds (# Rnds) in the primitive.\(^3\) Since different primitives operate on different state sizes, we also calculate the total multiplicative depth per bit (Tot MD/bit) by dividing the total MD by the block size \(B\).\(^4\)

**Hardware with focus on small area:** The area cost of masked hardware implementations comes from registers on the one hand and combinational logic on the other. On ASIC devices, the registers are quite expensive, whereas on Field Programmable Gate Array (FPGA) devices, they are relatively cheap from being available in large quantities. The register cost is considerably affected by the block or state size \(B\), especially in a serial implementation. Furthermore, also the multiplicative depth of the S-box (S-box MD) contributes to the registers, but this is more prominent in round-based implementations. As for the combinational logic, its area grows most with the multiplicative complexity of the S-box (S-box MC). Again, to account for the scalability with the number of bits being operated on, we also calculate the multiplicative complexity per bit (MC/bit) as the S-box MC divided by the S-box size \(n\). The number of rounds is not important for the area of a serial or round-based implementation.

**Software with focus on low latency:** In software, the speed can be approximated by the number of instructions. For masked software implementations, we care most about the total multiplicative complexity, i.e. the multiplicative complexity of the S-box,

---

\(^3\)We note that some sponge-based proposals use a higher number of rounds during the initialization phase of a mode. Since asymptotically only the rounds per plaintext block matter, we will not consider initialization rounds here. Note however that for short messages, the initialization rounds will be dominant.

\(^4\)When a primitive is used in a sponge construction, we divide by the rate \(r\), since this indicates the number of plaintext bits being processed per iteration.
multiplied with the number of rounds. We also consider the same metric scaled per bit (Tot MC/bit) by multiplying MC/bit with the number of rounds (# Rnds).

We note that linear operations are not entirely negligible, especially when the masking order $d$ is not very high, but since their cost is typically taken into account in the design of unmasked primitives, we do not consider them here.

**Observations.** The existence of Table 2 is immediately justified by the large variability in some of its columns. We make some interesting observations here.

- The popularity of 4-bit S-boxes continues. It is clear that they systematically result in S-box $MD = 2$. However, many proposals use odd-sized S-boxes, which achieve the minimal depth of one.

- The $MC$/bit is almost identical for all non-AES proposals and there is little to no need for improvement in that aspect.

- The largest contrasts arise from differences in the number of rounds, which plays an important role when speed is a priority. With respect to the metrics of total multiplicative complexity or depth, we see that several primitives are not competitive with AES (e.g. Spongent, Skinny, . . .), due to a large number of rounds. We note that this design choice is also highly dependent on the designer’s choice of security margin.

- **On Sponge Constructions:**
  - On the one hand, sponge constructions often use a larger number of initialization rounds than the number of rounds used per plaintext block. This is beneficial for the speed of a hardware implementation.
  - On the other hand, permutations in a sponge construction typically require a larger state size than block ciphers where the message is the entire state. Large state sizes are bad for area requirements on ASIC devices.
  - We also see that the throughput of software implementations (Tot MC/bit) is badly affected by the fact that only $r$ of the $B$ bits of the state are processed. For example, the Tot MC/bit of PHOTON and Shadow are worse than that of Clyde, despite having equivalent S-box properties and number of rounds.

- In terms of throughput of masked software implementations (Tot MC/bit), the frontrunners are Pyjamask, Clyde and Subterranean.

- In terms of throughput of masked hardware implementations (Tot $MD$/bit), the frontrunners are (G)ASCON, Keccak and Subterranean.

**A Note on Leakage Resilience.** We note that Table 2 considers only the internal building blocks of the NIST proposals, regardless of whether they are used in a leakage resilient mode or not. We see this as a necessary first step for comparison. Moreover, since different candidates rely on different types of leakage resilience, making a more detailed comparison is challenging. For an investigation into the leakage resilience of several candidates, we refer to the work of Bellizia et al. [BBC+20].

---

5 For primitives in a sponge construction, we have to multiply this with the state size $B$ and divide by the rate $r$. 
Table 2: Comparison of NIST candidates. \((n = \text{S-box size}, B = \text{block size or permutation state size}, r = \text{rate for Sponge})\)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>(n)</th>
<th>(B)</th>
<th>(r)</th>
<th># Rnds</th>
<th>S-box MC</th>
<th>MC/bit</th>
<th>Tot MC/bit</th>
<th>S-box MD</th>
<th>Tot MD</th>
<th>Tot MD/bit</th>
<th>Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOOODOO</td>
<td>3</td>
<td>384</td>
<td>128</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>36</td>
<td>1</td>
<td>12</td>
<td>0.09375</td>
<td>Xoodyak</td>
</tr>
<tr>
<td>Pyjamask</td>
<td>3/4</td>
<td>96/128</td>
<td>14</td>
<td>3</td>
<td>3/4</td>
<td>1</td>
<td>14</td>
<td>1/2</td>
<td>14/28</td>
<td>0.146</td>
<td>Pyjamask</td>
</tr>
<tr>
<td>Clyde</td>
<td>4</td>
<td>128</td>
<td>12</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>12</td>
<td>2</td>
<td>24</td>
<td>0.1875</td>
<td>Spook</td>
</tr>
<tr>
<td>GIFT (HW)</td>
<td>4</td>
<td>64/128</td>
<td>28/40</td>
<td>5</td>
<td>1.25</td>
<td></td>
<td>35/50</td>
<td>2</td>
<td>56/80</td>
<td>0.625</td>
<td>ESTATE, GIFT-COFB, HYENA, LOTUS/LOCUS, SUNDAE-GIFT</td>
</tr>
<tr>
<td>GIFT (SW)</td>
<td>4</td>
<td>64/128</td>
<td>28/40</td>
<td>4</td>
<td>1</td>
<td></td>
<td>28/40</td>
<td>4</td>
<td>112/160</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>KNOT</td>
<td>4</td>
<td>256</td>
<td>64</td>
<td>28*</td>
<td>4</td>
<td>1</td>
<td>112*</td>
<td>2</td>
<td>56*</td>
<td>0.875</td>
<td>KNOT</td>
</tr>
<tr>
<td>PHOTON</td>
<td>4</td>
<td>256</td>
<td>32/128</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>96/24</td>
<td>2</td>
<td>24</td>
<td>0.1875</td>
<td>PHOTON-Beetle</td>
</tr>
<tr>
<td>Shadow</td>
<td>4</td>
<td>512</td>
<td>256</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>24</td>
<td>2</td>
<td>24</td>
<td>0.09375</td>
<td>Spook</td>
</tr>
<tr>
<td>Spongent</td>
<td>4</td>
<td>160/176</td>
<td>80/90</td>
<td>5</td>
<td>1.25</td>
<td></td>
<td>100</td>
<td>2</td>
<td>160/180</td>
<td>1</td>
<td>Elephant</td>
</tr>
<tr>
<td>ForkSkinny</td>
<td>4/8</td>
<td>64/128</td>
<td>40/48</td>
<td>4</td>
<td>1</td>
<td>40/48</td>
<td>80/192</td>
<td>2</td>
<td>80/192</td>
<td>1.25</td>
<td>ForkAE</td>
</tr>
<tr>
<td>ASCON</td>
<td>5</td>
<td>320</td>
<td>64/128</td>
<td>6*</td>
<td>5</td>
<td>1</td>
<td>30*</td>
<td>1</td>
<td>6*</td>
<td>0.0625</td>
<td>Ascon, ISAP</td>
</tr>
<tr>
<td>GASCON</td>
<td>5</td>
<td>320</td>
<td>128</td>
<td>7*</td>
<td>5</td>
<td>1</td>
<td>17.5*</td>
<td>1</td>
<td>7*</td>
<td>0.055</td>
<td>DryGASCON</td>
</tr>
<tr>
<td>Keccak</td>
<td>5</td>
<td>200</td>
<td>18</td>
<td>5</td>
<td>1</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>0.09</td>
<td>Elephant</td>
</tr>
<tr>
<td>AES</td>
<td>8</td>
<td>128</td>
<td></td>
<td>8*</td>
<td>5</td>
<td>1</td>
<td>22.2*</td>
<td>1</td>
<td>8*</td>
<td>0.056</td>
<td>ISAP</td>
</tr>
<tr>
<td>Skinny</td>
<td>8</td>
<td>128</td>
<td></td>
<td>8</td>
<td>40</td>
<td>10</td>
<td>40</td>
<td>4</td>
<td>40</td>
<td>0.3125</td>
<td>ESTATE, mixFEED, SAEAES</td>
</tr>
<tr>
<td>GIMLI</td>
<td>96</td>
<td>384</td>
<td>128</td>
<td>24</td>
<td>96</td>
<td>1</td>
<td>72</td>
<td>1</td>
<td>24</td>
<td>0.1875</td>
<td>Gimli</td>
</tr>
<tr>
<td>Subterranean 2.0</td>
<td>257</td>
<td>257</td>
<td>32</td>
<td>1*</td>
<td>257</td>
<td>1</td>
<td>8*</td>
<td>1</td>
<td>1*</td>
<td>0.03125</td>
<td>Subterranean 2.0</td>
</tr>
</tbody>
</table>

* Given a larger number of initialization rounds
Conclusion

In this work, we want to clarify the effect of some design decisions on the cost of masked implementations of symmetric primitives. In particular, our guidelines should demonstrate that 4-bit S-boxes, despite being the most popular in symmetric designs, do not exhibit the most beneficial properties in this context. We noted for each implementation goal (software vs. hardware, low area vs. low latency) which properties matter the most. Finally, we have created the first comparison of the NIST candidates based on those properties. While our analysis says nothing on the cryptanalytic security of these proposals, we believe these costs should be taken into account in the choice of candidates for standardization.

Acknowledgements

This work was supported in part by the NIST Research Grant 60NANB15D346. The author is funded by a PhD fellowship of the FWO. The author would like to thank François-Xavier Standaert and Thomas Peyrin for their feedback.

References


Looking at the NIST Lightweight Candidates from a Masking Point-of-View


